ABDK CONSULTING

SMART CONTRACT AUDIT

SENSE. Part I

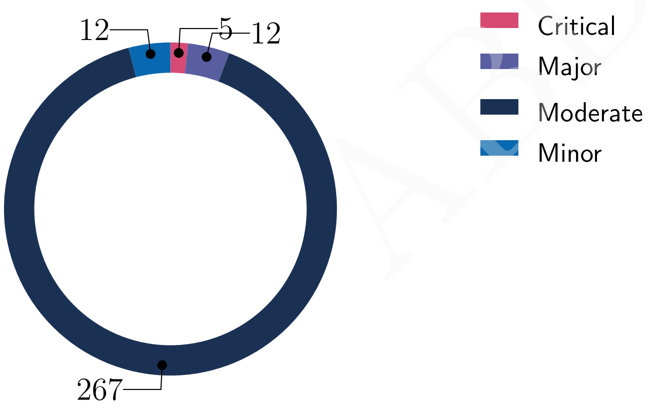
Solidity

abdk.consulting

SMART CONTRACT AUDIT CONCLUSION

by Mikhail Vladimirov and Dmitry Khovratovich
18th March 2022

We've been asked to review the 25 files in a Github repository. We found 5 critical, 12 major, and a few less important issues. All identified critical issues have been fixed or otherwise addressed in collaboration with the client.



Findings

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

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

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

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

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

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

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

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

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

ID	Severity	Category	Status
CVF-268	Minor	Bad datatype	Info
CVF-269	Major	Suboptimal	Fixed
CVF-270	Minor	Unclear behavior	Info
CVF-271	Minor	Bad naming	Info
CVF-272	Minor	Procedural	Info
CVF-273	Minor	Bad naming	Info
CVF-274	Minor	Suboptimal	Info
CVF-275	Minor	Bad datatype	Info
CVF-276	Minor	Bad datatype	Info
CVF-277	Minor	Bad datatype	Info
CVF-278	Minor	Bad datatype	Info
CVF-279	Major	Flaw	Info
CVF-280	Major	Flaw	Info
CVF-281	Minor	Procedural	Info
CVF-282	Minor	Procedural	Info
CVF-283	Minor	Unclear behavior	Info
CVF-284	Minor	Procedural	Info
CVF-285	Minor	Documentation	Info
CVF-286	Minor	Documentation	Info
CVF-287	Minor	Documentation	Info
CVF-288	Minor	Suboptimal	Info
CVF-289	Minor	Procedural	Info
CVF-290	Minor	Bad datatype	Info
CVF-291	Minor	Bad datatype	Info
CVF-292	Minor	Suboptimal	Info
CVF-293	Minor	Bad naming	Info
CVF-294	Minor	Overflow/Underflow	Info
CVF-295	Minor	Overflow/Underflow	Fixed
CVF-296	Minor	Suboptimal	Info
CVF-297	Minor	Procedural	Info

Contents

1 Document properties

Version

Version	Date	Author	Description
0.1	March 18, 2022	D. Khovratovich	Initial Draft
0.2	March 18, 2022	D. Khovratovich	Minor revision
1.0	March 18, 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 the commit:

- adapters/compound/CAdapter.sol
- adapters/compound/CFactory.sol
- adapters/lido/WstETHAdapter.sol
- adapters/BaseAdapter.sol
- adapters/BaseFactory.sol
- adapters/CropAdapter.sol
- adapters/CropFactory.sol
- external/balancer/Pool.sol
- external/balancer/Vault.sol
- external/DateTime.sol
- external/FixedMath.sol
- fuse/external/IRModel.sol
- fuse/external/PriceOracle.sol
- fuse/oracles/LP.sol
- fuse/oracles/Target.sol
- fuse/oracles/Underlying.sol
- fuse/oracles/Zero.sol
- fuse/PoolManager.sol
- modules/GClaimManager.sol
- tokens/Claim.sol
- tokens/Token.sol
- utils/libs/Errors.sol
- utils/EmergencyStop.sol
- Divider.sol
- Periphery.sol

The fixes were provided at the d88cae4 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
- **Category** Procedural
- **Status** Fixed
- **Source** Periphery.sol

Recommendation Should be "^0.8.0". Also relevant for the next files: Divider.sol, CropAdapter.sol, BaseFactory.sol, CropFactory.sol, BaseAdapter.sol, WstETHAdapter.sol, CFactory.sol, CAdapter.sol, PoolManager.sol, Zero.sol, Underlying.sol, Target.sol, LP.sol, PriceOracle.sol, IRModel.sol, GClaimManager.sol, EmergencyStop.sol, Errors.sol, Claim.sol, Token.sol, Pool.sol, Vault.sol,
Client Comment We've actually decided to bump the version to 0.8.11.

Listing 1:

```
pragma solidity ^0.8.6;
```

3.2 CVF-2

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

Description We didn't review these files.

Listing 2:

```
import { SafeERC20, ERC20 } from "@rari-capital/solmate/src/
    ↪ erc20/SafeERC20.sol";
import { Trust } from "@rari-capital/solmate/src/auth/Trust.sol
    ↪ ";
```

3.3 CVF-3

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

Description We didn't review this file.

Listing 3:

```
import { SafeERC20, ERC20 } from "@rari-capital/solmate/src/
    ↪ erc20/SafeERC20.sol";
import { Trust } from "@rari-capital/solmate/src/auth/Trust.sol
    ↪ ";

import { PoolManager } from "@sense-finance/v1-fuse/src/
    ↪ PoolManager.sol";
```

3.4 CVF-4

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

Recommendation This interface should be moved to a separate file named "YieldSpaceFactoryLike.sol". Addition comment: yes.
Client Comment Are we good with leaving it there?

Listing 4:

```
interface YieldSpaceFactoryLike {
```

3.5 CVF-5

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

Description The semantics of the arguments and the returned value is unclear.
Recommendation Consider giving then descriptive names and/or adding a documentation comment.

Listing 5:

```
    address,
    address,
    uint256
) external returns (address);
```

3.6 CVF-6

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

Recommendation The type of the "adapter" argument and the returned value could be more specific.

Listing 6:

```
26 function pools(address adapter, uint256 maturity) external view
    ↪ returns (address);
```

3.7 CVF-7

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

Recommendation This interface should be moved to a separate file named "YieldSpacePool-Like.sol".
Client Comment No longer there

Listing 7:

```
29 interface YieldSpacePoolLike {
```

3.8 CVF-8

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

Description The semantics of the returned value is unclear.
Recommendation Consider giving it a descriptive name and/or adding a documentation comment.
Client Comment No longer there

Listing 8:

```
35 ) external view returns (uint256);
```

3.9 CVF-9

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

Recommendation It would be more readable to render the value as "0.01e6".

Listing 9:

```
47 uint24 public constant UNI_POOL_FEE = 10000; // denominated in
   ↪ hundredths of a bip
```

3.10 CVF-10

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

Recommendation The key type for this factory should be "Factory".

Listing 10:

```
56 mapping(address => bool) public factories; // adapter factories
   ↪ -> is supported
```

3.11 CVF-11

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

Recommendation The types of t these arguments should be "Divider", "PoolManager", "YieldSpaceFactoryLike", and "BalancerValut" respectively.

Listing 11:

```
59 address _divider,
60 address _poolManager,
   address _ysFactory,
   address _balancerVault
```

3.12 CVF-12

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

Description The decimals property of a token is used by UI to render token amounts in a human-friendly way. Usage of this property in smart contracts and non-UI applications is discouraged.
Recommendation Consider treating all token amounts as integer numbers.

Listing 12:

```
80 uint256 stakeDecimals = ERC20(stake).decimals();
```

3.13 CVF-13

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

Description The same tokens are transferred twice.
Recommendation Consider refactoring the token flow to transfer tokens only once.

Listing 13:

```
 81 ERC20(stake).safeTransferFrom(msg.sender, address(this),
      ↪ _convertToBase(stakeSize, stakeDecimals));

 86 (zero, claim) = divider.initSeries(adapter, maturity, msg.sender
      ↪ );

116 ERC20(Adapter(adapter).getTarget()).safeTransferFrom(msg.sender,
      ↪  address(this), tBal); // pull target
    return _swapTargetForZeros(adapter, maturity, tBal, minAccepted)
      ↪ ;

130 ERC20(Adapter(adapter).underlying()).safeTransferFrom(msg.sender
      ↪ , address(this), uBal); // pull underlying

132 uint256 tBal = Adapter(adapter).wrapUnderlying(uBal); // convert
      ↪  target to underlying

145 ERC20(Adapter(adapter).getTarget()).safeTransferFrom(msg.sender,
      ↪  address(this), tBal);
    return _swapTargetForClaims(adapter, maturity, tBal);

158 ERC20(Adapter(adapter).underlying()).safeTransferFrom(msg.sender
      ↪ , address(this), uBal); // pull target

160 uint256 tBal = Adapter(adapter).wrapUnderlying(uBal); // wrap
      ↪ underlying into target

376 ERC20(zero).safeTransferFrom(msg.sender, address(this), zBal);
      ↪ // pull zeros

378 return _swap(zero, Adapter(adapter).getTarget(), zBal, pool.
      ↪ getPoolId(), minAccepted); // swap zeros for underlying

422 if (sender != address(this)) ERC20(claim).safeTransferFrom(msg.
      ↪ sender, address(this), cBal);

430 return _flashBorrow("0x", adapter, maturity, targetToBorrow);
```

(... 446, 457, 499, 502)

3.14 CVF-14

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

Description Approving uint256 maximum value is redundant and don't save gas.
Recommendation Consider approving exactly the stake size.

Listing 14:

```
ERC20(stake).safeApprove(address(divider), type(uint256).max);
```

3.15 CVF-15

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

Recommendation The type of the "factory" argument should be "Factory". The type of the "target" argument should be "ERC20".

