ABDK CONSULTING

SMART CONTRACT AUDIT

Voltz

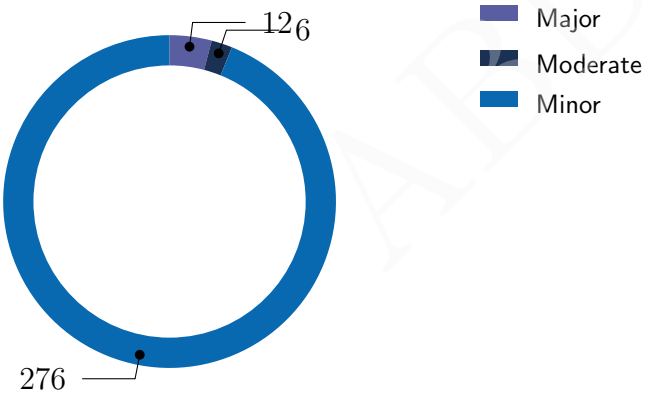
Solidity

abdk.consulting

SMART CONTRACT AUDIT CONCLUSION

by Mikhail Vladimirov and Dmitry Khovratovich
25th April 2022

We've been asked to review 40 files in a Github repository. We found 12 major and a few less important issues. All identified major issues have been fixed or otherwise addressed in collaboration with the client.



Findings

ID	Severity	Category	Status
CVF-1	Minor	Procedural	Info
CVF-2	Minor	Suboptimal	Fixed
CVF-3	Minor	Suboptimal	Fixed
CVF-4	Minor	Bad datatype	Fixed
CVF-5	Minor	Suboptimal	Info
CVF-6	Minor	Suboptimal	Info
CVF-7	Minor	Suboptimal	Fixed
CVF-8	Minor	Procedural	Fixed
CVF-9	Minor	Bad datatype	Fixed
CVF-10	Minor	Bad datatype	Fixed
CVF-11	Minor	Unclear behavior	Fixed
CVF-12	Minor	Procedural	Fixed
CVF-13	Major	Unclear behavior	Fixed
CVF-14	Minor	Unclear behavior	Fixed
CVF-15	Minor	Suboptimal	Fixed
CVF-16	Minor	Procedural	Fixed
CVF-17	Minor	Unclear behavior	Fixed
CVF-18	Moderate	Unclear behavior	Fixed
CVF-19	Minor	Readability	Info
CVF-20	Minor	Bad datatype	Fixed
CVF-21	Minor	Unclear behavior	Info
CVF-22	Minor	Readability	Fixed
CVF-23	Minor	Suboptimal	Fixed
CVF-24	Minor	Overflow/Underflow	Fixed
CVF-25	Minor	Unclear behavior	Info
CVF-26	Minor	Suboptimal	Info
CVF-27	Minor	Suboptimal	Fixed

ID	Severity	Category	Status
CVF-28	Minor	Suboptimal	Fixed
CVF-29	Minor	Suboptimal	Fixed
CVF-30	Minor	Bad naming	Fixed
CVF-31	Major	Suboptimal	Fixed
CVF-32	Major	Unclear behavior	Fixed
CVF-33	Major	Unclear behavior	Fixed
CVF-34	Minor	Suboptimal	Fixed
CVF-35	Minor	Suboptimal	Fixed
CVF-36	Minor	Readability	Info
CVF-37	Minor	Suboptimal	Fixed
CVF-38	Minor	Suboptimal	Fixed
CVF-39	Minor	Suboptimal	Fixed
CVF-40	Minor	Readability	Fixed
CVF-41	Minor	Bad datatype	Fixed
CVF-42	Minor	Bad datatype	Fixed
CVF-43	Minor	Bad datatype	Fixed
CVF-44	Minor	Documentation	Fixed
CVF-45	Minor	Documentation	Fixed
CVF-46	Minor	Unclear behavior	Fixed
CVF-47	Minor	Procedural	Info
CVF-48	Minor	Suboptimal	Fixed
CVF-49	Minor	Bad datatype	Fixed
CVF-50	Minor	Bad datatype	Fixed
CVF-51	Minor	Suboptimal	Fixed
CVF-52	Minor	Suboptimal	Info
CVF-53	Minor	Suboptimal	Fixed
CVF-54	Minor	Bad datatype	Fixed
CVF-55	Minor	Unclear behavior	Info
CVF-56	Minor	Suboptimal	Info
CVF-57	Minor	Bad datatype	Fixed

ID	Severity	Category	Status
CVF-58	Minor	Bad datatype	Fixed
CVF-59	Minor	Suboptimal	Fixed
CVF-60	Minor	Procedural	Info
CVF-61	Minor	Procedural	Info
CVF-62	Minor	Suboptimal	Fixed
CVF-63	Minor	Suboptimal	Fixed
CVF-64	Minor	Bad datatype	Fixed
CVF-65	Minor	Suboptimal	Info
CVF-66	Minor	Suboptimal	Fixed
CVF-67	Minor	Suboptimal	Fixed
CVF-68	Minor	Suboptimal	Fixed
CVF-69	Minor	Unclear behavior	Fixed
CVF-70	Moderate	Flaw	Info
CVF-71	Major	Flaw	Fixed
CVF-72	Minor	Suboptimal	Fixed
CVF-73	Minor	Unclear behavior	Fixed
CVF-74	Minor	Suboptimal	Fixed
CVF-75	Minor	Procedural	Fixed
CVF-76	Minor	Procedural	Info
CVF-77	Minor	Suboptimal	Fixed
CVF-78	Minor	Readability	Info
CVF-79	Minor	Suboptimal	Fixed
CVF-80	Minor	Flaw	Info
CVF-81	Major	Suboptimal	Fixed
CVF-82	Minor	Suboptimal	Fixed
CVF-83	Minor	Suboptimal	Fixed
CVF-84	Minor	Suboptimal	Fixed
CVF-85	Minor	Unclear behavior	Info
CVF-86	Minor	Unclear behavior	Info
CVF-87	Minor	Overflow/Underflow	Fixed

ID	Severity	Category	Status
CVF-88	Minor	Suboptimal	Fixed
CVF-89	Minor	Suboptimal	Fixed
CVF-90	Minor	Bad naming	Fixed
CVF-91	Minor	Suboptimal	Fixed
CVF-92	Minor	Procedural	Fixed
CVF-93	Minor	Procedural	Fixed
CVF-94	Minor	Procedural	Fixed
CVF-95	Minor	Procedural	Fixed
CVF-96	Minor	Unclear behavior	Fixed
CVF-97	Minor	Suboptimal	Fixed
CVF-98	Minor	Bad datatype	Fixed
CVF-99	Minor	Bad datatype	Fixed
CVF-100	Minor	Procedural	Fixed
CVF-101	Minor	Bad naming	Fixed
CVF-102	Minor	Procedural	Fixed
CVF-103	Minor	Unclear behavior	Info
CVF-104	Minor	Unclear behavior	Fixed
CVF-105	Minor	Unclear behavior	Info
CVF-106	Minor	Procedural	Info
CVF-107	Minor	Procedural	Fixed
CVF-108	Minor	Suboptimal	Fixed
CVF-109	Minor	Suboptimal	Fixed
CVF-110	Minor	Bad datatype	Fixed
CVF-111	Minor	Suboptimal	Fixed
CVF-112	Minor	Bad naming	Fixed
CVF-113	Minor	Readability	Fixed
CVF-114	Minor	Suboptimal	Info
CVF-115	Minor	Overflow/Underflow	Fixed
CVF-116	Minor	Suboptimal	Info
CVF-117	Minor	Readability	Fixed

ID	Severity	Category	Status
CVF-118	Minor	Readability	Fixed
CVF-119	Minor	Overflow/Underflow	Fixed
CVF-120	Minor	Overflow/Underflow	Fixed
CVF-121	Minor	Unclear behavior	Fixed
CVF-122	Minor	Suboptimal	Fixed
CVF-123	Minor	Readability	Fixed
CVF-124	Minor	Documentation	Fixed
CVF-125	Minor	Readability	Info
CVF-126	Minor	Procedural	Info
CVF-127	Minor	Bad naming	Fixed
CVF-128	Minor	Readability	Fixed
CVF-129	Minor	Readability	Fixed
CVF-130	Minor	Suboptimal	Fixed
CVF-131	Minor	Unclear behavior	Fixed
CVF-132	Minor	Suboptimal	Fixed
CVF-133	Minor	Readability	Fixed
CVF-134	Minor	Suboptimal	Fixed
CVF-135	Minor	Suboptimal	Fixed
CVF-136	Minor	Suboptimal	Fixed
CVF-137	Minor	Suboptimal	Fixed
CVF-138	Minor	Suboptimal	Fixed
CVF-139	Minor	Suboptimal	Fixed
CVF-140	Minor	Suboptimal	Fixed
CVF-141	Minor	Suboptimal	Info
CVF-142	Minor	Suboptimal	Info
CVF-143	Major	Overflow/Underflow	Fixed
CVF-144	Major	Overflow/Underflow	Fixed
CVF-145	Minor	Procedural	Fixed
CVF-146	Minor	Readability	Fixed
CVF-147	Minor	Readability	Fixed

ID	Severity	Category	Status
CVF-148	Minor	Suboptimal	Info
CVF-149	Minor	Suboptimal	Fixed
CVF-150	Minor	Suboptimal	Fixed
CVF-151	Minor	Suboptimal	Fixed
CVF-152	Minor	Suboptimal	Fixed
CVF-153	Minor	Bad naming	Fixed
CVF-154	Minor	Suboptimal	Fixed
CVF-155	Minor	Suboptimal	Fixed
CVF-156	Minor	Suboptimal	Fixed
CVF-157	Moderate	Unclear behavior	Fixed
CVF-158	Minor	Suboptimal	Fixed
CVF-159	Minor	Bad datatype	Fixed
CVF-160	Minor	Readability	Info
CVF-161	Minor	Bad datatype	Fixed
CVF-162	Minor	Documentation	Info
CVF-163	Minor	Suboptimal	Fixed
CVF-164	Minor	Suboptimal	Info
CVF-165	Minor	Suboptimal	Info
CVF-166	Minor	Suboptimal	Info
CVF-167	Minor	Bad datatype	Fixed
CVF-168	Minor	Procedural	Fixed
CVF-169	Minor	Procedural	Fixed
CVF-170	Minor	Documentation	Fixed
CVF-171	Minor	Overflow/Underflow	Fixed
CVF-172	Minor	Suboptimal	Fixed
CVF-173	Minor	Unclear behavior	Fixed
CVF-174	Minor	Suboptimal	Fixed
CVF-175	Minor	Suboptimal	Info
CVF-176	Minor	Unclear behavior	Info
CVF-177	Minor	Procedural	Info

ID	Severity	Category	Status
CVF-178	Minor	Bad naming	Fixed
CVF-179	Minor	Readability	Fixed
CVF-180	Minor	Suboptimal	Info
CVF-181	Minor	Bad datatype	Fixed
CVF-182	Minor	Suboptimal	Info
CVF-183	Minor	Unclear behavior	Fixed
CVF-184	Minor	Bad datatype	Fixed
CVF-185	Minor	Suboptimal	Info
CVF-186	Minor	Documentation	Info
CVF-187	Minor	Suboptimal	Info
CVF-188	Minor	Bad datatype	Fixed
CVF-189	Minor	Suboptimal	Fixed
CVF-190	Minor	Suboptimal	Fixed
CVF-191	Minor	Suboptimal	Fixed
CVF-192	Minor	Suboptimal	Fixed
CVF-193	Moderate	Overflow/Underflow	Fixed
CVF-194	Major	Flaw	Fixed
CVF-195	Minor	Suboptimal	Fixed
CVF-196	Minor	Procedural	Info
CVF-197	Minor	Overflow/Underflow	Fixed
CVF-198	Minor	Suboptimal	Fixed
CVF-199	Minor	Procedural	Fixed
CVF-200	Minor	Readability	Info
CVF-201	Minor	Suboptimal	Info
CVF-202	Minor	Bad datatype	Info
CVF-203	Minor	Suboptimal	Info
CVF-204	Minor	Unclear behavior	Info
CVF-205	Minor	Procedural	Fixed
CVF-206	Minor	Suboptimal	Fixed
CVF-207	Minor	Suboptimal	Fixed

ID	Severity	Category	Status
CVF-208	Minor	Procedural	Fixed
CVF-209	Minor	Suboptimal	Fixed
CVF-210	Minor	Suboptimal	Info
CVF-211	Minor	Suboptimal	Info
CVF-212	Minor	Procedural	Fixed
CVF-213	Minor	Suboptimal	Fixed
CVF-214	Minor	Readability	Fixed
CVF-215	Minor	Suboptimal	Fixed
CVF-216	Minor	Suboptimal	Fixed
CVF-217	Minor	Procedural	Fixed
CVF-218	Minor	Suboptimal	Info
CVF-219	Minor	Suboptimal	Fixed
CVF-220	Minor	Overflow/Underflow	Info
CVF-221	Minor	Bad datatype	Info
CVF-222	Major	Overflow/Underflow	Info
CVF-223	Minor	Procedural	Info
CVF-224	Minor	Procedural	Fixed
CVF-225	Minor	Procedural	Info
CVF-226	Minor	Unclear behavior	Fixed
CVF-227	Minor	Suboptimal	Info
CVF-228	Minor	Suboptimal	Fixed
CVF-229	Minor	Suboptimal	Fixed
CVF-230	Minor	Suboptimal	Info
CVF-231	Minor	Suboptimal	Fixed
CVF-232	Moderate	Overflow/Underflow	Fixed
CVF-233	Minor	Overflow/Underflow	Info
CVF-234	Minor	Suboptimal	Fixed
CVF-235	Minor	Suboptimal	Fixed
CVF-236	Major	Suboptimal	Fixed
CVF-237	Minor	Suboptimal	Info

ID	Severity	Category	Status
CVF-238	Minor	Bad naming	Info
CVF-239	Minor	Suboptimal	Info
CVF-240	Minor	Bad naming	Info
CVF-241	Major	Suboptimal	Fixed
CVF-242	Minor	Suboptimal	Fixed
CVF-243	Minor	Suboptimal	Info
CVF-244	Minor	Bad naming	Info
CVF-245	Minor	Suboptimal	Fixed
CVF-246	Minor	Suboptimal	Fixed
CVF-247	Minor	Documentation	Fixed
CVF-248	Minor	Documentation	Fixed
CVF-249	Minor	Bad datatype	Fixed
CVF-250	Minor	Documentation	Fixed
CVF-251	Minor	Documentation	Fixed
CVF-252	Minor	Bad naming	Info
CVF-253	Minor	Bad naming	Info
CVF-254	Minor	Bad naming	Fixed
CVF-255	Minor	Suboptimal	Fixed
CVF-256	Minor	Bad datatype	Fixed
CVF-257	Minor	Documentation	Fixed
CVF-258	Minor	Bad datatype	Fixed
CVF-259	Minor	Procedural	Fixed
CVF-260	Minor	Documentation	Fixed
CVF-261	Moderate	Procedural	Fixed
CVF-262	Minor	Documentation	Fixed
CVF-263	Minor	Suboptimal	Info
CVF-264	Minor	Suboptimal	Info
CVF-265	Minor	Suboptimal	Info
CVF-266	Minor	Suboptimal	Info
CVF-267	Minor	Procedural	Info

ID	Severity	Category	Status
CVF-268	Minor	Suboptimal	Info
CVF-269	Minor	Procedural	Info
CVF-270	Minor	Unclear behavior	Fixed
CVF-271	Minor	Procedural	Info
CVF-272	Minor	Bad datatype	Fixed
CVF-273	Minor	Procedural	Fixed
CVF-274	Minor	Bad naming	Fixed
CVF-275	Minor	Bad datatype	Fixed
CVF-276	Minor	Bad datatype	Fixed
CVF-277	Minor	Bad datatype	Fixed
CVF-278	Minor	Bad datatype	Fixed
CVF-279	Minor	Bad datatype	Fixed
CVF-280	Minor	Bad datatype	Fixed
CVF-281	Minor	Unclear behavior	Fixed
CVF-282	Minor	Bad datatype	Fixed
CVF-283	Minor	Bad datatype	Fixed
CVF-284	Minor	Bad datatype	Fixed
CVF-285	Minor	Bad datatype	Fixed
CVF-286	Minor	Bad datatype	Fixed
CVF-287	Minor	Bad datatype	Fixed
CVF-288	Minor	Bad datatype	Fixed
CVF-289	Minor	Bad datatype	Fixed
CVF-290	Minor	Bad datatype	Fixed
CVF-291	Minor	Bad datatype	Fixed

Contents

1 Document properties

Version

Version	Date	Author	Description
0.1	April 25, 2022	D. Khovratovich	Initial Draft
0.2	April 25, 2022	D. Khovratovich	Minor revision
1.0	April 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:

- aave/AaveDataTypes.sol
- core_libraries/FixedAndVariableMath.sol
- core_libraries/MarginCalculator.sol
- core_libraries/Position.sol
- core_libraries/SafeTransferLib.sol
- core_libraries/SwapMath.sol
- core_libraries/Tick.sol
- core_libraries/TickBitmap.sol
- core_libraries/Time.sol
- core_libraries/TraderWithYieldBearingAssets.sol
- interfaces/aave/IAaveV2LendingPool.sol
- interfaces/aave/IAToken.sol
- interfaces/rate_oracles/IAaveRateOracle.sol
- interfaces/rate_oracles/IRateOracle.sol
- interfaces/IERC20Minimal.sol
- interfaces/IFactory.sol
- interfaces/IFCM.sol
- interfaces/IMarginEngine.sol
- interfaces/IPositionStructs.sol
- interfaces/IVAMM.sol
- periphery/peripheral_libraries/LiquidityAmounts.sol
- periphery/Periphery.sol
- rate_oracles/AaveRateOracle.sol
- rate_oracles/BaseRateOracle.sol

- rate_oracles/OracleBuffer.sol
- utils/BitMath.sol
- utils/Errors.sol
- utils/FixedPoint96.sol
- utils/FixedPoint128.sol
- utils/FullMath.sol
- utils/LiquidityMath.sol
- utils/SafeCast.sol
- utils/SqrtPriceMath.sol
- utils/TickMath.sol
- utils/UnsafeMath.sol
- utils/WayRayMath.sol
- AaveFCM.sol
- Factory.sol
- MarginEngine.sol
- VAMM.sol

The fixes were provided in a pull request.

