ABDK CONSULTING

SMART CONTRACT AUDIT

Rift

Rift protocol

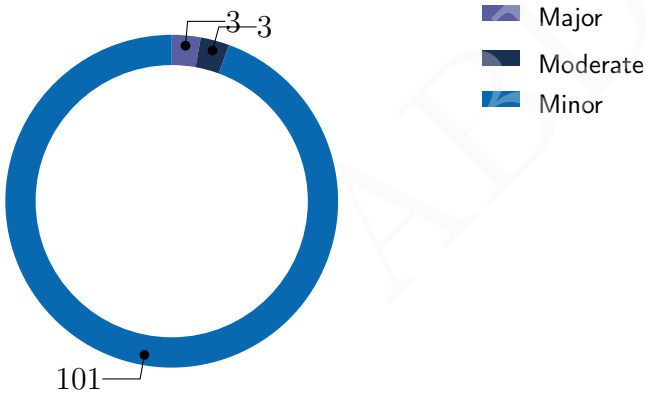
Solidity

abdk.consulting

SMART CONTRACT AUDIT CONCLUSION

by Mikhail Vladimirov and Dmitry Khovratovich
19th May 2022

We've been asked to review 16 files in a GitHub repository. We found 3 major, and a few less important issues. All identified major issues have been fixed.



Findings

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

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

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

ID	Severity	Category	Status
CVF-88	Minor	Unclear behavior	Info
CVF-89	Minor	Suboptimal	Fixed
CVF-90	Major	Flaw	Fixed
CVF-91	Minor	Bad naming	Info
CVF-92	Minor	Suboptimal	Fixed
CVF-93	Minor	Suboptimal	Info
CVF-94	Minor	Suboptimal	Info
CVF-95	Minor	Procedural	Info
CVF-96	Minor	Suboptimal	Info
CVF-97	Minor	Procedural	Fixed
CVF-98	Minor	Bad naming	Info
CVF-99	Minor	Suboptimal	Fixed
CVF-100	Minor	Documentation	Fixed
CVF-101	Minor	Suboptimal	Info
CVF-102	Minor	Bad datatype	Info
CVF-103	Major	Procedural	Fixed
CVF-104	Minor	Bad naming	Info
CVF-105	Minor	Suboptimal	Fixed
CVF-106	Minor	Suboptimal	Info
CVF-107	Minor	Unclear behavior	Info

Contents

1 Document properties **10**

2 Introduction **11**
2.1 About ABDK . . . 11
2.2 Disclaimer . . . 11
2.3 Methodology . . . 12

3 Detailed Results **13**
3.1 CVF-1 . . . 13
3.2 CVF-2 . . . 13
3.3 CVF-3 . . . 13
3.4 CVF-4 . . . 14
3.5 CVF-5 . . . 14
3.6 CVF-6 . . . 14
3.7 CVF-7 . . . 15
3.8 CVF-8 . . . 15
3.9 CVF-9 . . . 15
3.10 CVF-10 . . . 16
3.11 CVF-11 . . . 17
3.12 CVF-12 . . . 18
3.13 CVF-13 . . . 18
3.14 CVF-14 . . . 18
3.15 CVF-15 . . . 19
3.16 CVF-16 . . . 19
3.17 CVF-17 . . . 19
3.18 CVF-18 . . . 20
3.19 CVF-19 . . . 20
3.20 CVF-20 . . . 21
3.21 CVF-21 . . . 21
3.22 CVF-22 . . . 21
3.23 CVF-23 . . . 22
3.24 CVF-24 . . . 22
3.25 CVF-25 . . . 23
3.26 CVF-26 . . . 23
3.27 CVF-27 . . . 24
3.28 CVF-28 . . . 24
3.29 CVF-29 . . . 24
3.30 CVF-30 . . . 25
3.31 CVF-31 . . . 25
3.32 CVF-32 . . . 25
3.33 CVF-33 . . . 26
3.34 CVF-34 . . . 26
3.35 CVF-35 . . . 26
3.36 CVF-36 . . . 27
3.37 CVF-37 . . . 27

3.38 CVF-38 . . . 27
3.39 CVF-39 . . . 28
3.40 CVF-40 . . . 28
3.41 CVF-41 . . . 28
3.42 CVF-42 . . . 29
3.43 CVF-43 . . . 29
3.44 CVF-44 . . . 29
3.45 CVF-45 . . . 30
3.46 CVF-46 . . . 30
3.47 CVF-47 . . . 30
3.48 CVF-48 . . . 31
3.49 CVF-49 . . . 31
3.50 CVF-50 . . . 31
3.51 CVF-51 . . . 32
3.52 CVF-52 . . . 32
3.53 CVF-53 . . . 32
3.54 CVF-54 . . . 33
3.55 CVF-55 . . . 33
3.56 CVF-56 . . . 33
3.57 CVF-57 . . . 34
3.58 CVF-58 . . . 34
3.59 CVF-59 . . . 35
3.60 CVF-60 . . . 35
3.61 CVF-61 . . . 35
3.62 CVF-62 . . . 36
3.63 CVF-63 . . . 36
3.64 CVF-64 . . . 36
3.65 CVF-65 . . . 37
3.66 CVF-66 . . . 37
3.67 CVF-67 . . . 37
3.68 CVF-68 . . . 38
3.69 CVF-69 . . . 38
3.70 CVF-70 . . . 38
3.71 CVF-71 . . . 39
3.72 CVF-72 . . . 39
3.73 CVF-73 . . . 39
3.74 CVF-74 . . . 40
3.75 CVF-75 . . . 40
3.76 CVF-76 . . . 40
3.77 CVF-77 . . . 41
3.78 CVF-78 . . . 41
3.79 CVF-79 . . . 42
3.80 CVF-80 . . . 42
3.81 CVF-81 . . . 42
3.82 CVF-82 . . . 43
3.83 CVF-83 . . . 43