Listing 15:

```
function onboardAdapter(address factory, address target)
    ↪ external returns (address adapterClone) {
```

3.16 CVF-16

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

Recommendation The type of the "Adapter" arguments should be "Adapter".

Listing 16:

```
111 address adapter,

125 address adapter,

141 address adapter,

154 address adapter,

170 address adapter,

186 address adapter,

203 address adapter,

217 address adapter,

233 address adapter,

247 address adapter,

266 address adapter,

284 address adapter,

307 address srcAdapter,
    address dstAdapter,

370 address adapter,

382 address adapter,

395 address adapter,

414 address adapter,

434 address adapter,

487 address adapter,

522 address adapter,

538 address adapter,
```

3.17 CVF-17

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

Description The semantics of the returned value is unclear.
Recommendation Consider giving it a descriptive name and/or explaining in the documentation comment.

Listing 17:

```
) external returns (uint256) {
```

3.18 CVF-18

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

Recommendation These function should accept an additional argument to specify the minimum output amount.

Listing 18:

```
function swapTargetForClaims(

function swapUnderlyingForClaims(

function swapClaimsForTarget(

function swapClaimsForUnderlying(

function addLiquidityFromTarget(

function addLiquidityFromUnderlying(
```

3.19 CVF-19

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

Description A user probably wants to specify the minimum amount of target to be received.
Recommendation Consider replacing these limits with a single "minOutput" limit.

Listing 19:

```
uint256[] memory minAmountsOut,
uint256 minAccepted
```

3.20 CVF-20

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

Description A user probably wants to specify the minimum amount of underlying to be received.
Recommendation Consider replacing these limits with a single "minOutput" limit.

Listing 20:

```
/// @param minAmountsOut lower limits for the tokens to receive
    ↪ (useful to account for slippage)
/// @param minAccepted only used when removing liquidity on/
    ↪ after maturity and its the min accepted when swapping
    ↪ Zeros to underlying
```

3.21 CVF-21

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

Description A user probably wants to specify how much underlying he wants to receive.
Recommendation Consider replacing these limits with a single "minOutput" limit.

Listing 21:

```
uint256[] memory minAmountsOut,
uint256 minAccepted
```

3.22 CVF-22

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

Recommendation There should be a check to ensure that both adapter use the same Target token. Otherwise it is possible to use this function to take Target tokens that are at the Periphery contract balance.

Listing 22:

```
address srcAdapter,
address dstAdapter,
```

3.23 CVF-23

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

Description A user probably wants to specify the minimum amount of LP token in the destination pool to receive.
Recommendation Consider replacing these limits with a single "minLPOutput" limit.

Listing 23:

```
uint256[] memory minAmountsOut,
uint256 minAccepted,
```

3.24 CVF-24

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

Description This function could send a significant part of the liquidity back to the user.
Recommendation Consider implementing migration in such a way that all the liquidity is migrated.

Listing 24:

```
_addLiquidity(dstAdapter, dstMaturity, tBal, mode);
```

3.25 CVF-25

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

Description This function is useless.
Recommendation Consider removing it.
Client Comment No longer there

Listing 25:

```
function price(address tokenA, address tokenB) public view
   ↪ returns (uint256) {
    // TODO: unimplemented solve this with the yield space for
       ↪ the optimal swap
    return 0.95e18;
}
```

3.26 CVF-26

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

Description Some internal function do pull tokens from the user, while other assume the tokens to be already pulled.
Recommendation Consider using a consistent approach, i.e. always pull the tokens in an external function.

Listing 26:

```
ERC20(zero).safeTransferFrom(msg.sender, address(this), zBal);
    ↪ // pull zeros

if (sender != address(this)) ERC20(claim).safeTransferFrom(msg.
    ↪ sender, address(this), cBal);

    target.safeTransferFrom(msg.sender, address(this), tBal);

ERC20(address(pool)).safeTransferFrom(msg.sender, address(this),
    ↪  lpBal);
```

3.27 CVF-27

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

Recommendation This function should accept an additional argument to specify the minimum Claim amount to receive.

Listing 27:

```
function _swapTargetForClaims(
```

3.28 CVF-28

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

Description This effectively returns part of the Target amount back to the user.
Recommendation Consider implementing the function is such a way that all the provided Target balance is consumed. This would require a flash loan to take additional Target, issue more Claim, sell Zero for Target and repay the flash loan.

Listing 28:

```
ERC20(Adapter(adapter).getTarget()).safeTransfer(msg.sender,
  ↪ tBal);
```

3.29 CVF-29

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

Description This function should accept an additional argument to specify the minimum Claim to receive.

Listing 29:

```
function _swapClaimsForTarget(
```

3.30 CVF-30

- **Severity** Moderate
- **Category** Unclear behavior
- **Status** Info
- **Source** Periphery.sol

Description The returned value doesn't include the collected interest.
Recommendation Consider returning the full obtained amount.

Listing 30:

```
return _flashBorrow("0x", adapter, maturity, targetToBorrow);
```

3.31 CVF-31

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

Description This function implements an incorrect formula.
Recommendation See the following memo for the correct one: https://hackmd.io/f6QDr5jyRd278h6RMF37ew?view#Add-Liquidity

Listing 31:

```
433 function _addLiquidity(
```

3.32 CVF-32

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

Description This function should accept an additional argument to specify the minimum LP to receive.

Listing 32:

```
433 function _addLiquidity(
```

3.33 CVF-33

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

Description The "zBalInTarget" value is always less than the "tBal" value.
Client Comment No longer there.

Listing 33:

```
451 uint256 zBalInTarget = (balances[1] * tBal) / (balances[1] +
     ↪ balances[0]);

454 uint256 tBalToPovide = tBal > zBalInTarget ? tBal - zBalInTarget
     ↪  : zBalInTarget - tBal;
```

3.34 CVF-34

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

Description Swapping Claims for Target here doesn't make sense. It will recombine the same amount of Zero and Claim that was just issued.
Recommendation Consider removing the "Sell claims" mode.

Listing 34:

```
uint256 tAmount = _swapClaimsForTarget(address(this), adapter,
    ↪ maturity, issued);
```

3.35 CVF-35

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

Description A user probably want to sent the minimum target amount to receive.
Recommendation Consider replacing these limits with a single "minOuput" limit.

Listing 35:

```
uint256[] memory minAmountsOut,
uint256 minAccepted
```

3.36 CVF-36

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

Recommendation This could be simplified as: require (result);

Listing 36:

```
require(result == true);
```

3.37 CVF-37

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

Description Despite the comment, this callback doesn't comply with ERC-3156.
Recommendation Consider either making it compliant or removing the ERC-3156 reference from the comment.

Listing 37:

```
/// @dev ERC-3156 Flash loan callback
function onFlashLoan(
    bytes calldata,
    address initiator,
    address adapter,
    uint48 maturity,
    uint256 amount
) external returns (bytes32, uint256) {
```

3.38 CVF-38

- **Severity** Moderate
- **Category** Unclear behavior
- **Status** Info
- **Source** Periphery.sol

Description The value returned from the "combine" call doesn't include Target collected as an interest, thus the actual amount of Target obtained from the "combine" call could be bigger than the returned value.

Listing 38:

```
uint256 tBal = divider.combine(adapter, maturity, zBal);
```

3.39 CVF-39

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

Description This function should return the actual amount of BPT minted.

Listing 39:

```
function _addLiquidityToSpace(
```

3.40 CVF-40

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

Recommendation Passing a single array of structs with two elements would be more efficient than two parallel arrays.

Listing 40:

```
ERC20[] memory tokens,
uint256[] memory amounts
```

3.41 CVF-41

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

Recommendation There should be a named constant used instead of "1".
Client Comment No longer there

Listing 41:

```
userData: abi.encode(1, amounts), // EXACT_TOKENS_IN_FOR_BPT_OUT
    ↪ = 1, user sends precise quantities of tokens, and
    ↪ receives an estimated but unknown (computed at run time)
    ↪ quantity of BPT. (more info here https://github.com/
    ↪ balancer-labs/docs-developers/blob/main/resources/joins-
    ↪ and-exits/pool-joins.md)
```

3.42 CVF-42

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

Recommendation The types of these arguments should be "IERC20".

Listing 42:

```
address zero,
address target,
```

3.43 CVF-43

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

Description This function looks like a hack to solve some API inconsistency problem.
Recommendation It would be better to fix the API that wants arrays of "Asset".

Listing 43:

```
function _convertERC20sToAssets(ERC20[] memory tokens) internal
    ↪ pure returns (IAsset[] memory assets) {
```

3.44 CVF-44

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

Recommendation Events are usually named via nouns, such as "FactoryChange", "SeriesSponsorship", and "AdapterOnboarding".

Listing 44:

```
event FactoryChanged(address indexed adapter, bool isOn);
event SeriesSponsored(address indexed adapter, uint256 indexed
    ↪ maturity, address indexed sponsor);
event AdapterOnboarded(address adapter);
```

3.45 CVF-45

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

Recommendation This "idOn" parameter should be indexed. Alternatively, consider replacing this event with two events: one for enabled and another for disabled factory.

Listing 45:

```
event FactoryChanged(address indexed adapter, bool isOn);
```

3.46 CVF-46

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

Recommendation The first parameter name should be "factory".

Listing 46:

```
event FactoryChanged(address indexed adapter, bool isOn);
```

3.47 CVF-47

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

Recommendation The types of adapter and factory parameters should be "Adapter" and "Factory" respectively.