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** Info
- **Category** Procedural
- **Source** MarginEngine.sol

Description We didn't review this file.

Listing 1:

```
13 import "prb-math/contracts/PRBMathUD60x18.sol";
```

3.2 CVF-2

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** MarginEngine.sol

Description This variable is never read.
Recommendation Consider removing it.

Listing 2:

```
55 address private deployer;
```

3.3 CVF-3

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** MarginEngine.sol

Description The value assigned here is never used.
Recommendation Consider removing the assignment.

Listing 3:

```
65 deployer = msg.sender; /// this is presumably the factory
```

3.4 CVF-4

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** MarginEngine.sol

Recommendation The type of the "_underlyingToken" argument should be "IERC20". The type of the "rateOracleAddress" argument should be "IRateOracle".

Listing 4:

```
69 function initialize(address _underlyingToken, address
   ↪ _rateOracleAddress, uint256 _termStartTimestampWad,
   ↪ uint256 _termEndTimestampWad) external override
   ↪ initializer {
```

3.5 CVF-5

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** MarginEngine.sol

Recommendation prefer to keep the sanity check, even though
Client Comment Prefer to keep the sanity check. CR: My take is that it's debatable whether or not it's worth guarding against 0x000..0000. There were certainly instances earlier in the life of Ethereum tooling where parameters would get missed off by mistake, and if that happens then 0x000...000 is a the default and almost always a sign that something's gone wrong.

Listing 5:

```
70 require(_underlyingToken != address(0), "UT must be set");
   require(_rateOracleAddress != address(0), "RO must be set");
```

3.6 CVF-6

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** MarginEngine.sol

Description There is no check that "_termStartTimestampWad" is less that "_termEndTimestampWad".
Recommendation Consider adding such check.

Listing 6:

```
69 function initialize(address _underlyingToken, address
   ↪ _rateOracleAddress, uint256 _termStartTimestampWad,
   ↪ uint256 _termEndTimestampWad) external override
   ↪ initializer {
```

3.7 CVF-7

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** MarginEngine.sol

Description even though zero is a valid timestamp, it
Recommendation Consider removing these checks.
Client Comment Even though 0 is a valid timestamp, i can't think of a scenario where it would be valid for our purposes.

Listing 7:

```
72 require(_termStartTimestampWad != 0, "TS must be set");
   require(_termEndTimestampWad != 0, "TE must be set");
```

3.8 CVF-8

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** MarginEngine.sol

Description These functions should emit some events.

Listing 8:

```
147 function setMarginCalculatorParameters(

154 function setVAMM(address _vAMMAddress) external override
      ↪ onlyOwner {

158 function setFCM(address _fcm) external override onlyOwner {
```

3.9 CVF-9

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** MarginEngine.sol

Recommendation The argument type should be "IVAM".

Listing 9:

```
154 function setVAMM(address _vAMMAddress) external override
      ↪ onlyOwner {
```

3.10 CVF-10

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** MarginEngine.sol

Recommendation The argument type should be "IFCM".

Listing 10:

```
158 function setFCM(address _fcm) external override onlyOwner {
```

3.11 CVF-11

- **Severity** Minor
- **Status** Fixed
- **Category** Unclear behavior
- **Source** MarginEngine.sol

Description There are no range checks for the arguments.
Recommendation Consider adding appropriate checks.

Listing 11:

```
163 function setSecondsAgo(uint256 _secondsAgo)

174 function setCacheMaxAgeInSeconds(uint256 _cacheMaxAgeInSeconds)

203 function setLiquidatorReward(uint256 _liquidatorRewardWad)
      ↪ external override onlyOwner {
```

3.12 CVF-12

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** MarginEngine.sol

Recommendation This comment should be removed.

Listing 12:

```
214 /// Costin: update position to account for fees?
```

3.13 CVF-13

- **Severity** Major
- **Status** Fixed
- **Category** Unclear behavior
- **Source** MarginEngine.sol

Description This returns a stored position that could be out of date.
Recommendation Consider calculating and returning an up-to-date position without storing it.

Listing 13:

```
215 return positions.get(_owner, tickLower, tickUpper);
```

3.14 CVF-14

- **Severity** Minor
- **Status** Fixed
- **Category** Unclear behavior
- **Source** MarginEngine.sol

Description This will revert in case marginDelta = -2^255.
Recommendation Consider surrounding with an "unchecked" block.

Listing 14:

```
230 if (uint256(-_marginDelta) > marginEngineBalance) {
        uint256 remainingDeltaToCover = uint256(-_marginDelta);

238     underlyingToken.safeTransfer(_account, uint256(-_marginDelta
            ↪ ));
```

3.15 CVF-15

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** MarginEngine.sol

Description The expression "uint256(-_marginDelta)" is calculated several times.
Recommendation Consider calculating once and reusing.

Listing 15:

```
230 if (uint256(-_marginDelta) > marginEngineBalance) {
        uint256 remainingDeltaToCover = uint256(-_marginDelta);

238     underlyingToken.safeTransfer(_account, uint256(-_marginDelta
            ↪ ));
```

3.16 CVF-16

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** MarginEngine.sol

Recommendation When passing a boolean literal to a function, it is a good practice to put the corresponding argument name as a comment, to make the code easier to read.

Listing 16:

```
257 updatePositionTokenBalancesAndAccountForFees(position, tickLower
    ↪ , tickUpper, false);

301 updatePositionTokenBalancesAndAccountForFees(position, tickLower
    ↪ , tickUpper, false);

371 updatePositionTokenBalancesAndAccountForFees(position, tickLower
    ↪ , tickUpper, false);

404 updatePositionTokenBalancesAndAccountForFees(position, params.
    ↪ tickLower, params.tickUpper, true);

421 updatePositionTokenBalancesAndAccountForFees(position, tickLower
    ↪ , tickUpper, false);

430     getPositionMarginRequirement(position, tickLower, tickUpper,
        ↪  false)

488     getPositionMarginRequirement(position, tickLower, tickUpper,
        ↪  false)

711     true

774             true,

861             true

886             false
```

3.17 CVF-17

- **Severity** Minor
- **Status** Fixed
- **Category** Unclear behavior
- **Source** MarginEngine.sol

Description This check should be done only when "marginDelta" is negative.

Listing 17:

```
259 require((position.margin + marginDelta) >= 0, "can't withdraw
    ↪ more than have");
```

3.18 CVF-18

- **Severity** Moderate
- **Status** Fixed
- **Category** Unclear behavior
- **Source** MarginEngine.sol

Description This logic makes the contract behavior hard to predict and also open an attack vector. The owner may front-run a transaction and revoke approval for the "msg.sender", thus token will be transferred from "msg.sender" rather than from "owner".
Recommendation Consider always transferring token from "owner".
Client Comment Always transferring from the msg.sender (so that other accounts can also choose to top up the balance of the position, e.g. the Voltz Insurance module).

Listing 18:

```
278 if (factory.isApproved(_owner, msg.sender)) {
        depositor = _owner;
280 } else {
        depositor = msg.sender;
    }
```

3.19 CVF-19

- **Severity** Minor
- **Status** Info
- **Category** Readability
- **Source** MarginEngine.sol

Description This line is way too long.
Recommendation Consider reformatting and/or refactoring.
Client Comment Would prefer to keep the way it is.

Listing 19:

```
303 int256 settlementCashflow = FixedAndVariableMath.
    ↪ calculateSettlementCashflow(position.fixedTokenBalance,
    ↪ position.variableTokenBalance, termStartTimestampWad,
    ↪ termEndTimestampWad, rateOracle.variableFactor(
    ↪ termStartTimestampWad, termEndTimestampWad));
```

3.20 CVF-20

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** MarginEngine.sol

Description The "position.isSettled" is always true here.
Recommendation Consider replacing with the "true" literal.

Listing 20:

```
309 emit SettlePosition(_owner, tickLower, tickUpper, position.
    ↪ fixedTokenBalance, position.variableTokenBalance, position
    ↪ .margin, position.isSettled);
```

3.21 CVF-21

- **Severity** Minor
- **Status** Info
- **Category** Unclear behavior
- **Source** MarginEngine.sol

Description Underflow and thus revert is possible here.
Recommendation Consider refactoring like this: if (block.timestamp - cachedHistoricalApyRefreshTimestamp > cacheMaxAgeInSeconds) {

Listing 21:

```
319 if (cachedHistoricalApyRefreshTimestamp < block.timestamp -
    ↪ cacheMaxAgeInSeconds) {

333 if (cachedHistoricalApyRefreshTimestamp < block.timestamp -
    ↪ cacheMaxAgeInSeconds) {
```

3.22 CVF-22

- **Severity** Minor
- **Status** Fixed
- **Category** Readability
- **Source** MarginEngine.sol

Recommendation Should be "else return ..." for readability.
Client Comment Would prefer to keep the way it is.

Listing 22:

```
337 return cachedHistoricalApy;
```

3.23 CVF-23

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** MarginEngine.sol

Recommendation These variables are redundant, as obtaining the current block timestamp is cheaper than accessing a local variable.

Listing 23:

```
347 uint256 to = block.timestamp;
    uint256 from = to - secondsAgo;
```

3.24 CVF-24

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

Description Overflow is possible when converting to "int256".
Recommendation Consider using safe conversion.

Listing 24:

```
383 position.updateMarginViaDelta(-int256(liquidatorRewardValue));

424     position.updateMarginViaDelta(-int256(cumulativeFeeIncurred)
          ↪ );

429 int256 positionMarginRequirement = int256(

464     position.updateMarginViaDelta(int256(feeDelta));

487 int256 positionMarginRequirement = int256(

594         position.updateMarginViaDelta(-int256(
              ↪ _cumulativeFeeIncurred));

713 if (position.margin < int256(marginRequirement)) {

772     int256(

784     int256(

851     fixedTokenDeltaUnbalanced = int256(

876     fixedTokenDeltaUnbalanced = -int256(
```

3.25 CVF-25

- **Severity** Minor
- **Status** Info
- **Category** Unclear behavior
- **Source** MarginEngine.sol

Description This code reverts even if the account was already in margin call, and swap actually improved the situation.
Recommendation Consider not reverting in such a case.

Listing 25:

```
433 if (positionMarginRequirement > position.margin) {
        revert MarginRequirementNotMet();
    }
```

3.26 CVF-26

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** MarginEngine.sol

Recommendation This code is the same for both branches of the conditional statement, and should be placed before the conditional statement.
Client Comment Not sure it is possible since even though both branches of the conditional statement have the code, one of them has an extra check for isMint.

Listing 26:

```
457 (int256 fixedTokenGrowthInsideX128, int256
    ↪ variableTokenGrowthInsideX128, uint256 feeGrowthInsideX128
    ↪ ) = vamm.computeGrowthInside(tickLower, tickUpper);

468     (int256 fixedTokenGrowthInsideX128, int256
        ↪ variableTokenGrowthInsideX128, uint256
        ↪ feeGrowthInsideX128) = vamm.computeGrowthInside(
        ↪ tickLower, tickUpper);
```

3.27 CVF-27

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** MarginEngine.sol

Recommendation This code is the same for both branches of the conditional statement, and should be placed after the conditional statement.
Client Comment Not sure it is possible since even though both branches of the conditional statement have the code, one of them has an extra check for isMint

Listing 27:

```
462 position.updateFixedAndVariableTokenGrowthInside(
    ↪ fixedTokenGrowthInsideX128, variableTokenGrowthInsideX128)
    ↪ ;

465 position.updateFeeGrowthInside(feeGrowthInsideX128);

469     position.updateFixedAndVariableTokenGrowthInside(
        ↪ fixedTokenGrowthInsideX128,
        ↪ variableTokenGrowthInsideX128);
470     position.updateFeeGrowthInside(feeGrowthInsideX128);
```

3.28 CVF-28

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** MarginEngine.sol

Description These code blocks differ only in the value of the "sqrtPriceLimitX96" field.
Recommendation Consider merging them together, and using the ternary operator to populate the differing field.

Listing 28:

```
563 IVAMM.SwapParams memory params = IVAMM.SwapParams({
        recipient: _owner,
        amountSpecified: position.variableTokenBalance,
        sqrtPriceLimitX96: TickMath.MIN_SQRT_RATIO + 1,
        isExternal: true,
        tickLower: tickLower,
        tickUpper: tickUpper
570 });

580 IVAMM.SwapParams memory params = IVAMM.SwapParams({
        recipient: _owner,
        amountSpecified: position.variableTokenBalance,
        sqrtPriceLimitX96: TickMath.MAX_SQRT_RATIO - 1,
        isExternal: true,
        tickLower: tickLower,
        tickUpper: tickUpper
    });
```

3.29 CVF-29

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** MarginEngine.sol

Recommendation This code is the same for both branches of the conditional statement, and should be placed after the conditional statement.

Listing 29:

```
572 (_fixedTokenDelta, _variableTokenDelta, _cumulativeFeeIncurred)
        ↪ = vamm.swap(params);

589 (_fixedTokenDelta, _variableTokenDelta, _cumulativeFeeIncurred)
        ↪ = vamm.swap(params);
```

3.30 CVF-30

- **Severity** Minor
- **Status** Fixed
- **Category** Bad naming
- **Source** MarginEngine.sol

Description The names "from" and "to" are too generic.
Recommendation Consider renaming to "fromTick" and "toTick".

Listing 30:

```
604 function getExtraBalances(int24 from, int24 to, uint128
    ↪ liquidity, uint256 variableFactorWad) internal view
    ↪ returns (int256 extraFixedTokenBalance, int256
    ↪ extraVariableTokenBalance) {
```

3.31 CVF-31

- **Severity** Major
- **Status** Fixed
- **Category** Suboptimal
- **Source** MarginEngine.sol

Description The expressions "TickMath.getSqrtRatioAtTick(from)" and "TickMath.getSqrtRatioAtTick(to)" are calculated twice.
Recommendation Consider calculating once and reusing.