3.84 CVF-84 . . . 43
3.85 CVF-85 . . . 44
3.86 CVF-86 . . . 44
3.87 CVF-87 . . . 44
3.88 CVF-88 . . . 45
3.89 CVF-89 . . . 45
3.90 CVF-90 . . . 45
3.91 CVF-91 . . . 46
3.92 CVF-92 . . . 46
3.93 CVF-93 . . . 47
3.94 CVF-94 . . . 47
3.95 CVF-95 . . . 48
3.96 CVF-96 . . . 48
3.97 CVF-97 . . . 48
3.98 CVF-98 . . . 49
3.99 CVF-99 . . . 49
3.100 CVF-100 . . . 50
3.101 CVF-101 . . . 50
3.102 CVF-102 . . . 50
3.103 CVF-103 . . . 51
3.104 CVF-104 . . . 51
3.105 CVF-105 . . . 51
3.106 CVF-106 . . . 52
3.107 CVF-107 . . . 52

1 Document properties

Version

Version	Date	Author	Description
0.1	May 18, 2022	D. Khovratovich	Initial Draft
0.2	May 18, 2022	D. Khovratovich	Repository links are added
0.3	May 18, 2022	D. Khovratovich	Minor revision
1.0	May 19, 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:

- core/Core.sol
- core/CorePermissions.sol
- core/CoreStorage.sol
- core/ICore.sol
- core/ICorePermissions.sol
- refs/CoreReference.sol
- token/Rift.sol
- vaults/sushiswap/MasterChefV2Vault.sol
- vaults/sushiswap/MasterChefVault.sol
- vaults/sushiswap/MasterChefWithRefVault.sol
- vaults/sushiswap/SushiswapVaultStorage.sol
- vaults/uniswap/UniswapVault.sol
- vaults/uniswap/UniswapVaultStorage.sol
- vaults/IVault.sol
- vaults/Vault.sol
- vaults/VaultStorage.sol

The fixes were provided in a new commit.

2.1 About ABDK

ABDK Consulting, established in 2016, is a leading service provider in the space of blockchain development and audit. It has contributed to numerous blockchain projects, and co-authored some widely known blockchain primitives like Poseidon hash function. The ABDK Audit Team, led by Mikhail Vladimirov and Dmitry Khovratovich, has conducted over 40 audits of blockchain projects in Solidity, Rust, Circom, C++, JavaScript, and other languages.

2.2 Disclaimer

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

2.3 Methodology

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

- **General Code Assessment**. The code is reviewed for clarity, consistency, style, and for whether it follows code best practices applicable to the particular programming language used. We check indentation, naming convention, commented code blocks, code duplication, confusing names, confusing, irrelevant, or missing comments etc. At this phase we also understand overall code structure.

- **Entity Usage Analysis**. Usages of various entities defined in the code are analysed. This includes both: internal usages from other parts of the code as well as potential external usages. We check that entities are defined in proper places and that their visibility scopes and access levels are relevant. At this phase we understand overall system architecture and how different parts of the code are related to each other.

- **Access Control Analysis**. For those entities, that could be accessed externally, access control measures are analysed. We check that access control is relevant and is done properly. At this phase we understand user roles and permissions, as well as what assets the system ought to protect.

- **Code Logic Analysis**. The code logic of particular functions is analysed for correctness and efficiency. We check that code actually does what it is supposed to do, that algorithms are optimal and correct, and that proper data types are used. We also check that external libraries used in the code are up to date and relevant to the tasks they solve in the code. At this phase we also understand data structures used and the purposes they are used for.

3 Detailed Results

3.1 CVF-1

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

Recommendation Specifying a particular compiler version makes it harder to migrate to a newer version. Consider specifying as "^0.8.0". Also relevant for the next files: UniswapVaultStorage.sol, MasterChefWithRefVault.sol, MasterChefVault.sol, UniswapVault.sol, SushiswapVaultStorage.sol, MasterChefV2Vault.sol, Rift.sol, CoreReference.sol, CoreStorage.sol, Core.sol, CorePermissions.sol, IVault.sol, VaultStorage.sol, ICore.sol, ICorePermissions.sol.
Client Comment We do not want others to compile with other versions of solidity to maintain consistency with our deployments.

Listing 1:

```
2 pragma solidity 0.8.11;
```

3.2 CVF-2

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

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

Listing 2:

```
11 import "../external/IWrappy.sol";
```

3.3 CVF-3

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

Recommendation This value could be rendered as "1e27".
Client Comment Updated for readability.

Listing 3:

```
24 uint256 public constant RAY = 10**27;
```

3.4 CVF-4

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

Recommendation This value could be rendered as "1e4".
Client Comment Noted, we think this is more readable for smaller numbers.

Listing 4:

```
26 uint256 public constant DENOM = 10_000;
```

3.5 CVF-5

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

Recommendation The type of this argument should be "ICore".
Client Comment We avoid creating dependencies from interfaces.

Listing 5:

```
41 address coreAddress,
```

3.6 CVF-6

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

Recommendation The type of these arguments should be "IIERC20".
Client Comment We avoid creating dependencies from interfaces.

Listing 6:

```
43 address _token0,
   address _token1,

55 address _token0,
   address _token1,
```

3.7 CVF-7

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

Description There is no range check for this argument.
Recommendation Consider adding appropriate checks.
Client Comment This parameter is designed to be flexible depending on token0 depositor preferences. multiple vaults can exist that only toggle this parameter so users can choose their desired duration.

Listing 7:

```
54 uint256 _epochDuration,
```

3.8 CVF-8

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

Description These checks are redundant. It is anyway possible to pass a dead token address.
Client Comment Updated.

Listing 8:

```
60 require(_token0 != address(0), "ZERO_ADDRESS");
   require(_token1 != address(0), "ZERO_ADDRESS");
```

3.9 CVF-9

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

Description The "_amount" argument value is silently ignored in this branch.
Recommendation Consider explicitly requiring "_amount" to equal "msg.value".
Client Comment If we require these values are equal, the second branch would have a user depositing ETH, which we don't want. This is documented in the natspec comment.