Listing 47:

```
event FactoryChanged(address indexed adapter, bool isOn);
event SeriesSponsored(address indexed adapter, uint256 indexed
    ↪ maturity, address indexed sponsor);
event AdapterOnboarded(address adapter);
```

3.48 CVF-48

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

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

Listing 48:

```
import { SafeERC20, ERC20 } from "@rari-capital/solmate/src/
    ↪ erc20/SafeERC20.sol";
import { Trust } from "@rari-capital/solmate/src/auth/Trust.sol
    ↪ ";
import { ReentrancyGuard } from "@rari-capital/solmate/src/utils
    ↪ /ReentrancyGuard.sol";
```

3.49 CVF-49

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

Description We didn't review the "Trust" contract.
Client Comment Noted

Listing 49:
```
contract Divider is Trust, ReentrancyGuard, Pausable {
```

3.50 CVF-50

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

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

Listing 50:
```
address public periphery;
```

3.51 CVF-51

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

Description Having this address immutable would not allow the sense team to upgrade its multisig in the future.
Recommendation Consider implementing an ability to change this address.

Listing 51:
```
address public immutable cup; // sense team multisig
```

3.52 CVF-52

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

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

Listing 52:
```
address public immutable tokenHandler; // zero/claim deployer
```

3.53 CVF-53

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

Recommendation The key type should be "Adapter".

Listing 53:

```
mapping(address => bool) public adapters;

mapping(address => uint256) public adapterIDs;

mapping(address => mapping(uint256 => Series)) public series;

mapping(address => mapping(uint256 => mapping(address => uint256
    ↪ ))) public lscales;
```

3.54 CVF-54

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

Recommendation It would be more efficient to merge all the mappings whose key is an adapter, into a single mapping whose value is a struct encapsulating values of the original mappings.

Listing 54:

```
mapping(address => bool) public adapters;

mapping(address => uint256) public adapterIDs;

mapping(address => mapping(uint256 => Series)) public series;

mapping(address => mapping(uint256 => mapping(address => uint256
    ↪ ))) public lscales;
```

3.55 CVF-55

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

Recommendation The value type should be "Adapter".

Listing 55:

```
mapping(uint256 => address) public adapterAddresses;
```

3.56 CVF-56

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

Recommendation It would be more efficient to merge all the mappings whole keys are an adapter and a maturity, into a single mapping whose value is as truct encapsulating values of the original mappings.

Listing 56:

```
mapping(address => mapping(uint256 => Series)) public series;

mapping(address => mapping(uint256 => mapping(address => uint256
    ↪ ))) public lscales;
```

3.57 CVF-57

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

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

Listing 57:

```
address zero; // Zero ERC20 token
address claim; // Claim ERC20 token
```

3.58 CVF-58

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

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

Listing 58:

```
address claim; // Claim ERC20 token
```

3.59 CVF-59

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

Recommendation The type of the "adapter" argument should be "Adapter".

Listing 59:

```
83      address adapter,

121  function settleSeries(address adapter, uint48 maturity) external
       ↪  nonReentrant whenNotPaused {

148      address adapter,

210      address adapter,

245      address adapter,

291      address adapter,

310      address adapter,

380      address adapter,

433  function setAdapter(address adapter, bool isOn) public
       ↪ requiresTrust {

479      address adapter,

516  function _exists(address adapter, uint48 maturity) internal view
       ↪  returns (bool) {

520  function _settled(address adapter, uint48 maturity) internal
       ↪ view returns (bool) {

524  function _canBeSettled(address adapter, uint48 maturity)
       ↪ internal view returns (bool) {

535  function _isValid(address adapter, uint48 maturity) internal
       ↪ view returns (bool) {

553  function _setAdapter(address adapter, bool isOn) internal {

574  modifier onlyClaim(address adapter, uint48 maturity) {

645  function deploy(address adapter, uint48 maturity) external
       ↪ returns (address zero, address claim) {
```

3.60 CVF-60

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

Description This argument is not documented.
Recommendation Consider documenting.

Listing 60:

```
85  address sponsor
```

3.61 CVF-61

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

Description In EIP-20, the "decimals" property is used to render token amounts in a human-friendly way. Relying on this property in smart contracts is discouraged.
Recommendation Consider treating all token amounts as integers.

Listing 61:

```
93  ERC20(stake).safeTransferFrom(msg.sender, adapter,
      ↪ _convertToBase(stakeSize, ERC20(stake).decimals()));

137 ERC20(stake).safeTransferFrom(adapter, msg.sender,
      ↪ _convertToBase(stakeSize, ERC20(stake).decimals()));

157 uint256 tDecimals = target.decimals();

232 tBal = uBal.fdiv(cscale, 10**target.decimals());

256 uint256 tBase = 10**ERC20(Adapter(adapter).getTarget()).decimals
      ↪ ();

509 ERC20(stake).safeTransferFrom(adapter, stakeDst, _convertToBase(
      ↪ stakeSize, ERC20(stake).decimals()));

650 uint8 decimals = target.decimals();
```

3.62 CVF-62

- **Severity** Major
- **Category** Flaw
- **Status** Info
- **Source** Divider.sol

Description Reentrancy attack is possible here.
Recommendation Consider calling untrusted external contracts after updating the state.

Listing 62:

```
93 ERC20(stake).safeTransferFrom(msg.sender, adapter,
   ↪ _convertToBase(stakeSize, ERC20(stake).decimals()));
```

3.63 CVF-63

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

Description The scale value is obtained from the adapter twice.
Recommendation Consider obtaining once and reusing.

Listing 63:

```
104 iscale: Adapter(adapter).scale(),

106 maxscale: Adapter(adapter).scale(),
```

3.64 CVF-64

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

Description The scale and tilt values are obtained from the adapter via separate calls.
Recommendation It would be more efficient to implement a single call that returns both values at once.

Listing 64:

```
104 iscale: Adapter(adapter).scale(),

108 tilt: Adapter(adapter).tilt()
```

3.65 CVF-65

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

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

Listing 65:

```
issuance: uint128(block.timestamp),
```

3.66 CVF-66

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

Description The expression "series[adapter][maturity]" is calculated several times.
Recommendation Consider calculating once and reusing.

Listing 66:

```
series[adapter][maturity].mscale = mscale;

if (mscale > series[adapter][maturity].maxscale) {
    series[adapter][maturity].maxscale = mscale;

ERC20(target).safeTransferFrom(adapter, msg.sender, series[
  ↪ adapter][maturity].reward);
```

3.67 CVF-67

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

Description This variable is redundant. See comment below where this variable is used.

Listing 67:

```
uint256 tBase = 10**tDecimals;
```

3.68 CVF-68

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

Recommendation This code is equivalent to: fee = tBal.fmul (issuanceFee, 1e18);

Listing 68:

```
if (tDecimals != 18) {
    uint256 base = (tDecimals < 18 ? issuanceFee / (10**(18 -
      ↪ tDecimals)) : issuanceFee * 10**(tDecimals - 18));
    fee = base.fmul(tBal, tBase);
} else {
    fee = issuanceFee.fmul(tBal, tBase);
}
```

3.69 CVF-69

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

Description The expression "series[adapter][maturity]" is calculated several times.
Recommendation Consider calculating once and reusing.

Listing 69:

```
series[adapter][maturity].reward += fee;

uBal = tBalSubFee.fmul(scale, Zero(series[adapter][maturity].
  ↪ zero).BASE_UNIT());

Zero(series[adapter][maturity].zero).mint(msg.sender, uBal);
Claim(series[adapter][maturity].claim).mint(msg.sender, uBal);
```

3.70 CVF-70

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

Recommendation These two calls are equivalent to a single call: target.safeTransferFrom(msg.sender, adapter, tBal);

Listing 70:

```
target.safeTransferFrom(msg.sender, adapter, tBalSubFee);
target.safeTransferFrom(msg.sender, adapter, fee);
```

3.71 CVF-71

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

Description Reentrancy attack is possible here.
Recommendation Consider calling untrusted external contracts after updating the state.

Listing 71:

```
target.safeTransferFrom(msg.sender, adapter, tBalSubFee);
target.safeTransferFrom(msg.sender, adapter, fee);
```

3.72 CVF-72

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

Description This means that even some dust of Claim token may dramatically affect the outcome of an issuance.
Recommendation Consider collecting earning for existing Claim first, and then issuing at the current max scale rather than at the current scale. This would also "price-in" any pending earnings on just issued Claim.

Listing 72:

```
// If the caller has not collected on Claims before, use the
    ↪ current scale value to determine how many Zeros/Claims to
    ↪ mint
// so that the Claims they mint here are "clean," in that they
    ↪ have no yet-to-be-collected yield
if (scale == 0) {
```

3.73 CVF-73

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

Description So, the "scale" precision is determined using the target's decimals property. This doesn't make sense.
Recommendation Consider using a constant denominator for scale, such as 1e18.

Listing 73:

```
uBal = tBalSubFee.fmul(scale, Zero(series[adapter][maturity].zero).BASE_UNIT());

tBal = uBal.fdiv(cscale, 10**target.decimals());
```

3.74 CVF-74

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

Description The expression "series[adapter][maturity].zero" is calculated twice.
Recommendation Consider calculating once and reusing.

Listing 74:

```
uBal = tBalSubFee.fmul(scale, Zero(series[adapter][maturity].zero).BASE_UNIT());

Zero(series[adapter][maturity].zero).mint(msg.sender, uBal);
```

3.75 CVF-75

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

Description The expression "series[adapter][maturity]" is calculated several times.
Recommendation Consider calculating once and reusing.

Listing 75:

```
Zero(series[adapter][maturity].zero).burn(msg.sender, uBal);

uint256 cscale = series[adapter][maturity].mscale;

    Claim(series[adapter][maturity].claim).burn(msg.sender, uBal);
```

3.76 CVF-76

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

Description The returned value is ignored. It should be added to the returned "tBal" value, as other function expect the value returned from the "combine" function to reflect to full amount of obtained Target token.