Listing 31:

```
608 TickMath.getSqrtRatioAtTick(from),
    TickMath.getSqrtRatioAtTick(to),

614 TickMath.getSqrtRatioAtTick(from),
    TickMath.getSqrtRatioAtTick(to),
```

3.32 CVF-32

- **Severity** Major
- **Status** Fixed
- **Category** Unclear behavior
- **Source** MarginEngine.sol

Recommendation Scenario 1 should be considered only when "tick" < "tickUpper".

Listing 32:

```
655 // scenario 1: a trader comes in and trades all the liquidity
    ↪ all the way to the the upper tick
```

3.33 CVF-33

- **Severity** Major
- **Status** Fixed
- **Category** Unclear behavior
- **Source** MarginEngine.sol

Recommendation Scenario 2 should be considered only when "tick" > "tickLower".

Listing 33:

```
656 // scenario 2: a trader comes in and trades all the liquidity
    ↪ all the way to the the lower tick
```

3.34 CVF-34

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** MarginEngine.sol

Description This expression is potentially calculated twice.
Recommendation Consider refactoring the code to calculate it at most once.

Listing 34:

```
676 ? ((sqrtPriceX96 > priceAtUpperTick) ? sqrtPriceX96 :
    ↪ priceAtUpperTick)

680 ? ((sqrtPriceX96 > priceAtUpperTick) ? sqrtPriceX96 :
    ↪ priceAtUpperTick)
```

3.35 CVF-35

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** MarginEngine.sol

Description This expression is potentially calculated twice.
Recommendation Consider refactoring the code to calculate it at most once.

Listing 35:

```
677 : ((sqrtPriceX96 < priceAtLowerTick) ? sqrtPriceX96 :
    ↪ priceAtLowerTick);

681 : ((sqrtPriceX96 < priceAtLowerTick) ? sqrtPriceX96 :
    ↪ priceAtLowerTick);
```

3.36 CVF-36

- **Severity** Minor
- **Status** Info
- **Category** Readability
- **Source** MarginEngine.sol

Recommendation Consider using "min" and "max" functions to make this code more readable.
Client Comment Would prefer to keep the current implementation.

Listing 36:

```
676     ? ((sqrtPriceX96 > priceAtUpperTick) ? sqrtPriceX96 :
          ↪ priceAtUpperTick)
        : ((sqrtPriceX96 < priceAtLowerTick) ? sqrtPriceX96 :
          ↪ priceAtLowerTick);

680     ? ((sqrtPriceX96 > priceAtUpperTick) ? sqrtPriceX96 :
          ↪ priceAtUpperTick)
        : ((sqrtPriceX96 < priceAtLowerTick) ? sqrtPriceX96 :
          ↪ priceAtLowerTick);

687 if (scenario1MarginRequirement > scenario2MarginRequirement) {
        return scenario1MarginRequirement;
    } else {
690     return scenario2MarginRequirement;
    }
```

3.37 CVF-37

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** MarginEngine.sol

Recommendation This code could be simplified as: return position.margin < int256(marginRequirement);

Listing 37:

```
713 if (position.margin < int256(marginRequirement)) {
        _isLiquidatable = true;
    } else {
        _isLiquidatable = false;
    }
```

3.38 CVF-38

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** MarginEngine.sol

Description The expression calculated here is always zero.
Recommendation Consider just using zero literal instead.

Listing 38:

```
901 int256 updatedVariableTokenBalance = variableTokenBalance +
        variableTokenDelta; // should be zero
```

3.39 CVF-39

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** MarginEngine.sol

Description This argument is always zero.
Recommendation Consider using zero literal instead.

Listing 39:

```
908 updatedVariableTokenBalance,
```

3.40 CVF-40

- **Severity** Minor
- **Status** Fixed
- **Category** Readability
- **Source** Factory.sol

Description Double slashes in paths look odd.
Recommendation Use single slashes.

Listing 40:

```
6 import "./interfaces//IMarginEngine.sol";
  import "./interfaces//IVAMM.sol";
  import "./interfaces//IFCM.sol";
```

3.41 CVF-41

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** Factory.sol

Recommendation The type of this variable should be "IMarginEngine".

Listing 41:

```
20 address public override masterMarginEngine;
```

3.42 CVF-42

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** Factory.sol

Recommendation The type of this variable should be "IVAMM".

Listing 42:

```
21 address public override masterVAMM;
```

3.43 CVF-43

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** Factory.sol

Recommendation The value type for this mapping should be "IFCM".

Listing 43:

```
22 mapping(uint8 => address) public override masterFCMs;
```

3.44 CVF-44

- **Severity** Minor
- **Status** Fixed
- **Category** Documentation
- **Source** Factory.sol

Description The semantics of the keys in this mapping is unclear.
Recommendation Consider documenting.

Listing 44:

```
23 mapping(address => mapping(address => bool)) private _approvals;
```

3.45 CVF-45

- **Severity** Minor
- **Status** Fixed
- **Category** Documentation
- **Source** Factory.sol

Description The semantics of the keys in this mapping is unclear.
Recommendation Consider documenting.

Listing 45:

```
22 mapping(uint8 => address) public override masterFCMs;
```

3.46 CVF-46

- **Severity** Minor
- **Status** Fixed
- **Category** Unclear behavior
- **Source** Factory.sol

Description This function should emit some event.

Listing 46:

```
25 function setApproval(address intAddress, bool allowIntegration)
     ↪ external override {
```

3.47 CVF-47

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** Factory.sol

Recommendation These variables should be declared immutable.
Client Comment We have introduced upgradability to the factory, meaning these values are no longer immutable

Listing 47:

```
20 address public override masterMarginEngine;
   address public override masterVAMM;
```

3.48 CVF-48

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** Factory.sol

Recommendation This function wouldn't be necessary if the "_approvals" mapping would be declared public, and would be named appropriately.

Listing 48:

```
29 function isApproved(address _owner, address intAddress) external
   ↪ override view returns (bool) {
```

3.49 CVF-49

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** Factory.sol

Recommendation The types of the arguments should be "IMarginEngine" and "IVAMM".

Listing 49:

```
33 constructor(address _masterMarginEngine, address _masterVAMM) {
```

3.50 CVF-50

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** Factory.sol

Recommendation The types of the arguments should be "IFCM" and "IRateOracle".

Listing 50:

```
38 function setMasterFCM(address masterFCMAddress, address
   ↪ _rateOracle) external override onlyOwner {
```

3.51 CVF-51

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** Factory.sol

Description The rate oracle is used only to obtain a protocol ID from it.
Recommendation Consider passing a protocol ID instead of a rate oracle.

Listing 51:

```
38 function setMasterFCM(address masterFCMAddress, address
   ↪ _rateOracle) external override onlyOwner {
```

3.52 CVF-52

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** Factory.sol

Description These checks are redundant. It is anyway possible to pass a dead address as an argument.
Recommendation We'd prefer to keep these checks for the reasons outlined in a related CVF above

Listing 52:

```
40 require(_rateOracle != address(0), "rate oracle must exist");
   require(masterFCMAddress != address(0), "master fcm must exist")
   ↪ ;
```

3.53 CVF-53

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** Factory.sol

Description The expression "masterFCMs[yieldBearingProtocolID]" is calculated twice.
Recommendation Consider calculating once and reusing.

Listing 53:

```
44 address masterFCMAddressOld = masterFCMs[yieldBearingProtocolID
   ↪ ];
masterFCMs[yieldBearingProtocolID] = masterFCMAddress;

67 require(masterFCMs[yieldBearingProtocolID] != address(0), "
   ↪ master FCM must be set");

69 return masterFCMs[yieldBearingProtocolID].
   ↪ predictDeterministicAddress(salt);

87 if (masterFCMs[yieldBearingProtocolID] != address(0)) {
     address masterFCM = masterFCMs[yieldBearingProtocolID];
```

3.54 CVF-54

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** Factory.sol

Recommendation The type of the "_underlyingToken" argument should be "IERC20". The type of the "_rateOracle" argument should be "IRateOracle".

Listing 54:

```
49 function getSalt(address _underlyingToken, address _rateOracle,
   ↪ uint256 _termStartTimestampWad, uint256
   ↪ _termEndTimestampWad, int24 _tickSpacing) internal pure
   ↪ returns (bytes32 salt) {

53 function getVAMMAddress(address _underlyingToken, address
   ↪ _rateOracle, uint256 _termStartTimestampWad, uint256
   ↪ _termEndTimestampWad, int24 _tickSpacing) external view
   ↪ override returns (address) {

59 function getMarginEngineAddress(address _underlyingToken,
   ↪ address _rateOracle, uint256 _termStartTimestampWad,
   ↪ uint256 _termEndTimestampWad, int24 _tickSpacing) external
   ↪  view override returns (address) {

65 function getFCMAddress(address _underlyingToken, address
   ↪ _rateOracle, uint256 _termStartTimestampWad, uint256
   ↪ _termEndTimestampWad, int24 _tickSpacing) external view
   ↪ override returns (address) {

72 function deployIrsInstance(address _underlyingToken, address
   ↪ _rateOracle, uint256 _termStartTimestampWad, uint256
   ↪ _termEndTimestampWad, int24 _tickSpacing) external
   ↪ override onlyOwner returns (address marginEngineProxy,
   ↪ address vammProxy, address fcmProxy) {
```

3.55 CVF-55

- **Severity** Minor
- **Status** Info
- **Category** Unclear behavior
- **Source** Factory.sol

Description There are no range checks for some of the arguments.
Recommendation Consider adding appropriate checks.
Client Comment geteSalt, getVAMMAddress and getFCMAddress no longer exist as functions in the Factory. We already have a range check for tick spacing in the VAMM, is it also worth adding a range check in the deployIRSInstance of the factory?

Listing 55:

```
49 function getSalt(address _underlyingToken, address _rateOracle,
   ↪ uint256 _termStartTimestampWad, uint256
   ↪ _termEndTimestampWad, int24 _tickSpacing) internal pure
   ↪ returns (bytes32 salt) {

53 function getVAMMAddress(address _underlyingToken, address
   ↪ _rateOracle, uint256 _termStartTimestampWad, uint256
   ↪ _termEndTimestampWad, int24 _tickSpacing) external view
   ↪ override returns (address) {

59 function getMarginEngineAddress(address _underlyingToken,
   ↪ address _rateOracle, uint256 _termStartTimestampWad,
   ↪ uint256 _termEndTimestampWad, int24 _tickSpacing) external
   ↪  view override returns (address) {

65 function getFCMAddress(address _underlyingToken, address
   ↪ _rateOracle, uint256 _termStartTimestampWad, uint256
   ↪ _termEndTimestampWad, int24 _tickSpacing) external view
   ↪ override returns (address) {

72 function deployIrsInstance(address _underlyingToken, address
   ↪ _rateOracle, uint256 _termStartTimestampWad, uint256
   ↪ _termEndTimestampWad, int24 _tickSpacing) external
   ↪ override onlyOwner returns (address marginEngineProxy,
   ↪ address vammProxy, address fcmProxy) {
```

3.56 CVF-56

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** Factory.sol

Recommendation These checks are redundant, as it is anyway possible to set a dead VAMM or MarginEngine address.
Client Comment Would prefer to keep the checks.

Listing 56:

```
54 require(masterVAMM != address(0), "master VAMM must be set");

60 require(masterMarginEngine != address(0), "master MarginEngine
   ↪ must be set");
```

3.57 CVF-57

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** Factory.sol

Recommendation The return types should be “IMarginEngine”, “IVAMM”, and “IFCM”.

Listing 57:

```
72 function deployIrsInstance(address _underlyingToken, address
   ↪ _rateOracle, uint256 _termStartTimestampWad, uint256
   ↪ _termEndTimestampWad, int24 _tickSpacing) external
   ↪ override onlyOwner returns (address marginEngineProxy,
   ↪ address vammProxy, address fcmProxy) {
```

3.58 CVF-58

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** Factory.sol

Recommendation The maximum tick spacing should be a named constant

Listing 58:

```
76 require(_tickSpacing > 0 && _tickSpacing < 16384);
```

3.59 CVF-59

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** Factory.sol

Recommendation These variables are redundant. Just use the named return values instead, i.e. "marginEngineProxy", "vammProxy", and "fcmProxy".

Listing 59:

```
78 IMarginEngine marginEngine = IMarginEngine(masterMarginEngine.
       ↪ cloneDeterministic(salt));
   IVAMM vamm = IVAMM(masterVAMM.cloneDeterministic(salt));

85 IFCM fcm;
```

3.60 CVF-60

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** VAMM.sol

Description We didn't review these libraries.

Listing 60:

```
13 import "prb-math/contracts/PRBMathUD60x18.sol";
   import "prb-math/contracts/PRBMathSD59x18.sol";
```

3.61 CVF-61

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** VAMM.sol

Recommendation This functionality should be moved to a utility base contract similar to "ReentrancyGuard" from OpenZeppelin. Alternatively, consider using "ReentrancyGuard" from open zeppelin instead of a custom implementation.
Client Comment Would prefer to keep the current implementation at this stage.

Listing 61:

```
39 bool public override unlocked;

55 modifier lock() {
     require(unlocked, "LOK");
     unlocked = false;
     _;
     unlocked = true;
60 }
```

3.62 CVF-62

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** VAMM.sol

Description This variable is never read.
Recommendation Consider removing it.

Listing 62:

```
41 address private deployer;
```

3.63 CVF-63

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** VAMM.sol

Recommendation The value assigned here is never used, consider removing the assignment.

Listing 63:

```
75 deployer = msg.sender;
```

3.64 CVF-64

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** VAMM.sol

Recommendation The type of the "_marginEngineAddress" should be "IMarginEngine".

Listing 64:

```
79 function initialize(address _marginEngineAddress, int24
   ↪ _tickSpacing) external override initializer {
```

3.65 CVF-65

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** VAMM.sol

Recommendation This check is redundant. It is anyway possible to pass a dead margin engine address.
Client Comment Would prefer to keep this check.

Listing 65:

```
80 require(_marginEngineAddress != address(0), "ME must be set");
```

3.66 CVF-66

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** VAMM.sol

Description There is no range check for the "_tickSpacing" argument.
Recommendation Consider adding an appropriate check.

Listing 66:

```
79 function initialize(address _marginEngineAddress, int24
   ↪ _tickSpacing) external override initializer {
```

3.67 CVF-67

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** VAMM.sol

Recommendation This line could be simplified using the "-=" operator.

Listing 67:

```
130 protocolFees = protocolFees - protocolFeesCollected;
```

3.68 CVF-68

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** VAMM.sol

Description There is no range check for the "sqrtPriceX96" argument.
Recommendation Consider adding an appropriate check.

Listing 68:

```
134 function initializeVAMM(uint160 sqrtPriceX96) external override
    ↪ {
```

3.69 CVF-69

- **Severity** Minor
- **Status** Fixed
- **Category** Unclear behavior
- **Source** VAMM.sol

Description It is not explicitly checked that the "sqrtPriceX96" value is not zero, thus it is not obvious that this condition efficiently prevents double initialization.
Recommendation Consider adding appropriate check or using another condition to prevent double initialization.

Listing 69:

```
135 if (vammVars.sqrtPriceX96 != 0) {
```

3.70 CVF-70

- **Severity** Moderate
- **Status** Info
- **Category** Flaw
- **Source** VAMM.sol

Description This function could be called by anyone and there is no economical incentives to provide a fair price here.
Recommendation Consider requiring the caller to provide certain amount of liquidity along with the call, which would motivate the caller to set the price close to the fair price.
Client Comment If they initialize at a bad price and add no liquidity it's relatively cheap to fix the price, it's a one time issue, only at pool creation. if they add a little bit of liquidity, e.g. 1 full range liquidity, you waste some money on fees but it is still a negligible amount. From Moody (uniswap): to date nobody has intentionally griefed afaik, griefing is inconsequential and costs the griefer money

Listing 70:

```
134 function initializeVAMM(uint160 sqrtPriceX96) external override
    ↪ {
```

3.71 CVF-71

- **Severity** Major
- **Status** Fixed
- **Category** Flaw
- **Source** VAMM.sol

Description It is not guaranteed that the "initialize" function was already executed, so it is possible to unlock a not fully initialized instance.
Recommendation Consider adding an appropriate check.