Listing 9:

```
87 if (isNativeVault) {
       IWrappy(address(token0)).deposit{ value: msg.value }();
       _depositAccounting(token0Data, msg.value, TOKEN0);
```

3.10 CVF-10

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

Description This check makes it impossible to deposit already wrapped native tokens.
Recommendation Consider leaving such ability, for example, in case msg.value < _amount, consider transferring the missing part from the user in the form of already wrapped native tokens.
Client Comment This is desired. Additionally logic branches would be less elegant, and most users generally keep their native tokens unwrapped.

Listing 10:

```
87 if (isNativeVault) {
```

3.11 CVF-11

- **Severity** Minor
- **Status** Info
- **Category** Overflow/Underflow
- **Source** Vault.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 calculation overflows.
Recommendation Consider using the "muldiv" function as described here: https://2π.com/21/muldiv/index.html or some other approach that prevents phantom overflows.
Client Comment Mostly this is epochToRate * tokenAmount computation. If we make some very reasonable assumptions about max token amount (max decimals for the ERC20 is 24, token amount is < 10^50), this computation is valid. Additional refactoring and testing adds complexity for a very long tail possibility, and we are comfortable with this version. However, this is a useful comment and we will consider incorporating this into a future version of the protocol. Additional comment: added comment in the code.

Listing 11:

```
148     balance += (reqAmount * RAY) / conversionRate;

198     assetData.claimable[msg.sender] += (req.amount * assetData.
          ↪ epochToRate[req.epoch]) / RAY;

260         claimable += (withdrawAmountDay0 * assetData.epochToRate
              ↪ [withdrawEpoch]) / RAY;

366 return ((balanceDay0 * currentConversionRate) / RAY,
      ↪ pendingDeposits, claimable);

403 require(_token0.poolBalance >= (expectedPoolToken0 * (DENOM -
      ↪ POOL_ERR)) / DENOM, "UNEXPECTED_POOL_BALANCES");
    require(_token0.poolBalance <= (expectedPoolToken0 * (DENOM +
      ↪ POOL_ERR)) / DENOM, "UNEXPECTED_POOL_BALANCES");
    require(_token1.poolBalance >= (expectedPoolToken1 * (DENOM -
      ↪ POOL_ERR)) / DENOM, "UNEXPECTED_POOL_BALANCES");
    require(_token1.poolBalance <= (expectedPoolToken1 * (DENOM +
      ↪ POOL_ERR)) / DENOM, "UNEXPECTED_POOL_BALANCES");

419 uint256 token0Floor = _token0Data.reserves + (_token0Data.active
      ↪  * token0FloorNum) / DENOM;
420 uint256 token1Floor = _token1Data.reserves + (_token1Data.active
      ↪  * token1FloorNum) / DENOM;

493     _token0.available -= ((_token0.available - _token0.original)
          ↪  * core.protocolFee()) / core.MAX_FEE();
```

(... 496, 503, 506, 509, 512, 671)

3.12 CVF-12

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

Description The expression "assetData.withdrawRequests[msg.sender]" is calculated several times.
Recommendation ;
Client Comment Updated.

Listing 12:

```
194 Request memory req = assetData.withdrawRequests[msg.sender];

210 assetData.withdrawRequests[msg.sender].amount =
    ↪ _withdrawAmountDay0 + req.amount;

212     assetData.withdrawRequests[msg.sender].epoch = currEpoch;
```

3.13 CVF-13

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

Description The function spends some gas calculate each of these three values. In case the caller needs only some of them, part of the used gas will be wasted.
Recommendation Consider providing an option to calculate only the needed values.
Client Comment This function should only be called off-chain, so gas is not a concern.

Listing 13:

```
319 uint256 _deposited,
320 uint256 _pendingDeposit,
    uint256 _claimable
```

3.14 CVF-14

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

Description This variable is redundant.
Recommendation Just use "_pendingDeposit" instead.
Client Comment Updated.

Listing 14:

```
327 uint256 pendingDeposits;
```

3.15 CVF-15

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

Description This variable is redundant as its value is used only once.
Recommendation Consider using an expression instead.
Client Comment Updated.

Listing 15:

```
390 uint256 timePassed = block.timestamp - lastEpochStart;
```

3.16 CVF-16

- **Severity** Moderate
- **Status** Fixed
- **Category** Suboptimal
- **Source** Vault.sol

Description This check doesn't make much sense as the strategist may change the epoch duration to an arbitrary value at any time.
Recommendation Consider either removing this check or changing the logic to make it reasonable. For example, make epoch duration changes not to affect the current epoch.
Client Comment The strategist is a trusted party and setting a different epoch duration should only be used in dire situations. we will change the setEpochDuration to be restriced to be used only when the contract is paused, meaning that the strategist cannot arbitrarily change it.

Listing 16:

```
391 require(timePassed >= epochDuration, "EPOCH_DURATION_UNMET");
```

3.17 CVF-17

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

Description This value id already calculated and is available as "token0Deficit".
Client Comment Updated.

Listing 17:

```
433 token0Floor - _token0.available,
```

3.18 CVF-18

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

Recommendation The brackets are redundant.
Client Comment Updated.

Listing 18:

```
443 : (

447 );
```

3.19 CVF-19

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

Description This code appear several times.
Recommendation Consider refactoring to have this code only once, after the conditional statement.
Client Comment It is dependent on which token was swapped, so should not be placed outside the conditional statement.