Listing 76:

```
_collect(msg.sender, adapter, maturity, uBal, uBal, address(0));
```

3.77 CVF-77

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

Description The stored value will be the current max scale rather than the current scale.
Recommendation Consider fixing the comment.

Listing 77:

```
// We use lscale since the current scale was already stored
    ↪ there in `_collect()`
```

3.78 CVF-78

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

Description This assignment should be made only if the series is already settled.

Listing 78:

```
uint256 cscale = series[adapter][maturity].mscale;
```

3.79 CVF-79

- **Severity** Moderate
- **Category** Flaw
- **Status** Fixed
- **Source** Divider.sol

Description These functions use incorrect formulas.
Recommendation See the following memo for the correct ones: https://hackmd.io/f6QDr5jyRd278h6RMF37ew?view#The-Final-Formulas

Listing 79:

```
function redeemZero(
```

3.80 CVF-80

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

Description The expression "series[adapter][maturity]" is calculated several times.
Recommendation Consider calculating once and reusing.

Listing 80:

```
Zero(series[adapter][maturity].zero).burn(msg.sender, uBal);

tBal = (uBal * (FixedMath.WAD - series[adapter][maturity].tilt))
  ↪ / series[adapter][maturity].mscale;

if (series[adapter][maturity].mscale < series[adapter][maturity
  ↪ ].maxscale) {

    uint256 tBalZeroActual = (uBal * (FixedMath.WAD - series[
      ↪ adapter][maturity].tilt)) /
        series[adapter][maturity].maxscale;

    uint256 tBalClaimActual = tBalZeroActual.fmul(series[adapter
      ↪ ][maturity].tilt, tBase);
```

3.81 CVF-81

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

Recommendation It would be more efficient to return WAD-tilt rather than tilt from an adapter.

Listing 81:

```
tBal = (uBal * (FixedMath.WAD - series[adapter][maturity].tilt)) / series[adapter][maturity].mscale;
```

3.82 CVF-82

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

Description This assignment should be done only if mscale >= maxscale.

Listing 82:

```
tBal = (uBal * (FixedMath.WAD - series[adapter][maturity].tilt)) / series[adapter][maturity].mscale;
```

3.83 CVF-83

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

Description This condition could be evaluated when settling as series. No need to calculate it on every redeem.

Listing 83:

```
if (series[adapter][maturity].mscale < series[adapter][maturity].maxscale) {

    if (_series.mscale < _series.maxscale) {
```

3.84 CVF-84

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

Description The expression "series[adapter][maturity].tilt" is calculated several times.
Recommendation Consider calculating once and reusing.

Listing 84:

```
tBal = (uBal * (FixedMath.WAD - series[adapter][maturity].tilt)) / series[adapter][maturity].mscale;

    uint256 tBalZeroActual = (uBal * (FixedMath.WAD - series[adapter][maturity].tilt)) /

    uint256 tBalClaimActual = tBalZeroActual.fmul(series[adapter][maturity].tilt, tBase);
```

3.85 CVF-85

- **Severity** Critical
- **Category** Flaw
- **Status** Fixed
- **Source** Divider.sol

Recommendation Should be "tBase" instead of "FixedMath.WAD".

Listing 85:

```
tBal = (uBal * (FixedMath.WAD - series[adapter][maturity].tilt)) / series[adapter][maturity].mscale;

    uint256 tBalZeroActual = (uBal * (FixedMath.WAD - series[adapter][maturity].tilt)) /
```

3.86 CVF-86

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

Description The "tBalZeroActual" value is calcualted rounding down, thus its impact into the "tBal" value is rounded up, i.e. toward the user. This could lead to a situation when the protocol will not be able to pay all Claim and Zero holders in full. A good practice is to always round towards the protocol, i.e. against a user.

Listing 86:

```
uint256 tBalZeroActual = (uBal * (FixedMath.WAD - series[adapter][maturity].tilt)) /
  series[adapter][maturity].maxscale;

    uint256 shortfall = tBal - tBalZeroActual;

        tBal += shortfall;
```

3.87 CVF-87

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

Description The expression "series[adapter][maturity].maxscale" is calculated twice.
Recommendation Consider calculating once and reusing.

Listing 87:

```
if (series[adapter][maturity].mscale < series[adapter][maturity].maxscale) {

        series[adapter][maturity].maxscale;
```

3.88 CVF-88

- **Severity** Critical
- **Category** Flaw
- **Status** Fixed
- **Source** Divider.sol

Description The "shortfall" value calculated here is always zero.

Listing 88:

```
tBal = tBalZeroActual;

    uint256 shortfall = tBal - tBalZeroActual;
```

3.89 CVF-89

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

Description The value "series[adapter][maturity]" is already read into the memory as is accessible as "_series". No need to read its field from the storage.

Listing 89:

```
323 Claim claim = Claim(series[adapter][maturity].claim);
```

3.90 CVF-90

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

Description It is really possible for a user to have non-zero Claim balance, but zero lscale? If so, setting the "lscale" value to "iscale" would allow such user to collect earnings as if he owns his balance singe the series was initialized.
Recommendation Consider setting "lscale" to the current "maxscale" here, rather than to the "iscale".

Listing 90:

```
326 // If this is the Claim holder's first time collecting and
    ↪ nobody sent these Claims to them,
    // set the "last scale" value to the scale at issuance for this
    ↪ series
    if (lscale == 0) lscale = _series.iscale;
```

3.91 CVF-91

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

Description The "maxscale" value used to calculate the "tBalNow" value was either just written to the storage or just read from it. In both cases, it was already seen and thus don't need to be read from the storage again.
Recommendation Consider reusing the already seen value.

Listing 91:

```
            _series.maxscale = cscale;

            lscales[adapter][maturity][usr] = _series.maxscale;

uint256 tBalNow = uBal.fdiv(_series.maxscale, claim.BASE_UNIT());
```

3.92 CVF-92

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

Description The "tBalNow" value is rounded down, thus its impact into the "collected" value is rounded up, i.e. toward the user. This could lead to a situation, that the protocol will not have enough target tokens to pay to all the Zero and Claim holders. A good practice is to always round towards the protocol, i.e. against a user.

Listing 92:

```
uint256 tBalNow = uBal.fdiv(_series.maxscale, claim.BASE_UNIT());
collected = uBal.fdiv(lscale, claim.BASE_UNIT()) - tBalNow;
```

3.93 CVF-93

- **Severity** Critical
- **Category** Flaw
- **Status** Fixed
- **Source** Divider.sol

Description In case the "to" address already has some Claim with non-collected earnings, these non-collected earnings will be lost.
Recommendation Consider collecting the earnings for the "to" address first.

Listing 93:

```
// If this collect is a part of a token transfer to another
    ↪ address, set the receiver's
// last collection to this scale (as all yield is being stripped
    ↪  off before the Claims are sent)
if (to != address(0)) {
    lscales[adapter][maturity][to] = _series.maxscale;
    uint256 tBalTransfer = uBalTransfer.fdiv(_series.maxscale,
        ↪ claim.BASE_UNIT());
    Adapter(adapter).notify(usr, tBalTransfer, false);
    Adapter(adapter).notify(to, tBalTransfer, true);
}
```

3.94 CVF-94

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

Description This function seems to implement the right formula, but does it in an over-complicated way.
Recommendation See the following memo for simpler version of the same formula: https://hackmd.io/f6QDr5jyRd278h6RMF37ew?view

Listing 94:

```
function _redeemClaim(
```

3.95 CVF-95

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

Description The "tBalZeroIdeal" value is calculated rounding down, thus its impact into the "tBal" value is rounded up. This could lead to a situation when the protocol will not be able to pay all Zero and Claim holders in full. A good practice is to always round towards the protocol, i.e. against a user.

Listing 95:

```
uint256 tBalZeroIdeal = (uBal * (FixedMath.WAD - _series.tilt)) / _series.mscale;

uint256 shortfall = tBalZeroIdeal - tBalZeroActual;

    tBal -= shortfall;
```

3.96 CVF-96

- **Severity** Critical
- **Category** Flaw
- **Status** Info
- **Source** Divider.sol

Recommendation It should be "10**target_decimals" instead of "FixedMath,.WAD".

Listing 96:

```
uint256 tBalZeroIdeal = (uBal * (FixedMath.WAD - _series.tilt)) / _series.mscale;

uint256 tBalZeroActual = (uBal * (FixedMath.WAD - _series.tilt)) / _series.maxscale;
```

3.97 CVF-97

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

Description These events are emitted even if nothing actually changed.

Listing 97:

```
emit GuardChanged(target, cap);

emit GuardedChanged(guarded);

emit PeripheryChanged(periphery);

emit PermissionlessChanged(permissionless);
```

3.98 CVF-98

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

Description In case of a negative yield for the target token, the scale at maturity could actually me less than the initial scale.
Recommendation Consider removing this check.
Client Comment No longer there

Listing 98:

```
require(mscale > series[adapter][maturity].iscale, Errors.
    ↪ InvalidScaleValue);
```

3.99 CVF-99

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

Description The expression "series[adapter][maturity]" is calculated several times.
Recommendation Consider calculating once and reusing.

Listing 99:

```
require(mscale > series[adapter][maturity].iscale, Errors.InvalidScaleValue);

series[adapter][maturity].mscale = mscale;
if (mscale > series[adapter][maturity].maxscale) {
    series[adapter][maturity].maxscale = mscale;

address stakeDst = block.timestamp <= maturity + SPONSOR_WINDOW ? series[adapter][maturity].sponsor : cup;
uint256 reward = series[adapter][maturity].reward;
```

3.100 CVF-100

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

Description This loop doesn't scale and for a really big number of user may not fit into the block gas limit.
Recommendation Consider implementing an ability to split the backfill operation into several transaction.