Listing 71:

```
143 unlocked = true;
```

3.72 CVF-72

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** VAMM.sol

Description There is no range check for the argument.
Recommendation Consider adding an appropriate check.

Listing 72:

```
148 function setFeeProtocol(uint8 feeProtocol) external override
    ↪ onlyOwner lock {

154 function setFee(uint256 _feeWad) external override onlyOwner
    ↪ lock {
```

3.73 CVF-73

- **Severity** Minor
- **Status** Fixed
- **Category** Unclear behavior
- **Source** VAMM.sol

Description These events are emitted even if nothing is actually changed

Listing 73:

```
151 emit SetFeeProtocol(feeProtocolOld, feeProtocol);

157 emit FeeSet(feeWadOld, feeWad);
```

3.74 CVF-74

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** VAMM.sol

Description There is no check to ensure that params.tickLower < params.tickUpper.
Recommendation Consider adding such check.

Listing 74:

```
187 function flipTicks(ModifyPositionParams memory params)
```

3.75 CVF-75

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** VAMM.sol

Recommendation When a boolean literal is passed as anargument, it is a good practice to put the agrument name in comment, to improve code readability.

Listing 75:

```
198 false,

209 true,
```

3.76 CVF-76

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** VAMM.sol

Description This logic is based on the knowledge about how tick update works internally.
Recommendation Consider moving this logic into the "Tick.update" function.
Client Comment Prefer to keep the current implementaiton.

Listing 76:

```
242 if (params.liquidityDelta < 0) {
      if (flippedLower) {
        ticks.clear(params.tickLower);
      }
      if (flippedUpper) {
        ticks.clear(params.tickUpper);
      }
    }
```

3.77 CVF-77

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** VAMM.sol

Recommendation This line could be simplified as: bool isFT = params.amountSpecified > 0;

Listing 77:

```
309 bool isFT = (params.amountSpecified > 0) ? true : false;
```

3.78 CVF-78

- **Severity** Minor
- **Status** Info
- **Category** Readability
- **Source** VAMM.sol

Description These three lines implicitly implement functionality very similar to the "lock" modifier.
Recommendation Consider using the modifier instead to make the code more readable and less error-prone.
Client Comment Prefer to keep the current implementation.

Listing 78:

```
311 checksBeforeSwap(params, vammVarsStart, isFT);

322 unlocked = false;

529 unlocked = true;
```

3.79 CVF-79

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** VAMM.sol

Recommendation This check should be performed at the very beginning of the function.

Listing 79:

```
318 Tick.checkTicks(params.tickLower, params.tickUpper);
```

3.80 CVF-80

- **Severity** Minor
- **Status** Info
- **Category** Flaw
- **Source** VAMM.sol

Description This reentrancy logic requires to set the variable in two distant parts of the code. This is error prone.
Recommendation Consider using modifiers

Listing 80:

```
322 unlocked = false;

529 unlocked = true;
```

3.81 CVF-81

- **Severity** Major
- **Status** Fixed
- **Category** Suboptimal
- **Source** VAMM.sol

Description On every iteration of this loop there are several places where different code is executed depending on the trade side.
Recommendation It would be more efficient to have two separate loop implementations and choose what implementation to run based on the trade side.

Listing 81:

```
359 while (
360   state.amountSpecifiedRemaining != 0 &&
      state.sqrtPriceX96 != params.sqrtPriceLimitX96
    ) {
```

3.82 CVF-82

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** VAMM.sol

Description The condition is already calculated and available as the "isFT" variable.

Listing 82:

```
406 if (params.amountSpecified > 0) {
```

3.83 CVF-83

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** VAMM.sol

Recommendation This code is the same for both branches of the conditional statement and should be placed before the conditional statement.

Listing 83:

```
494 vammVars.sqrtPriceX96 = state.sqrtPriceX96;

498 vammVars.sqrtPriceX96 = state.sqrtPriceX96;
```

3.84 CVF-84

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** VAMM.sol

Description There are no range checks for the ticks.
Recommendation Consider adding appropriate checks.

Listing 84:

```
534 int24 tickLower,
    int24 tickUpper
```

3.85 CVF-85

- **Severity** Minor
- **Status** Info
- **Category** Unclear behavior
- **Source** VAMM.sol

Recommendation Should be "<=".
Client Comment Inclined to keep the implementation aligned with that of Uni v3.

Listing 85:

```
594 params.sqrtPriceLimitX96 < TickMath.MAX_SQRT_RATIO
```

3.86 CVF-86

- **Severity** Minor
- **Status** Info
- **Category** Unclear behavior
- **Source** VAMM.sol

Recommendation Should be ">=".
Client Comment Inclined to keep the implementation aligned with that of Uni v3.

Listing 86:

```
596 params.sqrtPriceLimitX96 > TickMath.MIN_SQRT_RATIO,
```

3.87 CVF-87

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

Description Overflow is possible when converting to "int256".
Recommendation Consider using safe conversion.

Listing 87:

```
628 stateVariableTokenGrowthGlobalX128 = state.
    ↪ variableTokenGrowthGlobalX128 + int256(FullMath.mulDiv(
    ↪ uint256(step.variableTokenDelta), FixedPoint128.Q128,
    ↪ state.liquidity));

643 stateFixedTokenGrowthGlobalX128 = state.
    ↪ fixedTokenGrowthGlobalX128 - int256(FullMath.mulDiv(
    ↪ uint256(-fixedTokenDelta), FixedPoint128.Q128, state.
    ↪ liquidity));

651 stateVariableTokenGrowthGlobalX128= state.
    ↪ variableTokenGrowthGlobalX128 - int256(FullMath.mulDiv(
    ↪ uint256(-step.variableTokenDelta), FixedPoint128.Q128,
    ↪ state.liquidity));

666 stateFixedTokenGrowthGlobalX128 = state.
    ↪ fixedTokenGrowthGlobalX128 + int256(FullMath.mulDiv(
    ↪ uint256(fixedTokenDelta), FixedPoint128.Q128, state.
    ↪ liquidity));
```

3.88 CVF-88

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** VAMM.sol

Recommendation This code is the same for both branches of the conditional statement, and should be placed before the conditional statement.

Listing 88:

```
635 fixedTokenDelta = FixedAndVariableMath.getFixedTokenBalance(
      step.fixedTokenDeltaUnbalanced,
      step.variableTokenDelta,
      variableFactorWad,
      termStartTimestampWad,
640   termEndTimestampWad
    );

658 fixedTokenDelta = FixedAndVariableMath.getFixedTokenBalance(
      step.fixedTokenDeltaUnbalanced,
660   step.variableTokenDelta,
      variableFactorWad,
      termStartTimestampWad,
      termEndTimestampWad
    );
```

3.89 CVF-89

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** IMarginEngine.sol

Description This structure is not used in any of the interface functions.
Recommendation Consider moving its definition to the implementation.

Listing 89:

```
14 struct PositionMarginRequirementLocalVars2 {
```

3.90 CVF-90

- **Severity** Minor
- **Status** Fixed
- **Category** Bad naming
- **Source** IMarginEngine.sol

Recommendation Events are usually named via nouns, such as "HistoricalApyWindow", "CacheMaxAge" etc.

Listing 90:

```
64 event HistoricalApyWindowSet(uint256 secondsAgoOld, uint256
   ↪ secondsAgo);
   event CacheMaxAgeSet(

70 event CollectProtocol(address sender, address recipient, uint256
   ↪  amount);
   event LiquidatorRewardSet(

76 event VAMMSet(IVAMM vammOld, IVAMM vamm);

78 event FCMSet(IFCM fcmOld, IFCM fcm);

80 event MarginCalculatorParametersSet(

85 event UpdatePositionMargin(

92 event SettlePosition(

102 event LiquidatePosition(

112 event UpdatePositionPostSwap(

121 event UpdatePositionPostMintBurn(
```

3.91 CVF-91

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** IMarginEngine.sol

Description The old parameters are redundant, as their values could be derived from the previous events,.

Listing 91:

```
64 event HistoricalApyWindowSet(uint256 secondsAgoOld, uint256
   ↪ secondsAgo);

66     uint256 cacheMaxAgeInSecondsOld,

72     uint256 liquidatorRewardWadOld,

76 event VAMMSet(IVAMM vammOld, IVAMM vamm);

78 event FCMSet(IFCM fcmOld, IFCM fcm);

81     MarginCalculatorParameters marginCalculatorParametersOld,
```

3.92 CVF-92

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** IMarginEngine.sol

Recommendation The "recipient" parameter should be indexed.

Listing 92:

```
70 event CollectProtocol(address sender, address recipient, uint256
   ↪  amount);
```

3.93 CVF-93

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** IMarginEngine.sol

Recommendation The "vamm" parameter should be indexed.

Listing 93:

```
76 event VAMMSet(IVAMM vammOld, IVAMM vamm);
```

3.94 CVF-94

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** IMarginEngine.sol

Recommendation The "fcm" parameter should be indexed.

Listing 94:

```
78 event FCMSet(IFCM fcmOld, IFCM fcm);
```

3.95 CVF-95

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** IMarginEngine.sol

Recommendation These parameters should be indexed.

Listing 95:

```
86  address owner,
    int24 tickLower,
    int24 tickUpper,

93  address owner,
    int24 tickLower,
    int24 tickUpper,

103 address owner,
    int24 tickLower,
    int24 tickUpper,

113 address owner,
    int24 tickLower,
    int24 tickUpper,

122 address owner,
    int24 tickLower,
    int24 tickUpper,
```

3.96 CVF-96

- **Severity** Minor
- **Status** Fixed
- **Category** Unclear behavior
- **Source** IMarginEngine.sol

Description This event should include the settlement cash flow value.

Listing 96:

```
92 event SettlePosition(
```

3.97 CVF-97

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** IMarginEngine.sol

Description This parameter is always true.
Recommendation Consider removing it.

Listing 97:

```
99 bool isSettled
```

3.98 CVF-98

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** IMarginEngine.sol

Recommendation The type of this argument should be "IERC20".

Listing 98:

```
173 address _underlyingToken,
```

3.99 CVF-99

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** IMarginEngine.sol

Recommendation The type of this argument should be "IRateOracle".

Listing 99:

```
174 address _rateOracleAddress,
```

3.100 CVF-100

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** IMarginEngine.sol

Description Names of some function arguments have the underscore ('_') prefix, while names of other arguments don't have prefix.
Recommendation Consider using a consistent naming policy.

Listing 100:

```
173 address _underlyingToken,
    address _rateOracleAddress,
    uint256 _termStartTimestampWad,
    uint256 _termEndTimestampWad

198 address _owner,
    int24 tickLower,
200 int24 tickUpper
```

3.101 CVF-101

- **Severity** Minor
- **Status** Fixed
- **Category** Bad naming
- **Source** IMarginEngine.sol

Description The name is too generic.
Recommendation Consider making it more specific.

Listing 101:

```
208 function secondsAgo() external view returns (uint256);
```

3.102 CVF-102

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** IMarginEngine.sol

Description A position is identified by three arguments: "_owner", "tickLower", and "tickUpper", however the order of these arguments is different in different for different functions.
Recommendation Consider using a consistent order of similar arguments.

Listing 102:

```
236 address _owner,
    int24 tickLower,
    int24 tickUpper,

253 int24 tickLower,
    int24 tickUpper,
    address _owner

264 int24 tickLower,
    int24 tickUpper,
    address _owner

288 address _owner,
    int24 tickLower,
290 int24 tickUpper,
```

3.103 CVF-103

- **Severity** Minor
- **Status** Info
- **Category** Unclear behavior
- **Source** IMarginEngine.sol

Description This function should return the balanced delta.
Client Comment Acknowledged.

Listing 103:

```
252 function settlePosition(
```

3.104 CVF-104

- **Severity** Minor
- **Status** Fixed
- **Category** Unclear behavior
- **Source** IMarginEngine.sol

Description This function should return the token amounts transferred to the liquidator.

Listing 104:

```
263 function liquidatePosition(
```

3.105 CVF-105

- **Severity** Minor
- **Status** Info
- **Category** Unclear behavior
- **Source** IMarginEngine.sol

Description This function should return the token balances deltas.
Client Comment Acknowledged.

Listing 105:

```
276 function updatePositionPostVAMMInducedMintBurn(
```

3.106 CVF-106

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** AaveFCM.sol

Description We didn't review this file.
Client Comment Acknowledged.

Listing 106:

```
10 import "prb-math/contracts/PRBMathUD60x18.sol";
```

3.107 CVF-107

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** AaveFCM.sol

Description This variable should be declared as immutable.

Listing 107:

```
36 address private deployer;
```

3.108 CVF-108

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** AaveFCM.sol

Description This variable is never read.
Recommendation Consider removing it.

Listing 108:

```
36 address private deployer;
```

3.109 CVF-109

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** AaveFCM.sol

Description The value assigned here is never read.
Recommendation Consider removing this assignment.

Listing 109:

```
56 deployer = msg.sender; /// this is presumably the factory
```

3.110 CVF-110

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** AaveFCM.sol

Recommendation The argument types should be "IVAMM" and "IMarginEngine" respectively.

Listing 110:

```
69 function initialize(address _vammAddress, address
   ↪ _marginEngineAddress) external override initializer {
```

3.111 CVF-111

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** AaveFCM.sol

Description The underlying token is obtained from the margin engine multiple times.
Recommendation Consider obtaining once during initialization and saving in a storage variable.

Listing 111:

```
75  address underlyingTokenAddress = address(marginEngine.
      ↪ underlyingToken());

121 uint256 currentRNI = aaveLendingPool.getReserveNormalizedIncome(
      ↪ address(marginEngine.underlyingToken()));

140 uint256 currentRNI = aaveLendingPool.getReserveNormalizedIncome(
      ↪ address(marginEngine.underlyingToken()));

284   uint256 currentRNI = aaveLendingPool.
        ↪ getReserveNormalizedIncome(address(marginEngine.
        ↪ underlyingToken()));
```

3.112 CVF-112

- **Severity** Minor
- **Status** Fixed
- **Category** Bad naming
- **Source** AaveFCM.sol

Recommendation Events are usually named via nouns, such as "FullyCollateralizedSwap".

Listing 112:

```
84 event InitiateFullyCollateralisedSwap(
```

3.113 CVF-113

- **Severity** Minor
- **Status** Fixed
- **Category** Readability
- **Source** AaveFCM.sol

Recommendation This event should have an indexed "trader" parameter.

Listing 113:

```
84 event InitiateFullyCollateralisedSwap(
```

3.114 CVF-114

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** AaveFCM.sol

Description There are no range checks for the "sqrtPriceLimitX96" argument.
Recommendation Consider adding appropriate checks.
Client Comment We feel this is not necessary since checksBeforeSwap already does the range check.

Listing 114:

```
94 function initiateFullyCollateralisedFixedTakerSwap(uint256
     ↪ notional, uint160 sqrtPriceLimitX96) external override {
```

3.115 CVF-115

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

Description Overflow is possible here.

Listing 115:

```
107 amountSpecified: int256(notional),
```

3.116 CVF-116

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** AaveFCM.sol

Description This tick range looks odd in case the current tick is outside it.
Recommendation Consider using a tick range surrounding the current tick.
Client Comment The Aave FCM is not supposed to be used for liquidity provisioning, hence the choice of the tickLower and tickUpper doesn't really matter. The reason we went with -tickSpacing and +tickSpacing is because these two values should always be a valid input in order to create a position for the FCM that is uniquely identifyable.

Listing 116:

```
110 tickLower: -tickSpacing,
    tickUpper: tickSpacing

169 tickLower: -tickSpacing,
170 tickUpper: tickSpacing
```

3.117 CVF-117

- **Severity** Minor
- **Status** Fixed
- **Category** Readability
- **Source** AaveFCM.sol

Description It is assumed that "variableTokenDelta" is non-positive here, while this fact is not explicitly checked.
Recommendation Consider adding an explicit "require" statement to check it.

Listing 117:

```
117 underlyingYieldBearingToken.safeTransferFrom(msg.sender, address
    ↪ (this), uint256(-variableTokenDelta));

123 uint256 updatedTraderMargin = trader.
    ↪ marginInScaledYieldBearingTokens + uint256(-
    ↪ variableTokenDelta).rayDiv(currentRNI);
```

3.118 CVF-118

- **Severity** Minor
- **Status** Fixed
- **Category** Readability
- **Source** AaveFCM.sol

Description This assumes that "trader.variableTokenBalance" is non-positive, while there is not explicit check for this.
Recommendation Consider adding an explicit "require" statement.