Listing 19:

```
449 (uint256 amountOut, uint256 amountConsumed) = swap(token1,
    ↪ token0, swapAmount);
450 _token0.available += amountOut;
    _token1.available -= amountConsumed;

456 (uint256 amountOut, uint256 amountConsumed) = swap(token1,
    ↪ token0, _token1.available - token1Ceiling);
    _token0.available += amountOut;
    _token1.available -= amountConsumed;

476     (uint256 amountOut, uint256 amountConsumed) = swap(token0,
        ↪ token1, token0NeededToSwap);
        _token0.available -= amountConsumed;
        _token1.available += amountOut;

481     (uint256 amountOut, uint256 amountConsumed) = swap(token0,
        ↪ token1, _token0.available - token0Floor);
        _token0.available -= amountConsumed;
        _token1.available += amountOut;
```

3.20 CVF-20

- **Severity** Minor
- **Status** Fixed
- **Category**
- **Source** Vault.sol

Description This value was already calculated and is available as "token1Deficit".
Client Comment Updated.

Listing 20:

```
468 token1Ceiling - _token1.available,
```

3.21 CVF-21

- **Severity** Minor
- **Status** Info
- **Category**
- **Source** Vault.sol

Description The value "_token0.available - token0Floor" is used twice.
Recommendation Consider calculating once and reusing.
Client Comment We believe this is the cleaner implementation as calculating it once would require rearranging the first equation and thus, more checks to ensure that math isn't underflowing.

Listing 21:

```
474 if (token0Floor + token0NeededToSwap < _token0.available) {

481     (uint256 amountOut, uint256 amountConsumed) = swap(token0,
          ↪ token1, _token0.available - token0Floor);
```

3.22 CVF-22

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

Recommendation Simple assignments of zero values would be more readable.
Client Comment Updated.

Listing 22:

```
519 delete token0Data.depositRequestsTotal;
520 delete token0Data.withdrawRequestsTotal;
    delete token1Data.depositRequestsTotal;
    delete token1Data.withdrawRequestsTotal;
```

3.23 CVF-23

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

Recommendation This code could be simplified using a struct literal.
Client Comment Updated.

Listing 23:

```
536 AssetDataStatics memory _assetData;
    _assetData.reserves = assetData.reserves;
    _assetData.active = assetData.active;
    _assetData.depositRequestsTotal = assetData.depositRequestsTotal
      ↪ ;
540 _assetData.withdrawRequestsTotal = assetData.
      ↪ withdrawRequestsTotal;
    return _assetData;
```

3.24 CVF-24

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

Description The threshold of 1 gwei could be too large for tokens with little decimals or with a very high price.
Recommendation Consider making this threshold configurable per vault or consider removing the threshold completely.
Client Comment Will fix. This check is required, because the uniswap add liquidity function will revert if 0 liquidity is minted, which can happen if this check is removed and prices are extreme. however, we will reduce it to 1000 (instead of 1 gwei) so that tokens with 6 decimals are supported. we will note that this may be an issue for tokens with very high price or very few decimals.

Listing 24:

```
575 if ((availableToken0 < (1 gwei)) || (availableToken1 < (1 gwei))
      ↪ ) return (0, 0);
```

3.25 CVF-25

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

Description Temp "gwei" here is confusing, as it is applied here not to ether amounts but to amounts of arbitrary tokens.
Client Comment Updated.

Listing 25:

```
575 if ((availableToken0 < (1 gwei)) || (availableToken1 < (1 gwei))
    ↪ ) return (0, 0);
```

3.26 CVF-26

- **Severity** Moderate
- **Status** Info
- **Category** Suboptimal
- **Source** Vault.sol

Description This function allows rescuing ether and ERC-20 tokens, but not other assets such as NFT or non-tokenized DeFi liquidity positions.
Recommendation Consider allowing the guardian to invoke any external contracts with any arguments on behalf of this contract to make funds rescuing more universal.
Client Comment There is a ‘unstake‘ function which the guardian can call to unstake a position that isn't represented by token. Other types of vaults may be supported in future versions of the protocol.

Listing 26:

```
597 function rescueTokens(address[] calldata tokens, uint256[]
    ↪ calldata amounts)
```

3.27 CVF-27

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

Description This function allows the guardian to steal all the tokens from the contract.
Recommendation Consider protecting the users from malicious guardian in some way. For example, forbid the guardian to withdraw tokens claimable by the users, or allow teh guardian to withdraw token only after certain period of time passed since the contract was paused, allowing the users to withdraw their tokens during this period.
Client Comment Noted. This is an acceptable trade-off, since the guardian can only rescue tokens if the pauser has paused the vaults. These roles are delegated separately.

Listing 27:

```
597 function rescueTokens(address[] calldata tokens, uint256[]
    ↪ calldata amounts)
```

3.28 CVF-28

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

Recommendation Consider making the transfer only if amount >0.
Client Comment Don't need to make an extra if branch, the user bears the gas cost if this happens.

Listing 28:

```
610 (bool success, ) = msg.sender.call{ value: amount }("");
```

3.29 CVF-29

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

Description These events are emitted even if nothing actually changed.
Client Comment We reduce gas costs and trust that the strategist does not spam the contract with null events.

Listing 29:

```
714 emit Token0FloorUpdated(_token0FloorNum, msg.sender);

722 emit Token1FloorUpdated(_token1FloorNum, msg.sender);
```

3.30 CVF-30

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

Description There is no range check for the argument.
Recommendation Consider adding an appropriate check.
Client Comment This parameter is designed to be flexible depending on token1 depositor preferences. Multiple vaults can exist that only toggle this parameter so users can choose their desired duration.

Listing 30:

```
725 function setEpochDuration(uint256 _epochDuration) external
    ↪ override onlyStrategist {
```

3.31 CVF-31

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

Description This function should emit an event.
Client Comment Updated.

Listing 31:

```
725 function setEpochDuration(uint256 _epochDuration) external
    ↪ override onlyStrategist {
```

3.32 CVF-32

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

Recommendation It is a good practice to put a comment into an empty block to explain why the block is empty.
Client Comment Updated.