Listing 100:

```
// Set user's last scale values the Series (needed for the `collect` method)
for (uint256 i = 0; i < _usrs.length; i++) {
    lscales[adapter][maturity][_usrs[i]] = _lscales[i];
}
```

3.101 CVF-101

- **Severity** Moderate
- **Category** Flaw
- **Status** Fixed
- **Source** Divider.sol

Description The condition can only be true in case the adapter was disabled due to the cutoff check above. When the adapter is disable, it is not the sponsor's fault if a series wasn't settled in time. So this should be: address stakeDst = adapters[adapter] ? sup : series[adapter][maturity].sponsor;

Listing 101:

```
address stakeDst = block.timestamp <= maturity + SPONSOR_WINDOW ? series[adapter][maturity].sponsor : cup;
```

3.102 CVF-102

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

Recommendation The "cutoff" value should only be calculated in case the first part of the corresponding conjunction is false. Otherwise the "cutoff" value won't be used.

Listing 102:

```
uint256 cutoff = maturity + SPONSOR_WINDOW + SETTLEMENT_WINDOW;

    return maturity - SPONSOR_WINDOW <= block.timestamp && cutoff >= block.timestamp;

    return maturity + SPONSOR_WINDOW < block.timestamp && cutoff >= block.timestamp;
```

3.103 CVF-103

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

Recommendation There should be a enum for valid modes.

Listing 103:

```
if (mode == 0) {

if (mode == 1) {
```

3.104 CVF-104

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

Recommendation Should be "else if" for readability.

Listing 104:

```
if (mode == 1) {
```

3.105 CVF-105

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

Recommendation Should be "else return" for readability.

Listing 105:

```
return false;
```

3.106 CVF-106

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

Recommendation These values should be cleared when an adapter is disabled.

Listing 106:

```
adapterAddresses[adapterCounter] = adapter;
adapterIDs[adapter] = adapterCounter;
```

3.107 CVF-107

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

Description This function always rounds down, while it could be helpful to round up in some cases.
Recommendation Consider implementing a rounding up version of this function.
Client Comment No longer there

Listing 107:

```
function _convertToBase(uint256 amount, uint256 decimals)
    ↪ internal pure returns (uint256) {
```

3.108 CVF-108

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

Recommendation Events are usually named via nouns, such as "Backfill", "GuardChange", "AdapterChange", etc.

Listing 108:

```
event Backfilled(

event GuardChanged(address indexed target, uint256 indexed cap);
event AdapterChanged(address indexed adapter, uint256 indexed id
    ↪ , bool isOn);
event PeripheryChanged(address indexed periphery);

event SeriesInitialized(

event Issued(address indexed adapter, uint256 indexed maturity,
    ↪ uint256 balance, address indexed sender);
event Combined(address indexed adapter, uint256 indexed maturity
    ↪ , uint256 balance, address indexed sender);
event Collected(address indexed adapter, uint256 indexed
    ↪ maturity, uint256 collected);

event SeriesSettled(address indexed adapter, uint256 indexed
    ↪ maturity, address indexed settler);
event ZeroRedeemed(address indexed adapter, uint256 indexed
    ↪ maturity, uint256 redeemed);
event ClaimRedeemed(address indexed adapter, uint256 indexed
    ↪ maturity, uint256 redeemed);

event GuardedChanged(bool indexed guarded);
event PermissionlessChanged(bool indexed permissionless);
```

3.109 CVF-109

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

Description Indexing the "cap" parameter is redundant.

Listing 109:

```
event GuardChanged(address indexed target, uint256 indexed cap);
```

3.110 CVF-110

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

Description The "idOn" parameter should be indexed. Alternatively, consider replacing this event with two events: one for enabling an adapter and another for disabling it. The letter event wouldn't have the "id" parameter is such a case.

Listing 110:

```
event AdapterChanged(address indexed adapter, uint256 indexed id
    ↪ , bool isOn);
```

3.111 CVF-111

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

Recommendation The types of the parameters holding addresses of smart contracts, could be made more specific.

Listing 111:

```
event GuardChanged(address indexed target, uint256 indexed cap);
event AdapterChanged(address indexed adapter, uint256 indexed id, bool isOn);
event PeripheryChanged(address indexed periphery);

    address adapter,

    address zero,
    address claim,

    address indexed target

event Issued(address indexed adapter, uint256 indexed maturity, uint256 balance, address indexed sender);
event Combined(address indexed adapter, uint256 indexed maturity, uint256 balance, address indexed sender);
event Collected(address indexed adapter, uint256 indexed maturity, uint256 collected);

event SeriesSettled(address indexed adapter, uint256 indexed maturity, address indexed settler);
event ZeroRedeemed(address indexed adapter, uint256 indexed maturity, uint256 redeemed);
event ClaimRedeemed(address indexed adapter, uint256 indexed maturity, uint256 redeemed);
```

3.112 CVF-112

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

Recommendation This parameter should be indexed.

Listing 112:

```
address adapter,
```

3.113 CVF-113

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

Recommendation This contract should be moved to a separate file named "Token-Holder.sol".

Listing 113:

```
contract TokenHandler is Trust {
```

3.114 CVF-114

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

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

Listing 114:

```
address public divider;
```

3.115 CVF-115

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

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

Listing 115:

```
constructor() Trust(msg.sender) {}
```

3.116 CVF-116

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

Recommendation The type of the "_divider" argument should be "Divider".

Listing 116:

```
function init(address _divider) external requiresTrust {
```

3.117 CVF-117

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

Recommendation The type of the "adapter" argument should be "Adapter".

Listing 117:

```
function deploy(address adapter, uint48 maturity) external
  ↪ returns (address zero, address claim) {
```

3.118 CVF-118

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

Recommendation The types of the returned values should be "Zero" and "Claim".

Listing 118:

```
function deploy(address adapter, uint48 maturity) external
  ↪ returns (address zero, address claim) {
```

3.119 CVF-119

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

Description We didn't review the "DateTime" library.

Listing 119:

```
(, string memory m, string memory y) = DateTime.toDateString(
  ↪ maturity);

string memory adapterId = DateTime.uintToString(Divider(divider)
  ↪ .adapterIDs(adapter));
```

3.120 CVF-120

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

Description Deploying separate copies of Zero and Claim contracts for ever series is suboptimal.
Recommendation Consider deploying the code once and then deploying transparent proxies for them like described here: https://eips.ethereum.org/EIPS/eip-1167

Listing 120:

```
new Zero(

new Claim(
```

3.121 CVF-121

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

Description We didn't review this file.

Listing 121:

```
import { Trust } from "@rari-capital/solmate/src/auth/Trust.sol";
import { ERC20, SafeERC20 } from "@rari-capital/solmate/src/erc20/SafeERC20.sol";
```

3.122 CVF-122

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

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

Listing 122:

```
address public reward;
```

3.123 CVF-123

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

Description The semantics of the key and value for this mapping is unclear.
Recommendation Consider documenting.

Listing 123:

```
24 mapping(address => uint256) public tBalance;
```

3.124 CVF-124

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

Description The comment is inaccurate. Actually, this is reward tokens paid per user. The "per collected target" part shouldn't be here.

Listing 124:

```
25 mapping(address => uint256) public rewarded; // reward token per
    ↪ collected target per user
```

3.125 CVF-125

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

Recommendation Events are usually named via nouns.

Listing 125:

```
27 event Distributed(address indexed usr, address indexed token,
    ↪ uint256 amount);
```

3.126 CVF-126

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

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

Listing 126:

```
30 address _divider,
```

3.127 CVF-127

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

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

Listing 127:

```
32 address _reward
```

3.128 CVF-128

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

Recommendation This is calculated twice. Consider optimizing.

Listing 128:

```
56 rewarded[_usr] = tBalance[_usr].fmulUp(share, FixedMath.RAY);

68 uint256 curr = tBalance[_usr].fmul(share, FixedMath.RAY);
```

3.129 CVF-129

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

Description The expression "ERC20(reward).balanceOf(address(this))" is calculated twice.
Recommendation Consider calculating once and reusing.

Listing 129:

```
64 uint256 crop = ERC20(reward).balanceOf(address(this)) −
     ↪ rewardBal;

70 rewardBal = ERC20(reward).balanceOf(address(this));
```

3.130 CVF-130

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

Description We didn't review these files.

Listing 130:

```
import { ERC20 } from "@rari-capital/solmate/src/erc20/SafeERC20
    ↪ .sol";
import { Bytes32AddressLib } from "@rari-capital/solmate/src/
    ↪ utils/Bytes32AddressLib.sol";
```

3.131 CVF-131

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

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

Listing 131:

```
address public immutable divider;
```

3.132 CVF-132

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

Recommendation The type for this variable could be more specific.

Listing 132:

```
address public immutable protocol; // protocol's data contract
    ↪ address
```

3.133 CVF-133

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

Recommendation Events are usually named via nouns.

Listing 133:

```
event AdapterDeployed(address addr, address indexed target);
event DeltaChanged(uint256 delta);
event AdapterImplementationChanged(address implementation);
event ProtocolChanged(address protocol);
```

3.134 CVF-134

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

Recommendation The type of this field could be more specific.

Listing 134:

```
address oracle; // oracle address
```

3.135 CVF-135

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

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

Listing 135:

```
address stake; // token to stake at issuance
```

3.136 CVF-136

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

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

Listing 136:

```
address _divider,
```

3.137 CVF-137

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

Recommendation The type of this argument should be more specific.

Listing 137:

```
address _protocol ,
```

3.138 CVF-138

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

Description There are no range checks for the parameters.
Recommendation Consider adding proper checks.

Listing 138:

```
FactoryParams memory _factoryParams
```

3.139 CVF-139

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

Recommendation The type of the "_target" argument should be "IERC20".