Listing 118:

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

3.119 CVF-119

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

Description Underflow is possible when converting to "uint256".

Listing 119:

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

3.120 CVF-120

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

Description Overflow is possible when converting to "int256".

Listing 120:

```
166 amountSpecified: -int256(notionalToUnwind),
```

3.121 CVF-121

- **Severity** Minor
- **Status** Fixed
- **Category** Unclear behavior
- **Source** AaveFCM.sol

Description It is assumed that "variableTokenDelta" is non-negative, while this is not explicitly checked.
Recommendation Consider checking via an explicit "require" statement.

Listing 121:

```
184 underlyingYieldBearingToken.safeTransfer(msg.sender, uint256(
    ↪ variableTokenDelta));

188 uint256 updatedTraderMargin = trader.
    ↪ marginInScaledYieldBearingTokens - uint256(
    ↪ variableTokenDelta).rayDiv(currentRNI);
```

3.122 CVF-122

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** AaveFCM.sol

Description This call reads back trader balances that were just written into the storage.
Recommendation Consider refactoring the code to avoid excess storage access.

Listing 122:

```
192 checkMarginRequirement(trader);
```

3.123 CVF-123

- **Severity** Minor
- **Status** Fixed
- **Category** Readability
- **Source** AaveFCM.sol

Recommendation Should be "once" instead of "one".

Listing 123:

```
205 /// @dev one future variable cashflows are covered, we need to
    ↪ check if the remaining settlement cashflow is covered by
    ↪ the remaining margin in yield bearing tokens
```

3.124 CVF-124

- **Severity** Minor
- **Status** Fixed
- **Category** Documentation
- **Source** AaveFCM.sol

Description This comment is confusing.
Recommendation Consider elaborate a bit more on how the variable cash flow is calculated.

Listing 124:

```
245 /// @dev variable cashflow based on the term from start to now
    ↪ since the cashflow from now to maturity is fully
    ↪ collateralised by the yield bearing tokens
```

3.125 CVF-125

- **Severity** Minor
- **Status** Info
- **Category** Readability
- **Source** AaveFCM.sol

Description This line is way too long.
Recommendation Consider reformatting and/or refactoring.
Client Comment Acknowledged.

Listing 125:

```
277 int256 settlementCashflow = FixedAndVariableMath.
    ↪ calculateSettlementCashflow(trader.fixedTokenBalance,
    ↪ trader.variableTokenBalance, marginEngine.
    ↪ termStartTimestampWad(), marginEngine.termEndTimestampWad
    ↪ (), rateOracle.variableFactor(marginEngine.
    ↪ termStartTimestampWad(), marginEngine.termEndTimestampWad
    ↪ ()));
```

3.126 CVF-126

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** MarginCalculator.sol

Description We didn't review these files.
Client Comment Acknowledged.

Listing 126:

```
5 import "prb-math/contracts/PRBMathUD60x18.sol";
  import "prb-math/contracts/PRBMathSD59x18.sol";
```

3.127 CVF-127

- **Severity** Minor
- **Status** Fixed
- **Category** Bad naming
- **Source** MarginCalculator.sol

Description The names of this constant is confusing. 1 WEI is 1e-18 of 1 ETH, not 1r18 or something.
Recommendation Consider renaming.

Listing 127:

```
44 int256 public constant ONE_WEI = 10**18;
```

3.128 CVF-128

- **Severity** Minor
- **Status** Fixed
- **Category** Readability
- **Source** MarginCalculator.sol

Recommendation This value could be rendered as "1e18".

Listing 128:

```
44 int256 public constant ONE_WEI = 10**18;
```

3.129 CVF-129

- **Severity** Minor
- **Status** Fixed
- **Category** Readability
- **Source** MarginCalculator.sol

Recommendation This value could be rendered as "31536000e18".

Listing 129:

```
47 int256 public constant SECONDS_IN_YEAR = 31536000 * ONE_WEI;
```

3.130 CVF-130

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** MarginCalculator.sol

Description This check seems redundant. The end timestamp is never used alone, but only the difference between the end timestamp and the current timestamp is used.
Recommendation Consider removing this check.

Listing 130:

```
56 require(termEndTimestampWad > 0, "termEndTimestamp must be > 0")
   ↪ ;
```

3.131 CVF-131

- **Severity** Minor
- **Status** Fixed
- **Category** Unclear behavior
- **Source** MarginCalculator.sol

Recommendation Should be >= by the code.

Listing 131:

```
59 "endTime must be > currentTime"
```

3.132 CVF-132

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** MarginCalculator.sol

Description The expression "_marginCalculatorParameters.beteWad" is calculated twice.
Recommendation Consider calculating once and reusing.

Listing 132:

```
61 require(_marginCalculatorParameters.betaWad != 0, "parameters
   ↪ not set");

63 int256 betaWad = _marginCalculatorParameters.betaWad;
```

3.133 CVF-133

- **Severity** Minor
- **Status** Fixed
- **Category** Readability
- **Source** MarginCalculator.sol

Recommendation These calls could be made simpler and more readable via a "using"statements.

Listing 133:

```
 66 int256 scaledTimeWad = PRBMathSD59x18.div(

 71 int256 expInputWad = PRBMathSD59x18.mul((-betaWad),
      ↪ scaledTimeWad);

 73 timeFactorWad = PRBMathSD59x18.exp(expInputWad);

 92 int256 beta4Wad = PRBMathSD59x18.mul(

 97 int256 alpha4Wad = PRBMathSD59x18.mul(

112 apyBoundVars.kWad = PRBMathSD59x18.div(

117 apyBoundVars.zetaWad = PRBMathSD59x18.div(
        PRBMathSD59x18.mul(

125 apyBoundVars.lambdaNumWad = PRBMathSD59x18.mul(
        PRBMathSD59x18.mul(beta4Wad, apyBoundVars.timeFactorWad),

130 apyBoundVars.lambdaDenWad = PRBMathSD59x18.mul(

135 apyBoundVars.lambdaWad = PRBMathSD59x18.div(

140 apyBoundVars.criticalValueMultiplierWad = PRBMathSD59x18.mul(
        (PRBMathSD59x18.mul(

149         apyBoundVars.criticalValueWad = PRBMathSD59x18.mul(

151             PRBMathSD59x18.sqrt(apyBoundVars.
                  ↪ criticalValueMultiplierWad)

154         apyBoundVars.criticalValueWad = PRBMathSD59x18.mul(

156             PRBMathSD59x18.sqrt(apyBoundVars.
                  ↪ criticalValueMultiplierWad)

161         ? PRBMathSD59x18.mul(

167         : PRBMathSD59x18.mul(
```

(204, 215, 231, 242, 303, 305, 312, 323, 328, 335, 337, 363, 371)

3.134 CVF-134

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** MarginCalculator.sol

Recommendation These multiplications could be replaced with left shifts.

Listing 134:

```
92  int256 beta4Wad = PRBMathSD59x18.mul(
        _marginCalculatorParameters.betaWad,
        PRBMathSD59x18.fromInt(4)
    );

97  int256 alpha4Wad = PRBMathSD59x18.mul(
        _marginCalculatorParameters.alphaWad,
        PRBMathSD59x18.fromInt(4)
100 );

140 apyBoundVars.criticalValueMultiplierWad = PRBMathSD59x18.mul(
        (PRBMathSD59x18.mul(
            PRBMathSD59x18.fromInt(2),
            apyBoundVars.lambdaWad
        ) + apyBoundVars.kWad),
        (PRBMathSD59x18.fromInt(2))
    );
```

3.135 CVF-135

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** MarginCalculator.sol

Recommendation These values should be precomputed.

Listing 135:

```
94  PRBMathSD59x18.fromInt(4)

99  PRBMathSD59x18.fromInt(4)

109 PRBMathSD59x18.fromInt(1) -

142         PRBMathSD59x18.fromInt(2),

145 (PRBMathSD59x18.fromInt(2))

298 PRBMathUD60x18.fromUint(1),

304 PRBMathUD60x18.fromUint(1),

334 PRBMathSD59x18.fromInt(1) -
```

3.136 CVF-136

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** MarginCalculator.sol

Description This expression is calculated twice.
Recommendation Consider calculating once and reusing.

Listing 136:

```
94  PRBMathSD59x18.fromInt(4)

99  PRBMathSD59x18.fromInt(4)
```

3.137 CVF-137

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** MarginCalculator.sol

Description This expression is calculated twice.
Recommendation Consider calculating once and reusing.

Listing 137:

```
142         PRBMathSD59x18.fromInt(2),

145     (PRBMathSD59x18.fromInt(2))
```

3.138 CVF-138

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** MarginCalculator.sol

Recommendation This code could be simplified as: apyBoundVars.criticalValueWad = PRBMathSD59x18.mul(isUpper ? _marginCalculatorParameters.xiUpperWad : _marginCalculatorParameters.xiLowerWad, PRBMathSD59x18.sqrt (apyBoundVars.criticalValueMultiplierWad);

Listing 138:

```
148 if (isUpper) {
        apyBoundVars.criticalValueWad = PRBMathSD59x18.mul(
150             _marginCalculatorParameters.xiUpperWad,
                PRBMathSD59x18.sqrt(apyBoundVars.
                    ↪ criticalValueMultiplierWad)
        );
    } else {
        apyBoundVars.criticalValueWad = PRBMathSD59x18.mul(
                _marginCalculatorParameters.xiLowerWad,
                PRBMathSD59x18.sqrt(apyBoundVars.
                    ↪ criticalValueMultiplierWad)
        );
    }
```

3.139 CVF-139

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** MarginCalculator.sol

Recommendation This code could be simplified as: int256 apyBoundIntWad = PRBMathSD59x18.mul(apyBoundVars.zetaWad, (apyBoundVars.kWad + apyBoundVars.lambdaWad + (isUpper ? apyBoundVars.criticalValueWad : -apyBoundVars.criticalValueWad));

Listing 139:

```
160 int256 apyBoundIntWad = (isUpper)
        ? PRBMathSD59x18.mul(
            apyBoundVars.zetaWad,
            (apyBoundVars.kWad +
                apyBoundVars.lambdaWad +
                apyBoundVars.criticalValueWad)
        )
        : PRBMathSD59x18.mul(
            apyBoundVars.zetaWad,
            (apyBoundVars.kWad +
                apyBoundVars.lambdaWad -
170             apyBoundVars.criticalValueWad)
        );
```

3.140 CVF-140

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** MarginCalculator.sol

Description The result is multiplied by "timeInYearsFromStartUntilMaturityWad" in all four branches.
Recommendation Consider multiplying in one place after the conditional statements.

Listing 140:

```
204 variableFactorWad = PRBMathUD60x18.mul(

212     timeInYearsFromStartUntilMaturityWad
    );

215 variableFactorWad = PRBMathUD60x18.mul(

226     timeInYearsFromStartUntilMaturityWad
    );

231 variableFactorWad = PRBMathUD60x18.mul(

238     ),
        timeInYearsFromStartUntilMaturityWad

242 variableFactorWad = PRBMathUD60x18.mul(

252     ),
        timeInYearsFromStartUntilMaturityWad
```

3.141 CVF-141

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** MarginCalculator.sol

Recommendation This transformation could be implemented as multiplication by 2^96 / 10^18, i.e. by 79228162514
Client Comment Using a hardcoded number is less accurate, prefer to keep the current implementaiton.

Listing 141:

```
297 simulatedUnwindLocalVars.sqrtRatioCurrWad = FullMath.mulDiv(
        PRBMathUD60x18.fromUint(1),
        sqrtRatioCurrX96,
300     FixedPoint96.Q96
    );
```

3.142 CVF-142

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** MarginCalculator.sol

Description When the denominator is a power of two, it would be be more efficient to perform mul+shift instead of mul+div.
Recommendation Consider implementing a "mulShr" function similar to the "mulDiv" function.
Client Comment Acknowledged.

Listing 142:

```
300 FixedPoint96.Q96
```

3.143 CVF-143

- **Severity** Major
- **Status** Fixed
- **Category** Overflow/Underflow
- **Source** MarginCalculator.sol

Description Overflow is possible when converting to "int256".
Recommendation Consider using safe conversion.

Listing 143:

```
329 (-int256(gammaWad)),
```

3.144 CVF-144

- **Severity** Major
- **Status** Fixed
- **Category** Overflow/Underflow
- **Source** MarginCalculator.sol

Description Underflow is possible when converting to "uint256".
Recommendation Consider using safe conversion.

Listing 144:

```
339 uint256(simulatedUnwindLocalVars.oneMinusTimeFactorWad)
```

3.145 CVF-145

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** FixedAndVariableMath.sol

Description We didn't review these files.

Listing 145:

```
4 import "prb-math/contracts/PRBMathSD59x18.sol";
  import "prb-math/contracts/PRBMathUD60x18.sol";
```

3.146 CVF-146

- **Severity** Minor
- **Status** Fixed
- **Category** Readability
- **Source** FixedAndVariableMath.sol

Recommendation This value could be rendered as: 31536000e18

Listing 146:

```
14 uint256 public constant SECONDS_IN_YEAR_IN_WAD = 31536000 *
     ↪ 10**18;
```

3.147 CVF-147

- **Severity** Minor
- **Status** Fixed
- **Category** Readability
- **Source** FixedAndVariableMath.sol

Recommendation This value could be rendered as: 100e18

Listing 147:

```
15 uint256 public constant ONE_HUNDRED_IN_WAD = 100 * 10**18;
```

3.148 CVF-148

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** FixedAndVariableMath.sol

Recommendation Using per-second interest rates instead of annualized interest rates would make calculations simpler and cheaper.
Client Comment Good idea, but since this would require quite a bit of refactoring, we'd prefer to keep the current setup, but consider this approach for v2.

Listing 148:

```
14 uint256 public constant SECONDS_IN_YEAR_IN_WAD = 31536000 *
    ↪ 10**18;
```

3.149 CVF-149

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** FixedAndVariableMath.sol

Recommendation The "using" statement could be used to make invocations of library functions shorter and easier to read.

Listing 149:

```
36  int256 fixedTokenBalanceWad = PRBMathSD59x18.fromInt(
        ↪ fixedTokenBalance);
    int256 variableTokenBalanceWad = PRBMathSD59x18.fromInt(

41  int256 fixedCashflowWad = PRBMathSD59x18.mul(

48  int256 variableCashflowWad = PRBMathSD59x18.mul(

56  cashflow = PRBMathSD59x18.toInt(cashflowWad);

67  timeInYearsWad = PRBMathUD60x18.div(

101 fixedFactorValueWad = PRBMathUD60x18.div(

124 int256 exp1Wad = PRBMathSD59x18.mul(

136 fixedTokenBalanceWad = PRBMathSD59x18.div(

167 accruedValues.excessFixedAccruedBalanceWad = PRBMathSD59x18.mul(

174 accruedValues.excessVariableAccruedBalanceWad = PRBMathSD59x18.
        ↪ mul(

210 int256 amount0Wad = PRBMathSD59x18.fromInt(amount0);
    int256 amount1Wad = PRBMathSD59x18.fromInt(amount1);

230 fixedTokenBalance = PRBMathSD59x18.toInt(fixedTokenBalanceWad);
```

3.150 CVF-150

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** FixedAndVariableMath.sol

Description The expression "Time.blockTimestampScaled()" is calculated several times.
Recommendation Consider calculating once and reusing.

Listing 150:

```
86 require(Time.blockTimestampScaled() >= termStartTimestampWad, "B
   ↪ .T<S");

91     atMaturity || (Time.blockTimestampScaled() >=
          ↪ termEndTimestampWad)

96         Time.blockTimestampScaled() -
```

3.151 CVF-151

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** FixedAndVariableMath.sol

Recommendation This function could be simplified as: return amount0Wad - PRBMathSD59x18.div (excessBalanceWad, fixedFactor (true, termStartTimestampWad, termEndTimestampWad);

Listing 151:

```
113 function calculateFixedTokenBalance(
```

3.152 CVF-152

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** FixedAndVariableMath.sol

Description This expression is calculated twice.
Recommendation Consider calculating once and reusing.