Listing 32:

```
741 receive() external payable {}
```

3.33 CVF-33

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

Recommendation This contract should be moved to a separate file named "UniswapVaultStorageUnpadded.sol".
Client Comment We believe it is more intuitive for a user who would be upgrading storage to see this directly in the same file, so updating the __gap variable does not slip their mind.

Listing 33:

```
6 abstract contract UniswapVaultStorageUnpadded {
```

3.34 CVF-34

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

Recommendation The type of this field should be "IUniswapV2Factory".
Client Comment We avoid creating dependencies from interfaces.

Listing 34:

```
8 address public factory;
```

3.35 CVF-35

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

Recommendation The type of this field should be "IUniswapV2Router02".
Client Comment We avoid creating dependencies from interfaces.

Listing 35:

```
11 address public router;
```

3.36 CVF-36

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

Recommendation The type of this field should be "IUniswapV2Pair".
Client Comment We avoid creating dependencies from interfaces.

Listing 36:

```
14 address public pair;
```

3.37 CVF-37

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

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

Listing 37:

```
5 import { IMasterChefWithRef } from "../../external/sushiswap/
    ↪ IMasterChef.sol";
```

3.38 CVF-38

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

Description Approving every time is suboptimal.
Recommendation It would be more efficient to approve large amount once.
Client Comment We increase and decrease allowance each time for increased security.

Listing 38:

```
18 IERC20Upgradeable(pair).safeIncreaseAllowance(rewarder,
    ↪ lpTokenBalance);
```

3.39 CVF-39

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

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

Listing 39:

```
5 import { IMasterChef } from "../../external/sushiswap/
    ↪ IMasterChef.sol";
```

3.40 CVF-40

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

Recommendation The type of these arguments should be "ICore".
Client Comment We avoid creating dependencies from interfaces.

Listing 40:

```
15 address coreAddress,

43 address coreAddress,
```

3.41 CVF-41

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

Recommendation The type of these arguments should be "IERC20".
Client Comment We avoid creating dependencies from interfaces.

Listing 41:

```
17 address _token0,
   address _token1,

45 address _token0,
   address _token1,
```

3.42 CVF-42

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

Recommendation The type of these arguments should be "IUniswapV2Factory".
Client Comment We avoid creating dependencies from interfaces.

Listing 42:

```
21 address _sushiswapFactory,

49 address _sushiswapFactory,
```

3.43 CVF-43

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

Recommendation The type of these arguments should be "IUniswapV2Router02".
Client Comment We avoid creating dependencies from interfaces.

Listing 43:

```
22 address _sushiswapRouter,

50 address _sushiswapRouter,
```

3.44 CVF-44

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

Recommendation The type of these arguments should be "IERC20".
Client Comment We avoid creating dependencies from interfaces.

Listing 44:

```
23 address _sushi,

51 address _sushi,
```

3.45 CVF-45

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

Recommendation The type of these arguments should be "IMasterChef".
Client Comment We avoid creating dependencies from interfaces.

Listing 45:

```
24 address _masterChef,

52 address _masterChef,
```

3.46 CVF-46

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

Description These checks are redundant as it is anyway possible to pass a dead address as an argument.
Client Comment Updated.

Listing 46:

```
73 require(_sushi != address(0), "ZERO_ADDRESS");
   require(_masterChef != address(0), "ZERO_ADDRESS");
```

3.47 CVF-47

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

Description Approving every time is suboptimal.
Recommendation It would be more efficient to approve large amount once.
Client Comment We increase and decrease allowance each time for increased security.

Listing 47:

```
92      sushi.safeIncreaseAllowance(router, sushiBalance);

108 IERC20Upgradeable(pair).safeIncreaseAllowance(rewarder,
    ↪ lpTokenBalance);
```

3.48 CVF-48

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

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

Listing 48:

```
5 import "../../../lib/v2-periphery/contracts/interfaces/
    ↪ IUniswapV2Router02.sol";
import "../../../lib/v2-core/contracts/interfaces/
    ↪ IUniswapV2Factory.sol";
import "../../../lib/v2-core/contracts/interfaces/IUniswapV2Pair
    ↪ .sol";
```

3.49 CVF-49

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

Recommendation The type of these arguments should be "ICore".
Client Comment We avoid creating dependencies from interfaces.

Listing 49:

```
18 address coreAddress,

40 address coreAddress,
```

3.50 CVF-50

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

Recommendation The type of these arguments should be "IERC20".
Client Comment We avoid creating dependencies from interfaces.

Listing 50:

```
20 address _token0,
   address _token1,

42 address _token0,
   address _token1,
```

3.51 CVF-51

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

Recommendation The type of these arguments should be "IUniswapV2Factory".
Client Comment We avoid creating dependencies from interfaces.

Listing 51:

```
24 address _uniswapFactory,

46 address _uniswapFactory,
```

3.52 CVF-52

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

Recommendation The type of these arguments should be "IUniswapV2Router02".
Client Comment We avoid creating dependencies from interfaces.

Listing 52:

```
25 address _uniswapRouter

47 address _uniswapRouter
```

3.53 CVF-53

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

Recommendation The types of the arguments should be "IUniswapV2Factory" and "IUniswapV2Router02" respectively.
Client Comment We avoid creating dependencies from interfaces.

Listing 53:

```
53 function __UniswapVault_init_unchained(address _uniswapFactory,
   ↪ address _uniswapRouter) internal onlyInitializing {
```

3.54 CVF-54

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

Description These checks are redundant, as it is anyway possible to pass a dead address as an argument.
Client Comment Updated.

Listing 54:

```
54 require(_uniswapFactory != address(0), "ZERO_ADDRESS");
   require(_uniswapRouter != address(0), "ZERO_ADDRESS");
```

3.55 CVF-55

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

Description The semantics of the returned values is unclear.
Recommendation Consider giving descriptive names to the returned values and/or adding a documentation comment.
Client Comment Updated.