Listing 139:

```
function deployAdapter(address _target) external virtual returns (address adapterClone) {

function _exists(address _target) internal virtual returns (bool);
```

3.140 CVF-140

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

Recommendation The return type should be "Adapter".

Listing 140:

```
function deployAdapter(address _target) external virtual returns
    ↪ (address adapterClone) {
```

3.141 CVF-141

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

Description We didn't review this file.

Listing 141:

```
import { Bytes32AddressLib } from "@rari-capital/solmate/src/
    ↪ utils/Bytes32AddressLib.sol";
```

3.142 CVF-142

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

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

Listing 142:

```
address public immutable reward;
```

3.143 CVF-143

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

Recommendation The type for this field should be "Divider".

Listing 143:

```
address _divider,
```

3.144 CVF-144

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

Recommendation The type for this field should be more specific.

Listing 144:

```
address _protocol,
```

3.145 CVF-145

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

Recommendation The type for this field should be "IERC20".

Listing 145:

```
address _reward
```

3.146 CVF-146

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

Recommendation The type of the "_target" argument should be "IERC20".

Listing 146:

```
function deployAdapter(address _target) external override
    ↪ returns (address adapterClone) {
```

3.147 CVF-147

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

Recommendation The return type should be "CropAdapter".

Listing 147:

```
function deployAdapter(address _target) external override
    ↪ returns (address adapterClone) {
```

3.148 CVF-148

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

Description We didn't review this file.

Listing 148:

```
import { SafeERC20, ERC20 } from "@rari-capital/solmate/src/
    ↪ erc20/SafeERC20.sol";
```

3.149 CVF-149

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

Recommendation This interface should be moved to a separate file named "IPeriphery.sol".

Listing 149:

```
interface IPeriphery {
```

3.150 CVF-150

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

Recommendation The type for this argument should be "Adapter".

Listing 150:

```
address adapter,
```

3.151 CVF-151

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

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

Listing 151:

```
) external returns (bytes32, uint256);
```

3.152 CVF-152

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

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

Listing 152:

```
address public divider;
```

3.153 CVF-153

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

Recommendation The type of this field should be more specific.

Listing 153:

```
address oracle; // oracle address
```

3.154 CVF-154

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

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

Listing 154:

```
address stake; // token to stake at issuance
```

3.155 CVF-155

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

Recommendation There should be named constants for the supported modes, or even enum.

Listing 155:

```
uint8 mode; // 0 for monthly, 1 for weekly
```

3.156 CVF-156

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

Description This contract doesn't implement any token interface. Why does it need a name and, especially, a symbol?

Listing 156:

```
string public name;
string public symbol;
```

3.157 CVF-157

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

Recommendation Events are usually named via nouns.

Listing 157:

```
event Initialized();
```

3.158 CVF-158

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

Description This event is almost useless.
Recommendation Consider removing it or adding some helpful parameters to it.
Client Comment No longer there

Listing 158:

```
event Initialized();

    emit Initialized();
```

3.159 CVF-159

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

Recommendation The type of the "_divider" argument should be "Divider".

Listing 159:

```
function initialize(address _divider, AdapterParams memory
    ↪ _adapterParams) public virtual initializer {
```

3.160 CVF-160

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

Description There are no validity checks for most of the parameters.
Recommendation Consider adding proper checks. For example, check that "delta" is non-zero, "mode" is valid etc.

Listing 160:

```
function initialize(address _divider, AdapterParams memory
    ↪ _adapterParams) public virtual initializer {
```

3.161 CVF-161

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

Recommendation Should be "<=" to allow adapters with exactly one valid maturity.

Listing 161:

```
require(_adapterParams.minm < _adapterParams.maxm, Errors.
    ↪ InvalidMaturityOffsets);
```

3.162 CVF-162

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

Description The suffix is longer than a typical token symbol.
Recommendation Consider using a shorter suffix, or even some punctuation character, like: "@" + target.symbol().

Listing 162:

```
symbol = string(abi.encodePacked(ERC20(_adapterParams.target).symbol(), "-adapter"));
```

3.163 CVF-163

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

Description The argument is redundant. It is not used in this contract and is passed to the callback as is, while the caller could use the "data" argument to pass any necessary information to the callback.
Recommendation Consider removing this argument.

Listing 163:

```
address adapter,
```

3.164 CVF-164

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

Description Other contracts use the "SafeERC20" library for token transfers.
Recommendation Consider using it here as well.

Listing 164:

```
require(target.transfer(address(receiver), amount), Errors.FlashTransferFailed);

require(target.transferFrom(address(receiver), address(this), amount), Errors.FlashRepayFailed);
```

3.165 CVF-165

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

Description The delta change could be bypassed by sampling the scale, then moving it (i.e. via a flash loan), and then sampling again in the same block. In such a case, the "elapsed" value will be zero, this the delta check will not be performed, but the stored lscale value will be updated.
Client Comment We removed the delta check.

Listing 165:

```
101 if (elapsed > 0 && lvalue != 0) {

108     if (growthPerSec > adapterParams.delta) revert(Errors.
        ↪ InvalidScaleValue);
```

3.166 CVF-166

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

Description In EIP-20, the "decimals" property is used to render token amounts in a human-friendly way. Relying on this property in smart contracts is discouraged.
Recommendation Consider treating all token amounts as integers.

Listing 166:

```
105 10**ERC20(adapterParams.target).decimals()
```

3.167 CVF-167

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

Description This value could be precomputed at the initialization time.
Client Comment No longer there

Listing 167:

```
105 10**ERC20(adapterParams.target).decimals()
```

3.168 CVF-168

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

Description Block timestamps are inaccurate in Ethereum. For consequent blocks, the difference in timestamps could be as low as 1 second, thus even a small scale change could exceed the threshold.
Recommendation Consider setting two thresholds: one for a relative price change per second and another for a relative price change regardless of the time period. For example, allow at ,most 0.1% change per second or 5% change regardless of the time period.
Client Comment We removed the delta check.

Listing 168:

```
if (growthPerSec > adapterParams.delta) revert(Errors.
    ↪ InvalidScaleValue);
```

3.169 CVF-169

- **Severity** Moderate
- **Category** Unclear behavior
- **Status** Info
- **Source** BaseAdapter.sol

Description The stored timestamp is updated only in case the scale value did change. This makes the delta check less efficient. Lets, assume that at the time moment t_0 the scale was 1,0. Then at the time moment t_0 + 100000, the scale was still 1,0, however at the time moment t_0 + 100001 the scale is 2.0, i.e. raised 100% in one second. When performing the delta check, the time period concerned will be 100001 seconds, rather than 1 second, so this sharp move could go under radar.
Recommendation Consider always updating the lscale timestamp.
Client Comment We removed the delta check.

Listing 169:

```
if (_value != lvalue) {

    _lscale.timestamp = block.timestamp;
```

3.170 CVF-170

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

Description The Divider smart contract doesn't support a mutable tilt value, so this function should always return the same value for an adapter.
Recommendation Consider highlighting this in the documentation comment. Also, this function should be declared as "view".

Listing 170:

```
128 function tilt() external virtual returns (uint128) {
```

3.171 CVF-171

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

Description The format for the returned value is unclear.
Recommendation Consider documenting.

Listing 171:

```
152 function getUnderlyingPrice() external view virtual returns (
      ↪ uint256);

160 function getIssuanceFee() external view returns (uint256) {
```

3.172 CVF-172

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

Recommendation The return type type should be IERC20.

Listing 172:

```
156 function getTarget() external view returns (address) {
```

3.173 CVF-173

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

Recommendation This interface should be moved to a separate file named "WstETHInterface.sol".

Listing 173:

```
interface WstETHInterface {
```

3.174 CVF-174

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

Recommendation This interface should be moved to a separate file named "StETHInterface.sol".

Listing 174:

```
interface StETHInterface {
```

3.175 CVF-175

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

Recommendation This interface should be moved to a separate file named "ICurveStableSwap.sol".

Listing 175:

```
interface ICurveStableSwap {
```

3.176 CVF-176

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

Description The semantics of these functions are unclear.
Recommendation Consider documenting.

Listing 176:

```
function get_dy(
    int128 i,
    int128 j,
    uint256 dx
) external view returns (uint256);

function exchange(
    int128 i,
    int128 j,
    uint256 dx,
    uint256 min_dy
) external payable returns (uint256);
```

3.177 CVF-177

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

Recommendation This interface should be moved to a separate file named "IWETH.sol".

Listing 177:

```
interface IWETH {
```

3.178 CVF-178

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

Description Hardcoding mainnet addresses is discouraged, as it makes it harder to test the code in testnets.
Recommendation Consider turning these constants into immutable variables set in the constructor.

Listing 178:

```
address public constant CETH = 0
    ↪ x4Ddc2D193948926D02f9B1fE9e1daa0718270ED5;
address public constant WETH = 0
    ↪ xC02aaA39b223FE8D0A0e5C4F27eAD9083C756Cc2;
address public constant WSTETH = 0
    ↪ x7f39C581F595B53c5cb19bD0b3f8dA6c935E2Ca0;
address public constant STETH = 0
    ↪ xae7ab96520DE3A18E5e111B5EaAb095312D7fE84;
address public constant CURVESINGLESWAP = 0
    ↪ xDC24316b9AE028F1497c275EB9192a3Ea0f67022;
```

3.179 CVF-179

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

Recommendation The types of these constants should be more specific.