Listing 152:

```
127 fixedFactor(true, termStartTimestampWad, termEndTimestampWad)

139 fixedFactor(true, termStartTimestampWad, termEndTimestampWad)
```

3.153 CVF-153

- **Severity** Minor
- **Status** Fixed
- **Category** Bad naming
- **Source** FixedAndVariableMath.sol

Description Variable names are unclear.
Recommendation Consider using "amountFixed" and "amountVariable" instead.

Listing 153:

```
152 /// @param amount0Wad A fixed balance
    /// @param amount1Wad A variable balance

189 /// @param amount0 A fixed balance
190 /// @param amount1 A variable balance
```

3.154 CVF-154

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** FixedAndVariableMath.sol

Description Using a structure here is redundant.
Recommendation Just use local variables instead.

Listing 154:

```
165 AccruedValues memory accruedValues;
```

3.155 CVF-155

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** FixedAndVariableMath.sol

Recommendation This expression could be simplified as: if (amount0 == 0 || amount1 == 0 || amount0 ^ amount1 >= 0)

Listing 155:

```
203 !(((amount0 <= 0 && amount1 >= 0) ||
        (amount0 >= 0 && amount1 <= 0)) ||
        (amount0 == 0 && amount1 == 0))
```

3.156 CVF-156

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** FixedAndVariableMath.sol

Recommendation This part is redundant, as it is already covered be either of the previous parts.

Listing 156:

```
205 (amount0 == 0 && amount1 == 0))
```

3.157 CVF-157

- **Severity** Moderate
- **Status** Fixed
- **Category** Unclear behavior
- **Source** FixedAndVariableMath.sol

Description It is unclear, why both amounts for a position cannot have the same sign and why to forbid this situation.
Recommendation Consider allowing positions with amounts having the same sign.

Listing 157:

```
207 revert AmountSignsSame();
```

3.158 CVF-158

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** FixedAndVariableMath.sol

Recommendation This check should be performed earlier to save gas.

Listing 158:

```
213 require(termEndTimestampWad > termStartTimestampWad, "E<=S");
```

3.159 CVF-159

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** OracleBuffer.sol

Description This constant is redundant.
Recommendation Use “type(uint216).max” instead.

Listing 159:

```
12 uint256 private constant MAX_UINT216 = 2**216 - 1;
```

3.160 CVF-160

- **Severity** Minor
- **Status** Info
- **Category** Readability
- **Source** OracleBuffer.sol

Recommendation Consider wrapping the "Observation[65535]" type into a struct to make code easier to read.
Client Comment Would prefer to keep the current implementation.

Listing 160:

```
48  Observation[65535] storage self,
69  Observation[65535] storage self,
98  Observation[65535] storage self,
122 Observation[65535] storage self,
168 Observation[65535] storage self,
```

3.161 CVF-161

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** OracleBuffer.sol

Recommendation The value "65535" should be a named constant.

Listing 161:

```
48  Observation[65535] storage self,
69  Observation[65535] storage self,
98  Observation[65535] storage self,
122 Observation[65535] storage self,
168 Observation[65535] storage self,
```

3.162 CVF-162

- **Severity** Minor
- **Status** Info
- **Category** Documentation
- **Source** OracleBuffer.sol

Description It seems that the actual implementation may never return the same observation.
Recommendation Consider either fixing the comment or the implementation.
Client Comment Would prefer to preserve code's equivalence to Uniswap's Oracle.sol.

Listing 162:

```
112 /// The result may be the same observation, or adjacent
    ↪ observations.
```

3.163 CVF-163

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** OracleBuffer.sol

Recommendation Right shift would be more efficient than division by 2.

Listing 163:

```
135 i = (l + r) / 2;
```

3.164 CVF-164

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** OracleBuffer.sol

Recommendation These "% cardinality" parts could be avoided by checking before the loop, whether the desired observation is before 0 and "index" or between "index" and "cardinality.
Client Comment Would prefer to preserve code's equivalence to Uniswap's Oracle.sol.

Listing 164:

```
137 beforeOrAt = self[i % cardinality];

145 atOrAfter = self[(i + 1) % cardinality];
```

3.165 CVF-165

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** OracleBuffer.sol

Description Usually, during binary search it is enough to look at one element per iteration, but this implementation looks at two adjacent elements.
Recommendation Consider refactoring the code to look at one element only.
Client Comment Would prefer to preserve code's equivalence to Uniswap's Oracle.sol.

Listing 165:

```
145 atOrAfter = self[(i + 1) % cardinality];
```

3.166 CVF-166

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** OracleBuffer.sol

Description Here a return value is used as a local variable which is a bad practice. It makes code harder to read.
Recommendation Consider using a separate local variable instead.
Client Comment Would prefer to preserve code's equivalence to Uniswap's Oracle.sol.

Listing 166:

```
193 beforeOrAt = self[(index + 1) % cardinality];
```

3.167 CVF-167

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** AaveRateOracle.sol

Recommendation The type of this variable should be "IAaveV2LendingPool".

Listing 167:

```
20 address public override aaveLendingPool;
```

3.168 CVF-168

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** AaveRateOracle.sol

Recommendation Constants are usually named IN_UPPER_CASE.

Listing 168:

```
22 uint8 public constant override underlyingYieldBearingProtocolID
   ↪ = 1; // id of aave v2 is 1
```

3.169 CVF-169

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** AaveRateOracle.sol

Recommendation The argument types should be: "IAaveV2LendingPool" and "IERC20" respectively.

Listing 169:

```
24 constructor(address _aaveLendingPool, address underlying)
```

3.170 CVF-170

- **Severity** Minor
- **Status** Fixed
- **Category** Documentation
- **Source** AaveRateOracle.sol

Description This code appears several times.
Recommendation Consider extracting to a function.

Listing 170:

```
29 uint256 result = IAaveV2LendingPool(aaveLendingPool)
30     .getReserveNormalizedIncome(underlying);

54 uint256 resultRay = IAaveV2LendingPool(aaveLendingPool)
       .getReserveNormalizedIncome(underlying);

163         rateValueRay = IAaveV2LendingPool(aaveLendingPool)
                .getReserveNormalizedIncome(underlying);

171 uint256 currentValueRay = IAaveV2LendingPool(aaveLendingPool)
       .getReserveNormalizedIncome(underlying);
```

3.171 CVF-171

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

Description Underflow is possible here.
Recommendation Consider rewriting like this: if ‘(blockTimestamp - last.blockTimestamp < minSecondsSinceLastUpdate)

Listing 171:

```
51 if (blockTimestamp - minSecondsSinceLastUpdate < last.
   ↪ blockTimestamp)
```

3.172 CVF-172

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** AaveRateOracle.sol

Description There is no explicit check that “_from <= _to”.
Recommendation Consider adding such check.

Listing 172:

```
86 uint256 _from,
uint256 _to //  move docs to IRateOracle. Add additional
   ↪ parameter to use cache and implement cache.
```

3.173 CVF-173

- **Severity** Minor
- **Status** Fixed
- **Category** Unclear behavior
- **Source** AaveRateOracle.sol

Recommendation The Aave V2 protocol implementation doesn’t seem to support negative rates: https://github.com/aave/protocol-v2/blob/master/contracts/protocol/libraries/math/MathUtils.sol#L29

Listing 173:

```
121 /// is this precise, have there been instances where the aave
   ↪ rate is negative?
```

3.174 CVF-174

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** AaveRateOracle.sol

Recommendation This value could be precomputed.

Listing 174:

```
145 PRBMathUD60x18.fromUint(1);
```

3.175 CVF-175

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** AaveRateOracle.sol

Description String error messages are suboptimal.
Recommendation Consider using named errors.
Client Comment Acknowledged.

Listing 175:

```
157 require(currentTime >= queriedTime, "OOO");
```

3.176 CVF-176

- **Severity** Minor
- **Status** Info
- **Category** Unclear behavior
- **Source** AaveRateOracle.sol

Recommendation In case both surrounding observations have the same value, this value should be returned immediately. No need to do interpolation.
Client Comment While this is true, we don't see it being a case we hit often in the real world, so the scope for gas savings is minimal. No objection to the change but also would prefer to keep the implementation we have for v1.

Listing 176:

```
197 if (atOrAfter.observedValue > beforeOrAt.observedValue) {
```

3.177 CVF-177

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** BaseRateOracle.sol

Description We didn't review this file.
Client Comment Acknowledged.

Listing 177:

```
8 import "prb-math/contracts/PRBMathUD60x18.sol";
```

3.178 CVF-178

- **Severity** Minor
- **Status** Fixed
- **Category** Bad naming
- **Source** BaseRateOracle.sol

Description The name is confusing. One wei is 1e-18 ether, not 1e18 or something.
Recommendation Consider renaming.

Listing 178:

```
18 uint256 public constant ONE_WEI = 10**18;
```

3.179 CVF-179

- **Severity** Minor
- **Status** Fixed
- **Category** Readability
- **Source** BaseRateOracle.sol

Recommendation This value could be rendered as "1e18".

Listing 179:

```
18 uint256 public constant ONE_WEI = 10**18;
```

3.180 CVF-180

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** BaseRateOracle.sol

Recommendation It would be more efficient to use a single key of 64-bits encapsulating both, start and end time.
Client Comment Would prefer to keep the current implementation.

Listing 180:

```
21 mapping(uint32 => mapping(uint32 => uint256)) public
    ↪ settlementRateCache;
```

3.181 CVF-181

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** BaseRateOracle.sol

Recommendation The type of this variable should be "IERC20".

Listing 181:

```
32 address public immutable override underlying;
```

3.182 CVF-182

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** BaseRateOracle.sol

Description There is no range check for the argument.
Recommendation Consider adding an appropriate check.
Client Comment Ideally the minimum should not be zero, since the oracle buffer size is not infinite and theoretically we could have someone writing a value to the buffer every block.

Listing 182:

```
43 function setMinSecondsSinceLastUpdate(uint256
    ↪ _minSecondsSinceLastUpdate)
```

3.183 CVF-183

- **Severity** Minor
- **Status** Fixed
- **Category** Unclear behavior
- **Source** BaseRateOracle.sol

Description This event is emitted even if nothing actually changed.

Listing 183:

```
50 emit MinSecondsSinceLastUpdateSet(_minSecondsSinceLastUpdate);
```

3.184 CVF-184

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** BaseRateOracle.sol

Recommendation The argument type should be "IERC20".

Listing 184:

```
53 constructor(address _underlying) {
```

3.185 CVF-185

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** BaseRateOracle.sol

Recommendation Using time in seconds and per-second rate, instead of time in year and annual rate, would make the code more efficient, as the power would be integer and thus could be calculated more efficiently.
Client Comment Agree with the suggestion, however, at this stage such a change would require quite a bit of refactoring, hence we feel this is something we'd consider for v2.

Listing 185:

```
82 /// @param timeInYearsWad Time in years for the period for which
   ↪ we want to calculate the apy (in wei)
```

3.186 CVF-186

- **Severity** Minor
- **Status** Info
- **Category** Documentation
- **Source** BaseRateOracle.sol

Description The semantics of this function is unclear.
Recommendation Consider documenting.
Client Comment Acknowledged.

Listing 186:

```
101 function getRateFromTo(uint256 from, uint256 to)
```

3.187 CVF-187

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** BaseRateOracle.sol

Description These declarations are redundant and have no effect, as the same functions are already declared in the "IRateOracle" interface.
Client Comment Acknowledged.

Listing 187:

```
101 function getRateFromTo(uint256 from, uint256 to)
        public
        view
        virtual
        override
        returns (uint256);

197 function writeOracleEntry() external virtual override;
```

3.188 CVF-188

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** Periphery.sol

Recommendation The type of this field should be "IMarginEngine".

Listing 188:

```
 13 address marginEngineAddress;

 81 address marginEngineAddress;
```

3.189 CVF-189

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** Periphery.sol

Description This function just returns its argument.
Recommendation Consider removing this function.

Listing 189:

```
21 function getMarginEngine(address marginEngineAddress)
```

3.190 CVF-190

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** Periphery.sol

Description This variable is redundant.
Recommendation Just return the expression value.

Listing 190:

```
26 IMarginEngine marginEngine = IMarginEngine(marginEngineAddress);
```

3.191 CVF-191

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** Periphery.sol

Description This variable is redundant.
Recommendation Just return the expression value.

Listing 191:

```
37 IVAMM vamm = marginEngine.vamm();
```

3.192 CVF-192

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** Periphery.sol

Description This check makes the "params.recipient" field redundant.
Recommendation Just use "msg.sender" instead.

Listing 192:

```
44 require(
       msg.sender == params.recipient,
       "msg.sender must be the recipient"
   );

91 require(
       msg.sender == params.recipient,
       "msg.sender must be the recipient"
   );
```

3.193 CVF-193

- **Severity** Moderate
- **Status** Fixed
- **Category** Overflow/Underflow
- **Source** Periphery.sol

Description Overflow is possible here.

Listing 193:

```
101 amountSpecified = int256(params.notional);

103 amountSpecified = -int256(params.notional);
```

3.194 CVF-194

- **Severity** Major
- **Status** Fixed
- **Category** Flaw
- **Source** Periphery.sol

Description Zero is a valid tick index, but here zero is used as a special value. So it is impossible to specify, say tickLower = 0, tickUpper = 5.
Recommendation Consider using an invalid tick index as a special value.

Listing 194:

```
119 tickLower: params.tickLower == 0 ? -tickSpacing : params.
    ↪ tickLower,
120 tickUpper: params.tickUpper == 0 ? tickSpacing : params.
    ↪ tickUpper
```

3.195 CVF-195

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** TraderWithYieldBearingAssets.sol

Recommendation These variables are redundant. Just update the fields of "self" via "+=" operators.

Listing 195:

```
36 int256 fixedTokenBalance = _self.fixedTokenBalance +
       fixedTokenBalanceDelta;

39 int256 variableTokenBalance = _self.variableTokenBalance +
40     variableTokenBalanceDelta;
```

3.196 CVF-196

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** SwapMath.sol

Description We didn't review these files.
Client Comment Acknowledged.

Listing 196:

```
7 import "prb-math/contracts/PRBMathUD60x18.sol";
  import "prb-math/contracts/PRBMathSD59x18.sol";
```

3.197 CVF-197

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

Description Overflow is possible when calculating "-amountRemaining" and such overflow will cause revert.
Recommendation Consider wrapping into the "unchecked" block.

Listing 197:

```
97      if (uint256(-amountRemaining) >= amountOut)

103                 uint256(-amountRemaining),

154 if (!exactIn && amountOut > uint256(-amountRemaining)) {

156     amountOut = uint256(-amountRemaining);
```

3.198 CVF-198

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** SwapMath.sol

Description The expression "uint256(-amountRemaining)" is calculated several times.
Recommendation Consider calculating once and reusing.

Listing 198:

```
97      if (uint256(-amountRemaining) >= amountOut)

103                 uint256(-amountRemaining),

154 if (!exactIn && amountOut > uint256(-amountRemaining)) {

156     amountOut = uint256(-amountRemaining);
```

3.199 CVF-199

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** SafeTransferLib.sol

Recommendation Should be "^0.8.0".

Listing 199:

```
2 pragma solidity >=0.8.0;
```

3.200 CVF-200

- **Severity** Minor
- **Status** Info
- **Category** Readability
- **Source** SafeTransferLib.sol

Recommendation These functions could be implemented in pure Solidity. No need for assembly.
Client Comment Would prefer to keep unchanged (using as an external library).

Listing 200:

```
16  function safeTransferETH(address to, uint256 amount) internal {

31  function safeTransferFrom(

69  function safeTransfer(

102 function safeApprove(
```

3.201 CVF-201

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** SafeTransferLib.sol

Recommendation Applying these masks is redundant as the values are are addresses, i.e. guaranteed to fit into 160 bits. Is there a practical scenario how an attacker may substitute values that don't fit into 160 bits?
Client Comment Would prefer to keep unchanged (using as an external library).