Listing 55:

```
66 function getPoolBalances() internal view virtual override
   ↪ returns (uint256, uint256) {
```

3.56 CVF-56

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

Description Querying the token0 from the pair each time is suboptimal.
Recommendation Consider determining the token order once at initialization and storing the result in a storage variable.
Client Comment Noted, refactoring too major.

Listing 56:

```
68 return IUniswapV2Pair(pair).token0() == address(token0) ? (
   ↪ reservesA, reservesB) : (reservesB, reservesA);
```

3.57 CVF-57

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

Description Approving every time is suboptimal.
Recommendation Consider approving large amount once during initialization.
Client Comment We increase and decrease allowance each time for increased security.

Listing 57:

```
90  IERC20Upgradeable(pair).safeIncreaseAllowance(router,
        ↪ lpTokenBalance);

114 token0.safeIncreaseAllowance(router, availableToken0);
    token1.safeIncreaseAllowance(router, availableToken1);

153 tokenIn.safeIncreaseAllowance(router, amountIn);
```

3.58 CVF-58

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

Recommendation It would be simpler to just set allowance to zero.
Client Comment Updated.

Listing 58:

```
130 token0.safeDecreaseAllowance(router, availableToken0 −
        ↪ token0Deposited);

133 token1.safeDecreaseAllowance(router, availableToken1 −
        ↪ token1Deposited);
```

3.59 CVF-59

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

Recommendation It is a good practice to put a comment into an empty block to explain why the block is empty.
Client Comment We have a comment above it to explain why.

Listing 59:

```
138 function _unstakeLiquidity() internal virtual override {}

141 function _stakeLiquidity() internal virtual override {}
```

3.60 CVF-60

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

Recommendation This contract should be moved to a separate file named "SushiswapVaultStorageUnpadded.sol".
Client Comment We believe it is more intuitive for a user who would be upgrading storage to see this directly in the same file, so updating the __gap variable does not slip their mind.

Listing 60:

```
13 abstract contract SushiswapVaultStorageUnpadded is
   ↪ UniswapVaultStorage {
```

3.61 CVF-61

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

Recommendation The type of this variable should be "IMasterChefV2".
Client Comment It may be either a IMasterChefV2 or a IMasterChef, depending on the contract inheriting it, so it must remain an address.

Listing 61:

```
15 address public rewarder;
```

3.62 CVF-62

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** MasterChefV2Vault.sol

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

Listing 62:

```
5 import "../../external/sushiswap/IMasterChefV2.sol";
```

3.63 CVF-63

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** MasterChefV2Vault.sol

Recommendation The type of these arguments should be "ICore".
Client Comment We avoid creating dependencies from interfaces.

Listing 63:

```
15 address coreAddress,

43 address coreAddress,
```

3.64 CVF-64

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** MasterChefV2Vault.sol

Recommendation The type of these arguments should be "IERC20".
Client Comment We avoid creating dependencies from interfaces.

Listing 64:

```
17 address _token0,
   address _token1,

45 address _token0,
   address _token1,
```

3.65 CVF-65

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** MasterChefV2Vault.sol

Recommendation The type of these arguments should be "IUniswapV2Factory".
Client Comment We avoid creating dependencies from interfaces.

Listing 65:

```
21 address _sushiswapFactory,

49 address _sushiswapFactory,
```

3.66 CVF-66

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** MasterChefV2Vault.sol

Recommendation The type of these arguments should be "IUniswapV2Router02".
Client Comment We avoid creating dependencies from interfaces.

Listing 66:

```
22 address _sushiswapRouter,

50 address _sushiswapRouter,
```

3.67 CVF-67

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** MasterChefV2Vault.sol

Recommendation The type of these arguments should be "IERC20".
Client Comment We avoid creating dependencies from interfaces.

Listing 67:

```
23 address _sushi,

51 address _sushi,

69 address _sushi,
```

3.68 CVF-68

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** MasterChefV2Vault.sol

Recommendation The type of these arguments should be"IMasterChefV2".
Client Comment We avoid creating dependencies from interfaces.

Listing 68:

```
24 address _masterChefV2,

52 address _masterChefV2,

70 address _masterChefV2,
```

3.69 CVF-69

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** MasterChefV2Vault.sol

Description These checks are redundant as it is anyway possible to pass a dead address as an argument.
Client Comment Updated.

Listing 69:

```
73 require(_sushi != address(0), "ZERO_ADDRESS");
   require(_masterChefV2 != address(0), "ZERO_ADDRESS");
```

3.70 CVF-70

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** MasterChefV2Vault.sol

Description Approving every time is suboptimal.
Recommendation It would be more efficient to approve large amount once.
Client Comment We increase and decrease allowance each time for increased security.

Listing 70:

```
91      sushi.safeIncreaseAllowance(router, sushiBalance);

107 IERC20Upgradeable(pair).safeIncreaseAllowance(rewarder,
    ↪ lpTokenBalance);
```

3.71 CVF-71

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

Description Unlike other contract in this project, this contract doesn't use the role framework from OpenZeppelin, but implements its own role-based approach.
Recommendation Consider using the OpenZeppelin's role framework here for consistency.
Client Comment Updated.

Listing 71:

```
8 mapping(address => bool) public isBurner;
  mapping(address => bool) public isMinter;
```

3.72 CVF-72

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

Description Probably these functions should emit an event.
Client Comment Updated.

Listing 72:

```
31 function addMinter(address _newMinter) public onlyOwner {

36 function addBurner(address _newBurner) public onlyOwner {

41 function revokeMinter(address _oldMinter) public onlyOwner {
```

3.73 CVF-73

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

Description These checks are redundant. They don't save any gas.
Client Comment Updated.

Listing 73:

```
32 require(!isMinter[_newMinter], "ALREADY_MINTER");

37 require(!isBurner[_newBurner], "ALREADY_BURNER");

42 require(isMinter[_oldMinter], "NOT_MINTER");

47 require(isBurner[_oldBurner], "NOT_BURNER");
```

3.74 CVF-74

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

Recommendation It is a good practice to put a comment into an empty block to explain why the block is empty.
Client Comment We have a comment above it to explain why.