Listing 179:

```
address public constant CETH = 0
    ↪ x4Ddc2D193948926D02f9B1fE9e1daa0718270ED5;
address public constant WETH = 0
    ↪ xC02aaA39b223FE8D0A0e5C4F27eAD9083C756Cc2;
address public constant WSTETH = 0
    ↪ x7f39C581F595B53c5cb19bD0b3f8dA6c935E2Ca0;
address public constant STETH = 0
    ↪ xae7ab96520DE3A18E5e111B5EaAb095312D7fE84;
address public constant CURVESINGLESWAP = 0
    ↪ xDC24316b9AE028F1497c275EB9192a3Ea0f67022;
```

3.180 CVF-180

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

Recommendation The type of the "_divider" argument should be "Divider".

Listing 180:

```
function initialize(address _divider, AdapterParams memory
    ↪ _adapterParams) public virtual override initializer {
```

3.181 CVF-181

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

Description This function doesn't take the CurveStableSwap rate into account.

Listing 181:

```
function _scale() internal virtual override returns (uint256) {
```

3.182 CVF-182

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

Description The STETH <-> ETH conversion is implemented differently inside wrap and unwrap. This could lead to a situation when the wrapping and the unwrapping rates are different.

Listing 182:

```
uint256 eth = ICurveStableSwap(CURVESINGLESWAP).exchange(int128
    ↪ (1), int128(0), amount, minDy);

uint256 stETH = StETHInterface(STETH).submit{ value: amount }(
    ↪ address(0)); // stake ETH (returns wstETH)
```

3.183 CVF-183

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

Recommendation This function should check that msg.sender is eligible, such as WETH or CURVESINGLESWAP.

Listing 183:

```
fallback() external payable {}
```

3.184 CVF-184

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

Recommendation This interface should be moved to a separate file named "ComptrollerLike.sol".

Listing 184:

```
interface ComptrollerLike {
```

3.185 CVF-185

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

Recommendation The argument type should be "IERC20".

Listing 185:

```
function markets(address target)
```

3.186 CVF-186

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

Recommendation The return type should be "PriceOracleInterface".

Listing 186:

```
function oracle() external returns (address);
```

3.187 CVF-187

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

Description Hardcoding mainnet addresses is discouraged, as it makes it harder to test the code in testnets.
Recommendation Consider turning this constant into an immutable variable set in the constructor.

Listing 187:

```
address public constant COMPTROLLER = 0
    ↪ x3d9819210A31b4961b30EF54bE2aeD79B9c9Cd3B;
```

3.188 CVF-188

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

Recommendation The type for this argument should be "Divider".

Listing 188:

```
address _divider,
```

3.189 CVF-189

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

Recommendation The type for this argument should be "Adapter".

Listing 189:

```
address _adapterImpl,
```

3.190 CVF-190

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

Recommendation The type for this argument should be more specific.

Listing 190:

```
address _reward
```

3.191 CVF-191

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

Recommendation The argument type should be more specific.

Listing 191:

```
function _exists(address _target) internal virtual override
    ↪ returns (bool isListed) {
```

3.192 CVF-192

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

Description We didn't review this file.

Listing 192:

```
import { ERC20, SafeERC20 } from "@rari-capital/solmate/src/
    ↪ erc20/SafeERC20.sol";
```

3.193 CVF-193

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

Recommendation This interface should be moved to a separate file named "CTokenInterface.sol".

Listing 193:

```
interface CTokenInterface {
```

3.194 CVF-194

- **Severity** Moderate
- **Category** Bad datatype
- **Status** Fixed
- **Source** CAdapter.sol

Recommendation The "decimals" property has type "uint8" rather than "uint256".

Listing 194:

```
function decimals() external returns (uint256);
```

3.195 CVF-195

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

Recommendation This interface should be moved to a separate file named "ComptrollerInterface.sol".

Listing 195:

```
interface ComptrollerInterface {
```

3.196 CVF-196

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

Recommendation This contract should be moved to a separate file named "PriceOracleInterface.sol".

Listing 196:

```
interface PriceOracleInterface {
```

3.197 CVF-197

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

Description Hardcoding mainnet addresses is discouraged, as it makes it harder to test the code in testnets.
Recommendation Consider turning this constant into an immutable variable set in the constructor.

Listing 197:

```
address public constant COMPTROLLER = 0
    ↪ x3d9819210A31b4961b30EF54bE2aeD79B9c9Cd3B;
```

3.198 CVF-198

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

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

Listing 198:

```
address _divider,
```

3.199 CVF-199

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

Description Querying the underlying address from the CToken every time is suboptimal.
Recommendation Consider querying this address once during initialization and saving in the storage.

Listing 199:

```
uint256 decimals = CTokenInterface(t.underlying()).decimals();

ERC20 u = ERC20(CTokenInterface(adapterParams.target).underlying
    ↪ ());

ERC20 u = ERC20(CTokenInterface(adapterParams.target).underlying
    ↪ ());
```

3.200 CVF-200

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

Recommendation This is basically equivalent to: return t.exchangeRateCurrent() * 1e10;

Listing 200:

```
return t.exchangeRateCurrent().fdiv(10**(10 + decimals), 10**
    ↪ decimals);
```

3.201 CVF-201

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

Recommendation Zero here should be a named constant.

Listing 201:

```
require(CTokenInterface(adapterParams.target).mint(uBal) == 0, "
    ↪ Mint failed");

require(CTokenInterface(adapterParams.target).redeem(tBal) == 0,
    ↪ "Redeem failed");
```

3.202 CVF-202

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

Description We didn't review these files.

Listing 202:

```
import { Trust } from "@rari-capital/solmate/src/auth/Trust.sol
    ↪ ";
import { ERC20 } from "@rari-capital/solmate/src/erc20/ERC20.sol
    ↪ ";
import { Bytes32AddressLib } from "@rari-capital/solmate/src/
    ↪ utils/Bytes32AddressLib.sol";
```

3.203 CVF-203

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

Description These references look like external.
Recommendation Internal referenced would look like: ../utils/libs/Errors.sol ../Divider.sol ../adapters/BaseAdapter.sol

Listing 203:

```
17 import { Errors } from "@sense-finance/v1-utils/src/libs/Errors.
   ↪ sol";
import { Divider } from "@sense-finance/v1-core/src/Divider.sol
   ↪ ";
import { BaseAdapter as Adapter } from "@sense-finance/v1-core/
   ↪ src/adapters/BaseAdapter.sol";
```

3.204 CVF-204

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

Recommendation This interface should be moved to a separate file named "FuseDirectory-Like.sol".

Listing 204:

```
21 interface FuseDirectoryLike {
```

3.205 CVF-205

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

Recommendation The type of these arguments should be more specific.

Listing 205:

```
28     address priceOracle

44     address[] memory underlyings,

51 function add(address[] calldata underlyings, PriceOracle[]
   ↪ calldata _oracles) external;
```

3.206 CVF-206

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

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

Listing 206:

```
) external returns (uint256, address);

) external returns (uint256);

function _acceptAdmin() external returns (uint256);
```

3.207 CVF-207

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

Recommendation This interface should be moved to a separate file named "ComptrollerLike.sol".

Listing 207:

```
interface ComptrollerLike {
```

3.208 CVF-208

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

Recommendation This interface should be moved to a separate file named "MasterOracleLike.sol".

Listing 208:

```
interface MasterOracleLike {
```

3.209 CVF-209

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

Recommendation The types of these variables should be more specific.

Listing 209:

```
57 address public immutable comptrollerImpl;
   address public immutable cERC20Impl;
   address public immutable fuseDirectory;
60 address public immutable divider;

62 address public immutable oracleImpl; // master oracle from Fuse
   address public immutable targetOracle;
   address public immutable zeroOracle;
   address public immutable lpOracle;
   address public immutable underlyingOracle;

68 address public comptroller;
   address public masterOracle;
```

3.210 CVF-210

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

Recommendation The type of this field should be more specific.

Listing 210:

```
78 address irModel;
```

3.211 CVF-211

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

Recommendation The key type for this mapping should be "IERC20".

Listing 211:

```
90 mapping(address => bool) public tInits;
```

3.212 CVF-212

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

Recommendation The first key type for these mappings should be "Adapter".

Listing 212:

```
mapping(address => mapping(uint256 => SeriesStatus)) public
    ↪ sStatus;

mapping(address => mapping(uint256 => address)) public sPools;
```

3.213 CVF-213

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

Recommendation It would be more efficient to merge all mappings whose first key is "Adapter" into a single mapping whose key is "Adapter" and a value is a struct encapsulating the values of the original mappings.

Listing 213:

```
mapping(address => mapping(uint256 => SeriesStatus)) public
    ↪ sStatus;

mapping(address => mapping(uint256 => address)) public sPools;
```

3.214 CVF-214

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

Recommendation Events are usually named via nouns.

Listing 214:

```
event SetParams(bytes32 indexed what, AssetParams data);
event PoolDeployed(string name, address comptroller, uint256
    ↪ poolIndex, uint256 closeFactor, uint256 liqIncentive);
event TargetAdded(address target);
event SeriesAdded(address zero, address lpToken);
event SeriesQueued(address adapter, uint48 maturity, address
    ↪ pool);
```

3.215 CVF-215

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

Recommendation All parameters of these events should be indexed.

Listing 215:

```
event TargetAdded(address target);
event SeriesAdded(address zero, address lpToken);
event SeriesQueued(address adapter, uint48 maturity, address
    ↪ pool);
```

3.216 CVF-216

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

Recommendation The argument types should be more specific.

Listing 216:

```
    address _fuseDirectory,
    address _comptrollerImpl,
    address _cERC20Impl,
    address _divider,
    address _oracleImpl

    address fallbackOracle

function addTarget(address target, address adapter) external
    ↪ requiresTrust {

    address adapter,

    address pool

function addSeries(address adapter, uint48 maturity) external {
```

3.217 CVF-217

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

Recommendation The type of the "_controller" returned value should be more specific.

Listing 217:

```
126 ) external requiresTrust returns (uint256 _poolIndex, address
    ↪ _comptroller) {
```

3.218 CVF-218

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

Recommendation Zero here should be a named constant.