Listing 201:

```
50  and(from, 0xffffffffffffffffffffffffffffffffffffffff)

54  and(to, 0xffffffffffffffffffffffffffffffffffffffff)

87  and(to, 0xffffffffffffffffffffffffffffffffffffffff)

120 and(to, 0xffffffffffffffffffffffffffffffffffffffff)
```

3.202 CVF-202

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** SafeTransferLib.sol

Recommendation The mask should be a named constant.
Client Comment Would prefer to keep unchanged (using as an external library).

Listing 202:

```
50  and(from, 0xffffffffffffffffffffffffffffffffffffffff)

54  and(to, 0xffffffffffffffffffffffffffffffffffffffff)

87  and(to, 0xffffffffffffffffffffffffffffffffffffffff)

120 and(to, 0xffffffffffffffffffffffffffffffffffffffff)
```

3.203 CVF-203

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** SafeTransferLib.sol

Recommendation This function could be rewritten in pure Solidity in case the returned data would be passed to it as a bytes array.
Client Comment Would prefer to keep unchanged (using as an external library).

Listing 203:

```
136 function didLastOptionalReturnCallSucceed(bool callStatus)
```

3.204 CVF-204

- **Severity** Minor
- **Status** Info
- **Category** Unclear behavior
- **Source** SafeTransferLib.sol

Description For some failed transactions this function reverts, while for other it return false.
Recommendation Consider using a single way to signal a failed transaction, i.e. either always revert of always return false.
Client Comment Would prefer to keep unchanged (using as an external library).

Listing 204:

```
139 returns (bool success)
```

3.205 CVF-205

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** Tick.sol

Recommendation This commented out import should be removed.

Listing 205:

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

3.206 CVF-206

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** Tick.sol

Recommendation This could be optimized and simplified as int24 minTick = TickMath.MIN_TICK - TickMath.MIN_TICK % tockSpacing;

Listing 206:

```
42 int24 minTick = (TickMath.MIN_TICK / tickSpacing) * tickSpacing;
```

3.207 CVF-207

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** Tick.sol

Recommendation This could be optimized and simplified as int24 maxTick = TickMath.MAX_TICK - TickMath.MAX_TICK % tockSpacing; or even as: int24 maxTick = -minTick; taking into account that TickMath.MAX_TICK = -TickMath.MIN_TICK

Listing 207:

```
43 int24 maxTick = (TickMath.MAX_TICK / tickSpacing) * tickSpacing;
```

3.208 CVF-208

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** Tick.sol

Description These three functions are very similar, while the first one accepts all arguments separately, while the two other functions wrap several arguments into a structure.
Recommendation Consider using consistent API for similar functions.

Listing 208:

```
55  function getFeeGrowthInside(
        mapping(int24 => Tick.Info) storage self,
        int24 tickLower,
        int24 tickUpper,
        int24 tickCurrent,
60      uint256 feeGrowthGlobalX128
    ) internal view returns (uint256 feeGrowthInsideX128) {

101 function getVariableTokenGrowthInside(
        mapping(int24 => Tick.Info) storage self,
        VariableTokenGrowthInsideParams memory params
    ) internal view returns (int256 variableTokenGrowthInsideX128) {

143 function getFixedTokenGrowthInside(
        mapping(int24 => Tick.Info) storage self,
        FixedTokenGrowthInsideParams memory params
    ) internal view returns (int256 fixedTokenGrowthInsideX128) {
```

3.209 CVF-209

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** Tick.sol

Description These functions are very similar.
Recommendation Consider extracting common parts into utility functions to reduce code duplication.

Listing 209:

```
55  function getFeeGrowthInside(

101 function getVariableTokenGrowthInside(

143 function getFixedTokenGrowthInside(
```

3.210 CVF-210

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** Tick.sol

Description This code seems redundant.
Recommendation It would be more efficient to assume that all the previous growth happened "inside", rather than below, i.e. at the same side of the current tick. With such assumption, all the growth counter could remain zero.
Client Comment Would keep the implementation aligned with that of Uni v3: https://github.com/Uniswap/v3-core/blob/ed88be38ab2032d82bf10ac6f8d03aa631889d48/contracts/libraries/Tick.sol#L133

Listing 210:

```
210 // by convention, we assume that all growth before a tick was
    ↪ initialized happened _below_ the tick
    if (tick <= tickCurrent) {
        info.feeGrowthOutsideX128 = feeGrowthGlobalX128;

        info.fixedTokenGrowthOutsideX128 =
            ↪ fixedTokenGrowthGlobalX128;

        info
            .variableTokenGrowthOutsideX128 =
                ↪ variableTokenGrowthGlobalX128;
    }
```

3.211 CVF-211

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** AaveDataTypes.sol

Description This library contains only struct definitions. Solidity allows defining structs on the top level, outside contracts and libraries.
Recommendation Consider moving struct definitions to the top level.
Client Comment Would prefer to keep the current setup for Readability.

Listing 211:

```
4 library AaveDataTypes {
```

3.212 CVF-212

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** Time.sol

Description We didn't review this file.

Listing 212:

```
4 import "prb-math/contracts/PRBMathUD60x18.sol";
```

3.213 CVF-213

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** Time.sol

Description This constant is redundant.
Recommendation Just use "type(uint32).max" instead.

Listing 213:

```
7 uint256 private constant MAX_UINT32 = 2**32 - 1;
```

3.214 CVF-214

- **Severity** Minor
- **Status** Fixed
- **Category** Readability
- **Source** Time.sol

Recommendation This value could be rendered as: 86400e18

Listing 214:

```
8 uint256 public constant SECONDS_IN_DAY_WAD = 86400 * 10**18; ///
    ↪ convert into WAD via PRB
```

3.215 CVF-215

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** Time.sol

Recommendation "This could be simplified as: require ((timestamp = uint32(_timestamp)) == _timestamp, ""TSOFLOW"");"

Listing 215:

```
27 require(_timestamp <= MAX_UINT32, "TSOFLOW");
return uint32(_timestamp);
```

3.216 CVF-216

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** Time.sol

Recommendation This could be simplified as: return Time.blockTimestampScaled() + SECOND_IN_DAY_WAD >= termEndTimestampWad;

Listing 216:

```
36 uint256 currentTimestamp = Time.blockTimestampScaled();

38 if (currentTimestamp >= termEndTimestampWad) {
       vammInactive = true;
40 } else {
       uint256 timeDelta = termEndTimestampWad - currentTimestamp;
       if (timeDelta <= SECONDS_IN_DAY_WAD) {
           vammInactive = true;
       }
   }
```

3.217 CVF-217

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** WayRayMath.sol

Recommendation This library should be defined in a file named "WadRayMath.sol".

Listing 217:

```
14 library WadRayMath {
```

3.218 CVF-218

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** WayRayMath.sol

Recommendation The value of this constant should be derived from the "WAD" and "RAY" constants.
Client Comment External library from Aave.

Listing 218:

```
21 uint256 internal constant WAD_RAY_RATIO = 1e9;
```

3.219 CVF-219

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** WayRayMath.sol

Description These checks are redundant as Solidity anyway performs overflow checks.
Recommendation Consider either removing these checks or wrapping the calculations into "unchecked" blocks.

Listing 219:

```
 63  require(
         a <= (type(uint256).max - halfWAD) / b,
         Errors.MATH_MULTIPLICATION_OVERFLOW
     );

 81  require(
         a <= (type(uint256).max - halfB) / WAD,
         Errors.MATH_MULTIPLICATION_OVERFLOW
     );

100  require(
         a <= (type(uint256).max - halfRAY) / b,
         Errors.MATH_MULTIPLICATION_OVERFLOW
     );

118  require(
         a <= (type(uint256).max - halfB) / RAY,
120      Errors.MATH_MULTIPLICATION_OVERFLOW
     );

134  require(result >= halfRatio, Errors.MATH_ADDITION_OVERFLOW);

146  require(
         result / WAD_RAY_RATIO == a,
         Errors.MATH_MULTIPLICATION_OVERFLOW
     );
```

3.220 CVF-220

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

Description Phantom overflow is possible here, i.e. a situation when the final calculation result would fit into the destination type, while some intermediary operations overflow.
Recommendation Consider using the "mulDiv" function or some other approach resistant to phantom overflows.
Client Comment Acknowledged.

Listing 220:

```
68  return (a * b + halfWAD) / WAD;

86  return (a * WAD + halfB) / b;

105 return (a * b + halfRAY) / RAY;

123 return (a * RAY + halfB) / b;
```

3.221 CVF-221

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** WayRayMath.sol

Recommendation This variable should be turned into a named constant.
Client Comment Would keep the same as in the library.

Listing 221:

```
132 uint256 halfRatio = WAD_RAY_RATIO / 2;
```

3.222 CVF-222

- **Severity** Major
- **Status** Info
- **Category** Overflow/Underflow
- **Source** WayRayMath.sol

Description This "require" statement checks for a phantom overflow, as conversion from RAY to WAD is always possible.
Recommendation Consider refactoring the code to never revert.
Client Comment Could you please elaborate on the recommendation, show a practical implementation of the suggestion?

Listing 222:

```
134 require(result >= halfRatio, Errors.MATH_ADDITION_OVERFLOW);
```

3.223 CVF-223

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** Position.sol

Description We didn't review these files.
Client Comment Acknowledged.

Listing 223:

```
8 import "prb-math/contracts/PRBMathSD59x18.sol";
  import "prb-math/contracts/PRBMathUD60x18.sol";
```

3.224 CVF-224

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** Position.sol

Recommendation The "Position." prefix before "Info" is redundant, as this code is located inside the "Position" library.

Listing 224:

```
15 using Position for Position.Info;

55 ) internal view returns (Position.Info storage position) {
```

3.225 CVF-225

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** Position.sol

Description Unlike names of the other fields in this structure, this name has the underscore ("_") prefix.
Recommendation Consider using a consistent naming policy and removing the prefix.
Client Comment Would prefer to keep the current naming, liquidity is an exception since the VAMM also has a liqudiity variable which conflicts with the position's liquidity attribute.

Listing 225:

```
25 uint128 _liquidity;
```

3.226 CVF-226

- **Severity** Minor
- **Status** Fixed
- **Category** Unclear behavior
- **Source** Position.sol

Description There are no range checks for these values.
Recommendation Consider adding appropriate checks.

Listing 226:

```
53 int24 tickLower,
   int24 tickUpper
```

3.227 CVF-227

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** Position.sol

Description Here the full structure is loaded from the storage into the memory, while only a few fields are actually used.
Recommendation Consider reading only those fields that are actually needed.
Client Comment Would prefer to keep the current implementation since when extracting filed that are actually needed the function is no-longer pure.

Listing 227:

```
 71 Info memory _self = self;

 85     Info memory _self = self;

105 Info memory _self = self;

131 Info memory _self = self;

205 Info memory _self = self;
```

3.228 CVF-228

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** Position.sol

Recommendation This line could be simplified using the "+=" operator.

Listing 228:

```
72 self.margin = _self.margin + marginDelta;

87      self.fixedTokenBalance =
            _self.fixedTokenBalance +
            fixedTokenBalanceDelta;
90      self.variableTokenBalance =
            _self.variableTokenBalance +
            variableTokenBalanceDelta;
```

3.229 CVF-229

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** Position.sol

Recommendation This could be optimized as: if (fixedTokenBalanceDelta | variableTokenBalanceDelta != 0) {

Listing 229:

```
84 if (fixedTokenBalanceDelta != 0 || variableTokenBalanceDelta !=
   ↪ 0) {
```

3.230 CVF-230

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** Position.sol

Description When the denominator is a power of two, it would be more efficient to use mul+shift rather than mul+div.
Recommendation Consider implementing a non-overflowing "mulShr" function.
Client Comment Would prefer to keep the implementation aligned with that of Uni v3, ref: https://github.com/Uniswap/v3-core/blob/ed88be38ab2032d82bf10ac6f8d03aa631889d48/contracts/libraries/Position.sol#L66

Listing 230:

```
108 _feeDelta = FullMath.mulDiv(
        feeGrowthInsideX128 - _self.feeGrowthInsideLastX128,
110     _self._liquidity,
        FixedPoint128.Q128
    );

138     FullMath.mulDiv(
            uint256(fixedTokenGrowthInsideDeltaX128),
140         _self._liquidity,
            FixedPoint128.Q128
        )

146     FullMath.mulDiv(
            uint256(-fixedTokenGrowthInsideDeltaX128),
            _self._liquidity,
            FixedPoint128.Q128
150     )

159     FullMath.mulDiv(
160         uint256(variableTokenGrowthInsideDeltaX128),
            _self._liquidity,
            FixedPoint128.Q128
        )

167     FullMath.mulDiv(
            uint256(-variableTokenGrowthInsideDeltaX128),
            _self._liquidity,
170         FixedPoint128.Q128
        )
```

3.231 CVF-231

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** Position.sol

Description These code blocks are very similar.
Recommendation Consider extracting a signed version of the "mulDiv" function and using it here.

Listing 231:

```
136 if (fixedTokenGrowthInsideDeltaX128 > 0) {
        _fixedTokenDelta = int256(
            FullMath.mulDiv(
                uint256(fixedTokenGrowthInsideDeltaX128),
140             _self._liquidity,
                FixedPoint128.Q128
            )
        );
    } else {
        _fixedTokenDelta = -int256(
            FullMath.mulDiv(
                uint256(-fixedTokenGrowthInsideDeltaX128),
                _self._liquidity,
                FixedPoint128.Q128
150         )
        );
    }

157 if (variableTokenGrowthInsideDeltaX128 > 0) {
        _variableTokenDelta = int256(
            FullMath.mulDiv(
160             uint256(variableTokenGrowthInsideDeltaX128),
                _self._liquidity,
                FixedPoint128.Q128
            )
        );
    } else {
        _variableTokenDelta = -int256(
            FullMath.mulDiv(
                uint256(-variableTokenGrowthInsideDeltaX128),
                _self._liquidity,
170             FixedPoint128.Q128
            )
        );
    }
```

3.232 CVF-232

- **Severity** Moderate
- **Status** Fixed
- **Category** Overflow/Underflow
- **Source** Position.sol

Description Overflow is possible here.
Recommendation Consider using safe conversion.

Listing 232:

```
137 _fixedTokenDelta = int256(
145 _fixedTokenDelta = -int256(
158 _variableTokenDelta = int256(
166 _variableTokenDelta = -int256(
```

3.233 CVF-233

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

Description Underflow is possible here that will lead to transaction revert.
Recommendation Consider converting to int256 before negating.
Client Comment The number is already int256.

Listing 233:

```
147 uint256(-fixedTokenGrowthInsideDeltaX128),
168 uint256(-variableTokenGrowthInsideDeltaX128),
```

3.234 CVF-234

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** Position.sol

Description This assignment doesn't have any effect, as the assigned value is never used.
Recommendation Consider removing it.

Listing 234:

```
210 liquidityNext = _self._liquidity;
```

3.235 CVF-235

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** Position.sol

Recommendation This conditional statement should be merged with the previous conditions statement, as its condition is the inversion of the condition of the previous statement.

Listing 235:

```
218 if (liquidityDelta != 0) self._liquidity = liquidityNext;
```

3.236 CVF-236

- **Severity** Major
- **Status** Fixed
- **Category** Suboptimal
- **Source** TickMath.sol

Description These conditions are always false, as the maximum "absTick" value is 0x10DEC.
Recommendation Consider removing these lines.

Listing 236:

```
71 if (absTick & 0x20000 != 0)
       ratio = (ratio * 0x5d6af8dedb81196699c329225ee604) >> 128;
   if (absTick & 0x40000 != 0)
       ratio = (ratio * 0x2216e584f5fa1ea926041bedfe98) >> 128;
   if (absTick & 0x80000 != 0)
       ratio = (ratio * 0x48a170391f7dc42444e8fa2) >> 128;
```

3.237 CVF-237

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** TickMath.sol

Description As the tick range is reduced in comparison with Uniswap V3, it would probably be possible to reduce the number of iterations in this function.
Recommendation See the following link for details: https://github.com/Uniswap/v3-core/issues/500#issuecomment-1035387896
Client Comment Acknowledged.

Listing 237:

```
93 function getTickAtSqrtRatio(uint160 sqrtPriceX96)
```

3.238 CVF-238

- **Severity** Minor
- **Status** Info
- **Category** Bad naming
- **Source** BitMath.sol

Description String error messages are inefficient.
Recommendation Consider using named errors instead.
Client Comment Want to keep string error messages.