Listing 74:

```
19 constructor() initializer {}
```

3.75 CVF-75

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

Recommendation Events are usually named via nouns, such as "Pause", "Unpause".
Client Comment We use openzeppelin's standard here.

Listing 75:

```
22 event Paused(address indexed account);

25 event Unpaused(address indexed account);
```

3.76 CVF-76

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

Description The parameters are redundant, as their values could in most cases be extracted from the transaction data.
Client Comment Updated.

Listing 76:

```
22 event Paused(address indexed account);

25 event Unpaused(address indexed account);
```

3.77 CVF-77

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

Description These events and functions are very similar to those defined in the "ICore" interface.
Recommendation Consider extracting a separate "Pausable" interface and probably an abstract implementation or it.
Client Comment Noted. The implementation of ‘paused‘ is unique each time, so we keep it separate.

Listing 77:

```
22 event Paused(address indexed account);

25 event Unpaused(address indexed account);

68 function paused() public view returns (bool) {

72 function pause() external onlyPauser whenNotPaused {

77 function unpause() external onlyPauser whenPaused {
```

3.78 CVF-78

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

Recommendation The argument type should be "ICore".
Client Comment We avoid creating dependencies from interfaces.

Listing 78:

```
27 function __CoreReference_init(address coreAddress) internal
   ↪ onlyInitializing {

31 function __CoreReference_init_unchained(address coreAddress)
   ↪ internal onlyInitializing {
```

3.79 CVF-79

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

Recommendation This check is redundant, as it is anyway possible to pass a dead core address.
Client Comment Updated.

Listing 79:

```
32 require(coreAddress != address(0), "ZERO_ADDRESS");
```

3.80 CVF-80

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

Description These events are emitted even when nothing actually changed.
Client Comment Noted. We can expect the pauser to not spam the contract with null events.

Listing 80:

```
74 emit Paused(msg.sender);

79 emit Unpaused(msg.sender);
```

3.81 CVF-81

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

Description The number format of this variable is unclear.
Recommendation Consider documenting.
Client Comment Added a comment.

Listing 81:

```
13 uint256 public override protocolFee;
```

3.82 CVF-82

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

Description The number format of this constant is unclear.
Recommendation Consider documenting.
Client Comment Added a comment.

Listing 82:

```
15 uint256 public constant override MAX_FEE = 10000;
```

3.83 CVF-83

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

Recommendation It is a good practice to put a comment into an empty block to explain why the block is empty.
Client Comment We have a comment above it to explain why.

Listing 83:

```
24 constructor() initializer {}
```

3.84 CVF-84

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

Description The number format of this value is unclear.
Recommendation Consider documenting.
Client Comment It is documented in the natspec comment for the function.

Listing 84:

```
36 uint256 _protocolFee,
```

3.85 CVF-85

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

Recommendation The type of this argument could be more specific.
Client Comment We avoid creating dependencies from interfaces.

Listing 85:

```
38 address _wrappedNative,

50 address _wrappedNative,

63 address _wrappedNative
```

3.86 CVF-86

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

Recommendation These checks are redundant, it is anyway possible to pass a dead address as an argument.
Client Comment Updated.

Listing 86:

```
66 require(_feeTo != address(0), "ZERO_ADDRESS");
   require(_wrappedNative != address(0), "ZERO_ADDRESS");
```

3.87 CVF-87

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

Recommendation This check is redundant. It is anyway possible to add a dead vault address.
Client Comment Updated.

Listing 87:

```
80 require(vaults[i] != address(0), "ZERO_ADDRESS");
```

3.88 CVF-88

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

Description These events are emitted even if nothing actually changed.
Client Comment We assume those who can functions that emit these events are benevolent.

Listing 88:

```
104 emit ProtocolFeeUpdated(_protocolFee);

112 emit FeeToUpdated(_feeTo);

144 emit Paused(msg.sender);

151 emit Unpaused(msg.sender);
```

3.89 CVF-89

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

Recommendation These checks are redundant. It is anyway possible to pass a dead address as an argument.
Client Comment Updated.

Listing 89:

```
39 require(guardian != address(0), "ZERO_ADDRESS");
40 require(governor != address(0), "ZERO_ADDRESS");
   require(pauser != address(0), "ZERO_ADDRESS");
   require(strategist != address(0), "ZERO_ADDRESS");
```

3.90 CVF-90

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

Description This protection is not reliable as the remaining governor may be a dead address.
Recommendation It would be more reliable to forbid a governor to revoke the governor role from himself.
Client Comment Updated and removed dependency on costly AccessControlEnumerable.

Listing 90:

```
74 require(role != GOVERN_ROLE || getRoleMemberCount(role) >= 1, "
     ↪ LAST_GOVERNOR");
```

3.91 CVF-91

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

Recommendation Events are usually named via nouns, such as "Epoch", "Deposit", "Withdrawal", etc.
Client Comment Noted. We use openzeppelin's standard here.

Listing 91:

```
 9 event NextEpochStarted(uint256 indexed newEpoch, address indexed
     ↪  initiator, uint256 startTime);

16 event DepositScheduled(bytes32 indexed assetCode, address
     ↪ indexed user, uint256 amount, uint256 indexed epoch);

23 event WithdrawScheduled(bytes32 indexed assetCode, address
     ↪ indexed user, uint256 amountDay0, uint256 indexed epoch);

29 event AssetsClaimed(bytes32 indexed assetCode, address indexed
     ↪ user, uint256 amount);

33 event FundsRescued(address indexed guardian);

38 event Token0FloorUpdated(uint256 indexed newFloor, address
     ↪ indexed admin);

43 event Token1FloorUpdated(uint256 indexed newFloor, address
     ↪ indexed admin);
```

3.92 CVF-92

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

Recommendation The "newFloor" parameters shouldn't be indexed.
Client Comment Updated.

Listing 92:

```
38 event Token0FloorUpdated(uint256 indexed newFloor, address
     ↪ indexed admin);

43 event Token1FloorUpdated(uint256 indexed newFloor, address
     ↪ indexed admin);
```

3.93 CVF-93

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

Recommendation It would be more efficient to pass a single array of structs with two fields instead of two parallel arrays. Such approach would also make a length check unnecessary.
Client Comment Noted, in the unlikely scenario this function needs to be called, the guardian should put the necessary time into ensure inputs are valid.

Listing 93:

```
79 function rescueTokens(address[] calldata tokens, uint256[]
     ↪ calldata amounts) external;
```

3.94 CVF-94

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

Description In case somebody needs several of these values at once, he will have to call the contract several times.
Recommendation Consider providing a way to query several values in one call.
Client Comment Noted.

Listing 94:

```
87 function token0ValueLocked() external view returns (uint256);

89 function token1ValueLocked() external view returns (uint256);

91 function token0BalanceDay0(address user) external view returns (
     ↪ uint256);

93 function epochToken0Rate(uint256 _epoch) external view returns
     ↪  (uint256);

95 function token0WithdrawRequests(address user) external view
     ↪ returns (uint256);

97 function token1BalanceDay0(address user) external view returns (
     ↪ uint256);

99 function epochToken1Rate(uint256 _epoch) external view returns
     ↪  (uint256);

101 function token1WithdrawRequests(address user) external view
     ↪ returns (uint256);
```

3.95 CVF-95

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

Recommendation This contract should be moved to a separate file named "VaultStorage-Unpadded.sol".
Client Comment We believe it is more intuitive for a user who would be upgrading storage to see this directly in the same file, so updating the __gap variable does not slip their mind.

Listing 95:

```
10 abstract contract VaultStorageUnpadded {
```

3.96 CVF-96

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

Recommendation It would be more efficient to merge these mappings into a single mapping whose keys are user and values are structs encapsulating the values of the original mappings.
Client Comment Noted, refactoring too major.

Listing 96:

```
44 mapping(address => uint256) balanceDay0;
   mapping(address => uint256) claimable;

47 mapping(address => Request) depositRequests;
   mapping(address => Request) withdrawRequests;
```

3.97 CVF-97

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

Description The "DENOM" constant is defined in another file.
Recommendation Consider revering to it as "Vault.DENOM" to make it easier for a reader to find it.
Client Comment Updated.

Listing 97:

```
71 /// @notice minimum return for TOKEN0 (out of `DENOM`) as long
      ↪ as TOKEN1 is above its minimum return

73 /// @notice minimum return for TOKEN1 (out of `DENOM`)
```

3.98 CVF-98

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

Recommendation Events are usually named via nouns, such as "ProtocolFee", "FeeTo", "Pause", etc.
Client Comment Noted. We used open zeppelin's standard here.

Listing 98:

```
12 event ProtocolFeeUpdated(uint256 indexed protocolFee);

15 event FeeToUpdated(address indexed feeTo);

18 event Paused(address indexed account);

21 event Unpaused(address indexed account);

24 event VaultRegistered(address indexed vault, address indexed
   ↪ admin);

27 event VaultRemoved(address indexed vault, address indexed admin)
   ↪ ;
```

3.99 CVF-99

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

Recommendation The parameter shouldn't be indexed.
Client Comment Updated.

Listing 99:

```
12 event ProtocolFeeUpdated(uint256 indexed protocolFee);
```

3.100 CVF-100

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

Description The number format of fee values is unclear.
Recommendation Consider documenting.
Client Comment Updated.

Listing 100:

```
12 event ProtocolFeeUpdated(uint256 indexed protocolFee);

31 function MAX_FEE() external view returns (uint256);

35 function protocolFee() external view returns (uint256);

45 function setProtocolFee(uint256 _protocolFee) external;
```

3.101 CVF-101

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

Description The "admin" parameters are redundant.
Recommendation Their values in most cases could be derived from the transaction data.
Client Comment Noted. We used open zeppelin's standard here.

Listing 101:

```
24 event VaultRegistered(address indexed vault, address indexed
   ↪ admin);

27 event VaultRemoved(address indexed vault, address indexed admin)
   ↪ ;
```

3.102 CVF-102

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

Recommendation The return type could be more specific.
Client Comment We avoid creating dependencies from interfaces.

Listing 102:

```
37 function wrappedNative() external view returns (address);
```

3.103 CVF-103

- **Severity** Major
- **Status** Fixed
- **Category** Procedural
- **Source** ICore.sol

Description This function doesn't scale. When there are too many registered vaults, this function may exceed the block gas limit.
Recommendation Consider implementing an ability to query the registered values by index.
Client Comment This functionality was removed.

Listing 103:

```
51 function getRegisteredVaults() external view returns (address[]
   ↪ memory);
```

3.104 CVF-104

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

Recommendation Events are usually named via nouns.
Client Comment Noted. We use open zeppelin's standar here.

Listing 104:

```
12 event WhitelistDisabled(address indexed admin);

15 event WhitelistEnabled(address indexed admin);
```

3.105 CVF-105

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

Recommendation The event parameters are redundant. Their values could in most cases be extracted from the transaction data.
Client Comment Updated.

Listing 105:

```
12 event WhitelistDisabled(address indexed admin);

15 event WhitelistEnabled(address indexed admin);
```

3.106 CVF-106

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

Recommendation This function should emit some event and this event should be declared in this interface.
Client Comment The function emits RoleGranted when it calls _grantRole.

Listing 106:

```
21 function whitelistAll(address[] memory addresses) external;
```

3.107 CVF-107

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

Description There is no function to remove an address from the whitelist.
Client Comment grantRole and revokeRole allow are inherited external functions that allow the governor to add and revoke roles.

Listing 107:

```
21 function whitelistAll(address[] memory addresses) external;
```