Listing 238:

```
15 require(x > 0, "x must be > 0");

55 require(x > 0, "x must be > 0");
```

3.239 CVF-239

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** BitMath.sol

Recommendation Can be ‘r-=x&0x1‘
Client Comment Would keep the same as in the library.

Listing 239:

```
93 if (x & 0x1 > 0) r -= 1;
```

3.240 CVF-240

- **Severity** Minor
- **Status** Info
- **Category** Bad naming
- **Source** Errors.sol

Description String error codes are inefficient.
Recommendation Consider using named errors.
Client Comment Want to keep as it is since external library.

Listing 240:

```
23 library Errors {
```

3.241 CVF-241

- **Severity** Major
- **Status** Fixed
- **Category** Suboptimal
- **Source** LiquidityMath.sol

Description The "-y" subexpression will underflow and thus revert in case "y" is -2^127.
Recommendation Consider nesting this code inside an "unchecked" block.

Listing 241:

```
13 require((z = x - uint128(-y)) < x, "LS");
```

3.242 CVF-242

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LiquidityMath.sol

Description These "require" statement are redundant as Solidity automatically does overflow and underflow checks.
Recommendation Consider removing thsm.

Listing 242:

```
13 require((z = x - uint128(-y)) < x, "LS");

15 require((z = x + uint128(y)) >= x, "LA");
```

3.243 CVF-243

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** FullMath.sol

Recommendation This expression could be optimized and simplified as: uint256 twos = uint256(-int256(denominator)) & denominator;
Client Comment Would keep the same as in the library.

Listing 243:

```
71 uint256 twos = (type(uint256).max - denominator + 1) &
   ↪ denominator;
```

3.244 CVF-244

- **Severity** Minor
- **Status** Info
- **Category** Bad naming
- **Source** IVAMM.sol

Recommendation Events are usually named via nouns, such as "VAMMInitialization", "FeeProtocol", etc.

Listing 244:

```
19 event InitializeVAMM(uint160 sqrtPriceX96, int24 tick);

40 event SetFeeProtocol(uint8 feeProtocolOld, uint8 feeProtocol);

43 event FeeSet(uint256 feeWadOld, uint256 feeWad);
```

3.245 CVF-245

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** IVAMM.sol

Recommendation The old values are redundant as they could be derived from the previous events.

Listing 245:

```
40 event SetFeeProtocol(uint8 feeProtocolOld, uint8 feeProtocol);

43 event FeeSet(uint256 feeWadOld, uint256 feeWad);
```

3.246 CVF-246

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** IVAMM.sol

Description The parameter is always zero.
Recommendation Consider removing the parameter.

Listing 246:

```
63 error IRSNotionalAmountSpecifiedMustBeNonZero(int256
   ↪ amountSpecified);
```

3.247 CVF-247

- **Severity** Minor
- **Status** Fixed
- **Category** Documentation
- **Source** IVAMM.sol

Description It is unclear what "isTrader" refers to here.
Recommendation Consider explaining.

Listing 247:

```
94 /// @dev lower tick of the liquidity provider (needs to be set
   ↪ if isTrader is false)

96 /// @dev upper tick of the liqudiity provider (needs to be set
   ↪ if isTrader is false)
```

3.248 CVF-248

- **Severity** Minor
- **Status** Fixed
- **Category** Documentation
- **Source** IVAMM.sol

Description The "feeProtocol" field is not a percentage, but rather a share. Not (1/x)% but just (1/x).
Recommendation Consider fixing the comment.

Listing 248:

```
103 // the current protocol fee as a percentage of the swap fee
    ↪ taken on withdrawal
// represented as an integer denominator (1/x)%
```

3.249 CVF-249

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** IVAMM.sol

Recommendation The type of the "_marginEngineAddress" should be "IMarginEngine".

Listing 249:

```
162 function initialize(address _marginEngineAddress, int24
    ↪ _tickSpacing)
```

3.250 CVF-250

- **Severity** Minor
- **Status** Fixed
- **Category** Documentation
- **Source** IVAMM.sol

Description The number format of the fee arguments is unclear.
Recommendation Consider documenting.

Listing 250:

```
226 function setFeeProtocol(uint8 feeProtocol) external;
```

3.251 CVF-251

- **Severity** Minor
- **Status** Fixed
- **Category** Documentation
- **Source** IVAMM.sol

Description The number format of the argument is unclear.
Recommendation Consider documenting.

Listing 251:

```
229 function setFee(uint256 _fee) external;
```

3.252 CVF-252

- **Severity** Minor
- **Status** Info
- **Category** Bad naming
- **Source** IVAMM.sol

Description Despite the name, this function returns information about a single tick.
Recommendation Consider renaming.
Client Comment Would prefer to keep the name in alignment with the naming pattern in Uni v3.

Listing 252:

```
284 function ticks(int24 tick)
```

3.253 CVF-253

- **Severity** Minor
- **Status** Info
- **Category** Bad naming
- **Source** IVAMM.sol

Description Despite the name, this function returns only a single word from the tick bitmap.
Recommendation Consider renaming.
Client Comment Would prefer to keep the name in alignment with the naming pattern in Uni v3.

Listing 253:

```
297 function tickBitmap(int16 wordPosition) external view returns (
    ↪ uint256);
```

3.254 CVF-254

- **Severity** Minor
- **Status** Fixed
- **Category** Bad naming
- **Source** IRateOracle.sol

Recommendation Events are usually named via nouns, such as "MinSecondsSinceLastUpdate" and "RateCardinalityNextIncrease".

Listing 254:

```
12 event MinSecondsSinceLastUpdateSet(uint256
    ↪ _minSecondsSinceLastUpdate);
event OracleBufferWrite(

26 event IncreaserateCardinalityNext(
```

3.255 CVF-255

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** IRateOracle.sol

Recommendation This parameter is redundant as its value could be derived from the previous event.

Listing 255:

```
27 uint16 observationCardinalityNextOld,
```

3.256 CVF-256

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** IRateOracle.sol

Recommendation The return type should be 'IERC20".

Listing 256:

```
43 function underlying() external view returns (address);
```

3.257 CVF-257

- **Severity** Minor
- **Status** Fixed
- **Category** Documentation
- **Source** IRateOracle.sol

Description The number format of the returned values is unclear.
Recommendation Consider documenting.

Listing 257:

```
49 function variableFactor(uint256 termStartTimestamp, uint256
     ↪ termEndTimestamp) external returns(uint256 result);

55 function variableFactorNoCache(uint256 termStartTimestamp,
     ↪ uint256 termEndTimestamp) external view returns(uint256
     ↪ result);

65     returns (uint256);

75     returns (uint256 apyFromTo);
```

3.258 CVF-258

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** IAaveRateOracle.sol

Recommendation The return type should be "IAaveV2LendingPool".

Listing 258:

```
11 function aaveLendingPool() external view returns (address);
```

3.259 CVF-259

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** IAToken.sol

Recommendation These commented out functions should be removed.

Listing 259:

```
63 // function transferFrom(
   //     address sender,
   //     address recipient,
   //     uint256 amount
   // ) external returns (bool);

69 // function transfer(address recipient, uint256 amount)
70 //   external
   //   returns (bool);

73 // function balanceOf(address account) external view returns (
      ↪ uint256);
```

3.260 CVF-260

- **Severity** Minor
- **Status** Fixed
- **Category** Documentation
- **Source** IAaveV2LendingPool.sol

Description The semantics of the returned value is unclear.
Recommendation Consider documenting.

Listing 260:

```
10 function getReserveNormalizedIncome(address underlyingAsset)
      ↪ external view returns (uint256);
```

3.261 CVF-261

- **Severity** Moderate
- **Status** Fixed
- **Category** Procedural
- **Source** IAaveV2LendingPool.sol

Recommendation The actual signature of the "initReserve" function in Aave V2 lending pool is: function initReserve(address reserve, address aTokenAddress, address stableDebtAddress, address variableDebtAddress, address interestRateStrategyAddress) external; https://github.com/aave/protocol-v2/blob/master/contracts/interfaces/ILendingPool.sol#L340-L346

Listing 261:

```
12 function initReserve(
       address asset,
       address aTokenAddress
   ) external;
```

3.262 CVF-262

- **Severity** Minor
- **Status** Fixed
- **Category** Documentation
- **Source** IAaveV2LendingPool.sol

Description The semantics of the returned value is unclear.
Recommendation Consider documenting.

Listing 262:

```
23 ) external returns (uint256);
```

3.263 CVF-263

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** TickBitmap.sol

Recommendation Bitwise “and” would be more efficient.
Client Comment Would prefer to keep the implementation aligned with that of Uni v3. These suggestions are really good though, however at this stage we'd minimise changes to uni "native" libraries.

Listing 263:

```
21 bitPos = uint8(int8(tick % 256));
```

3.264 CVF-264

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** TickBitmap.sol

Description Passing ticks already divided by tickSpacing would make code more efficient.
Client Comment Would prefer to keep the implementation aligned with that of Uni v3. These suggestions are really good though, however at this stage we'd minimise changes to uni "native" libraries.

Listing 264:

```
31 int24 tickSpacing

50 int24 tickSpacing,
```

3.265 CVF-265

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** TickBitmap.sol

Recommendation This could be calculated as (1 « (uint(bitPos) + 1)) - 1.
Client Comment Would prefer to keep the implementation aligned with that of Uni v3. These suggestions are really good though, however at this stage we'd minimise changes to uni "native" libraries.

Listing 265:

```
59 uint256 mask = (1 << bitPos) - 1 + (1 << bitPos);
```

3.266 CVF-266

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** TickBitmap.sol

Recommendation Instead of masking, you may do a shift: self[wordPos] « (255 - bitPos).
Client Comment Would prefer to keep the implementation aligned with that of Uni v3. These suggestions are really good though, however at this stage we'd minimise changes to uni "native" libraries.

Listing 266:

```
60 uint256 masked = self[wordPos] & mask;
```

3.267 CVF-267

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** TickBitmap.sol

Recommendation The multiplication by tickSpacing should be done once, after the ternary operator.
Client Comment Would prefer to keep the implementation aligned with that of Uni v3. These suggestions are really good though, however at this stage we'd minimise changes to uni "native" libraries.

Listing 267:

```
69      )) * tickSpacing
70  : (compressed - int24(uint24(bitPos))) * tickSpacing;

86      )) * tickSpacing
    : (compressed + 1 + int24(uint24(type(uint8).max - bitPos))) *
        tickSpacing;
```

3.268 CVF-268

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** TickBitmap.sol

Recommendation Instead of masking, you may do a shift: self[wordPos] » bitPos
Client Comment Would prefer to keep the implementation aligned with that of Uni v3. These suggestions are really good though, however at this stage we'd minimise changes to uni "native" libraries.

Listing 268:

```
76  uint256 masked = self[wordPos] & mask;
```

3.269 CVF-269

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** IERC20Minimal.sol

Description Names of the event parameters differ from those defined in ERC-20. Note, that unlike names of function arguments, names of event parameters are part of the public API of a contract.
Recommendation Consider using names as defined in ERC-20.
Client Comment This already appears to match https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC20/IERC20.sol and https://github.com/Uniswap/v3-core/blob/main/contracts/interfaces/IERC20Minimal.sol

Listing 269:

```
51 event Transfer(address indexed from, address indexed to, uint256
   ↪ value);

58     address indexed owner,
       address indexed spender,
60     uint256 value
```

3.270 CVF-270

- **Severity** Minor
- **Status** Fixed
- **Category** Unclear behavior
- **Source** IFCM.sol

Description This function should return the actual settling cash flow amount.

Listing 270:

```
31 function settleTrader() external;
```

3.271 CVF-271

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** IFCM.sol

Description Some argument names have underscore ('_') prefix, while other don't have it.
Recommendation Consider using a consistent naming policy.
Client Comment Removed underscore from account, but kept double underscore in initialize().

Listing 271:

```
37      address _account,

45  function initialize(address _vammAddress, address
        ↪ _marginEngineAddress)
```

3.272 CVF-272

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** IFCM.sol

Recommendation The types of the arguments should be "IVAMM" and "IMarginEngine" respectively.

Listing 272:

```
45  function initialize(address _vammAddress, address
        ↪ _marginEngineAddress)
```

3.273 CVF-273

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** IPositionStructs.sol

Description This interface contains only a structure. Solidity allows defining structures at the top level, i.e. outside any contracts, libraries, or interfaces.
Recommendation Consider defining the structures on the top level and removing this interface.
Client Comment Would prefer to keep the current implememntation for readability.

Listing 273:

```
5  interface IPositionStructs {
```

3.274 CVF-274

- **Severity** Minor
- **Status** Fixed
- **Category** Bad naming
- **Source** IFactory.sol

Recommendation Events are usually named via nouns, such as "IrsInstance" or "MasterFCM".

Listing 274:

```
8  event IrsInstanceDeployed(

19 event MasterFCMSet(
```

3.275 CVF-275

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** IFactory.sol

Recommendation The type of the parameter should be 'IERC20".

Listing 275:

```
9  address indexed underlyingToken,
```

3.276 CVF-276

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** IFactory.sol

Recommendation The type of this parameter should be "IRateOracle".

Listing 276:

```
10 address indexed rateOracle,
```

3.277 CVF-277

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** IFactory.sol

Recommendation The type of this parameter should be "IMarginEngine".

Listing 277:

```
14 address marginEngine,
```

3.278 CVF-278

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** IFactory.sol

Recommendation The type of this parameter should be "IVAMM".

Listing 278:

```
15 address vamm,
```

3.279 CVF-279

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** IFactory.sol

Recommendation The type of this parameter should be "IFCM".

Listing 279:

```
16 address fcm
```

3.280 CVF-280

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** IFactory.sol

Recommendation The type of these parameters should be 'IFCM".

Listing 280:

```
20 address masterFCMAddressOld,
   address masterFCMAddress,
```

3.281 CVF-281

- **Severity** Minor
- **Status** Fixed
- **Category** Unclear behavior
- **Source** IFactory.sol

Description This function should emit some event and this event should be declared in this interface.

Listing 281:

```
25 function setApproval(address intAddress, bool allowIntegration)
     ↪ external;
```

3.282 CVF-282

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** IFactory.sol

Recommendation The argument types should be "IFCM" and "IRateOracle" respectively.

Listing 282:

```
32 function setMasterFCM(address masterFCMAddress, address
   ↪ _rateOracle)
```

3.283 CVF-283

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** IFactory.sol

Recommendation The type of these arguments should be "IERC20".

Listing 283:

```
36 address _underlyingToken,

44 address _underlyingToken,

52 address _underlyingToken,

65 address _underlyingToken,
```

3.284 CVF-284

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** IFactory.sol

Recommendation The type of these arguments should be "IRateOracle".

Listing 284:

```
37 address _rateOracle,

45 address _rateOracle,

53 address _rateOracle,

66 address _rateOracle,
```

3.285 CVF-285

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** IFactory.sol

Recommendation The return type should be "IVAMM".

Listing 285:

```
41 ) external view returns (address);
```

3.286 CVF-286

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** IFactory.sol

Recommendation The return type should be "IMarginEngine".

Listing 286:

```
49 ) external view returns (address);
```

3.287 CVF-287

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** IFactory.sol

Recommendation The return type should be "IFCM".

Listing 287:

```
57 ) external view returns (address);
```

3.288 CVF-288

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** IFactory.sol

Recommendation The return type should be "IVAMM".

Listing 288:

```
59 function masterVAMM() external view returns (address);
```

3.289 CVF-289

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** IFactory.sol

Recommendation The return type should be "IMarginEngine".

Listing 289:

```
61 function masterMarginEngine() external view returns (address);
```

3.290 CVF-290

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** IFactory.sol

Recommendation The types of the returned values should be "IMarginEngine", "IVAMM", and "IFCM" respectively.

Listing 290:

```
73 address marginEngineProxy,
   address vammProxy,
   address fcmProxy
```

3.291 CVF-291

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** IFactory.sol

Recommendation The return type should be "IFCM".

Listing 291:

```
80 returns (address masterFCMAddress);
```