Listing 218:

```
148 require(err == 0, "Failed to become admin");

187 require(err == 0, "Failed to add market");

255 require(errZero == 0, "Failed to add Zero market");

275 require(errLpToken == 0, "Failed to add LP market");
```

3.219 CVF-219

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

Recommendation Enum constant would be more efficient that string literals.

Listing 219:

```
283 if (what == "ZERO_PARAMS") zeroParams = data;
    else if (what == "LP_TOKEN_PARAMS") lpTokenParams = data;
    else if (what == "TARGET_PARAMS") targetParams = data;
```

3.220 CVF-220

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

Description We didn't review these files.

Listing 220:

```
import { ERC20 } from "@rari-capital/solmate/src/erc20/SafeERC20
    ↪ .sol";
import { Trust } from "@rari-capital/solmate/src/auth/Trust.sol
    ↪ ";

import { FixedPointMathLib } from "@rari-capital/solmate/src/
    ↪ utils/FixedPointMathLib.sol";
import { BalancerVault } from "@sense-finance/v1-core/src/
    ↪ external/balancer/Vault.sol";
```

3.221 CVF-221

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

Description "These references look like external.
Recommendation In order to be internal they should be: ../../tokens/Tokens.sol ../../adapters/BaseAdapter.sol"

Listing 221:

```
import { Token } from "@sense-finance/v1-core/src/tokens/Token.
    ↪ sol";
import { BaseAdapter as Adapter } from "@sense-finance/v1-core/
    ↪ src/adapters/BaseAdapter.sol";
```

3.222 CVF-222

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

Recommendation This interface should be moved to a separate file named "BalancerOracleLike.sol".

Listing 222:

```
interface BalancerOracleLike {
```

3.223 CVF-223

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

Description The semantics of the returned value is unclear.
Recommendation Consider documenting.

Listing 223:

```
returns (uint256[] memory results);
```

3.224 CVF-224

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

Description The format of these returned values is unclear.
Recommendation Consider documented.

Listing 224:

```
int256 logPairPrice,
int256 accLogPairPrice,
int256 logBptPrice,
int256 accLogBptPrice,
int256 logInvariant,
int256 accLogInvariant,
```

3.225 CVF-225

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

Recommendation This contract should be in a file named "ZeroOracle.sol".

Listing 225:

```
contract ZeroOracle is PriceOracle, Trust {
```

3.226 CVF-226

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

Recommendation The key and value types for this mapping should be more specific.

Listing 226:

```
mapping(address => address) public pools;
```

3.227 CVF-227

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

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

Listing 227:

```
constructor() Trust(msg.sender) {}
```

3.228 CVF-228

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

Description This function should emit some event.

Listing 228:

```
function setZero(address zero, address pool) external
    ↪ requiresTrust {
```

3.229 CVF-229

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

Recommendation The type of the zero argument should be "Zero". The type of the "pool" argument should be "BalancerOracleLike".

Listing 229:

```
58 function setZero(address zero, address pool) external
    ↪ requiresTrust {

60 }

72 function _price(address zero) internal view returns (uint256) {
```

3.230 CVF-230

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

Description We didn't review these files.

Listing 230:

```
5 import { Trust } from "@rari-capital/solmate/src/auth/Trust.sol
    ↪ ";

7 import { FixedPointMathLib } from "@rari-capital/solmate/src/
    ↪ utils/FixedPointMathLib.sol";
```

3.231 CVF-231

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

Description These references look like external.
Recommendation In order to be internal they should be: ../../tokens/Tokens.sol ../../adapters/BaseAdapter.sol"

Listing 231:

```
10 import { Token } from "@sense-finance/v1-core/src/tokens/Token.
    ↪ sol";
   import { BaseAdapter as Adapter } from "@sense-finance/v1-core/
    ↪ src/adapters/BaseAdapter.sol";
```

3.232 CVF-232

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

Recommendation This contract should be in a file named "UnderlyingOracle.sol".

Listing 232:

```
contract UnderlyingOracle is PriceOracle, Trust {
```

3.233 CVF-233

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

Recommendation The key and value types should be more specific.

Listing 233:

```
mapping(address => address) public adapters;
```

3.234 CVF-234

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

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

Listing 234:

```
constructor() Trust(msg.sender) {}
```

3.235 CVF-235

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

Recommendation The type of the "underlying" argument should be "IERC20". The type of the "adapter" argument should be "Adapter".

Listing 235:

```
function setUnderlying(address underlying, address adapter) external requiresTrust {

function price(address underlying) external view override returns (uint256) {

function _price(address underlying) internal view returns (uint256) {
```

3.236 CVF-236

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

Description We didn't review these files.

Listing 236:

```
import { Trust } from "@rari-capital/solmate/src/auth/Trust.sol";

import { FixedPointMathLib } from "@rari-capital/solmate/src/utils/FixedPointMathLib.sol";
```

3.237 CVF-237

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

Description These references look like external.
Recommendation In order to be internal they should be: ../../tokens/Tokens.sol ../../adapters/BaseAdapter.sol

Listing 237:

```
import { Token } from "@sense-finance/v1-core/src/tokens/Token.
    ↪ sol";
import { BaseAdapter as Adapter } from "@sense-finance/v1-core/
    ↪ src/adapters/BaseAdapter.sol";
```

3.238 CVF-238

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

Recommendation This contract should be in a file named "TargetOracle.sol".

Listing 238:

```
contract TargetOracle is PriceOracle, Trust {
```

3.239 CVF-239

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

Recommendation The key and value types should be more specific.

Listing 239:

```
mapping(address => address) public adapters;
```

3.240 CVF-240

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

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

Listing 240:

```
constructor() Trust(msg.sender) {}
```

3.241 CVF-241

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

Recommendation The type of the "target" argument should be "IERC20", The type of the "adapter" argument should be "Adapter".

Listing 241:

```
function setTarget(address target, address adapter) external
    ↪ requiresTrust {

function price(address target) external view override returns (
    ↪ uint256) {

function _price(address target) internal view returns (uint256)
    ↪ {
```

3.242 CVF-242

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

Recommendation This function should emit some event.

Listing 242:

```
function setTarget(address target, address adapter) external
    ↪ requiresTrust {
```

3.243 CVF-243

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

Description We didn't review these files.

Listing 243:

```
import { ERC20 } from "@rari-capital/solmate/src/erc20/SafeERC20
    ↪ .sol";
import { Trust } from "@rari-capital/solmate/src/auth/Trust.sol
    ↪ ";

import { FixedPointMathLib } from "@rari-capital/solmate/src/
    ↪ utils/FixedPointMathLib.sol";
import { BalancerVault } from "@sense-finance/v1-core/src/
    ↪ external/balancer/Vault.sol";
```

3.244 CVF-244

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

Recommendation This interface should be moved to a separate file named "BalancerOracleLike.sol".

Listing 244:

```
interface BalancerOracleLike {
```

3.245 CVF-245

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

Description The semantics of the returned value is unclear.
Recommendation Consider documenting.

Listing 245:

```
returns (uint256[] memory results);
```

3.246 CVF-246

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

Description The format of these returned values is unclear.
Recommendation Consider documented.

Listing 246:

```
int256 logPairPrice,
int256 accLogPairPrice,
int256 logBptPrice,
int256 accLogBptPrice,
int256 logInvariant,
int256 accLogInvariant,
```

3.247 CVF-247

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

Recommendation This contract should be in a file named "LPOracle.sol".

Listing 247:

```
contract LPOracle is PriceOracle, Trust {
```

3.248 CVF-248

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

Recommendation The key and value types for this mapping should be more specific.

Listing 248:

```
mapping(address => address) public pools;
```

3.249 CVF-249

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

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

Listing 249:

```
constructor() Trust(msg.sender) {}
```

3.250 CVF-250

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

Recommendation This abstract contract should be turned into an interface.

Listing 250:

```
abstract contract PriceOracle {
```

3.251 CVF-251

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

Recommendation When turning the abstract contract into an interface, this public constant should be turned into the following function: function isPriceOracle() external view virtual returns (bool);

Listing 251:

```
bool public constant isPriceOracle = true;
```

3.252 CVF-252

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

Recommendation This interface should be moved to a separate file names "CTokenLike.sol".

Listing 252:

```
interface CTokenLike {
```

3.253 CVF-253

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

Description This is an abstract contract rather than an interface.
Recommendation Consider turning this abstract contract into an interface.

Listing 253:

```
abstract contract InterestRateModel {
```

3.254 CVF-254

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

Description When turning this abstract contract into an interface, this public variable should be turned into the following function: function isInterestRateModel () external view virtual return (bool);

Listing 254:

```
bool public constant isInterestRateModel = true;
```

3.255 CVF-255

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

Description We didn't review this file.

Listing 255:

```
import { SafeERC20, ERC20 } from "@rari-capital/solmate/src/
    ↪ erc20/SafeERC20.sol";
```

3.256 CVF-256

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

Description The semantics of the keys for these mappings is unclear.
Recommendation Consider documenting.

Listing 256:

```
22  mapping(address => uint256) public inits;

26  mapping(address => uint256) public mscales;
    mapping(address => Token) public gclaims;
```

3.257 CVF-257

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

Description All these mappings use Claim as a key.
Recommendation It would be more efficient to merge these mappings into a single mapping whose keys are Claims and values are structs encapsulating the values of the original mappings.

Listing 257:

```
22  mapping(address => uint256) public inits;

24  mapping(address => uint256) public totals;

26  mapping(address => uint256) public mscales;
    mapping(address => Token) public gclaims;
```

3.258 CVF-258

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

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

Listing 258:

```
28  address public divider;
```

3.259 CVF-259

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

Recommendation The type of the "_divider" argument should be "Divider".

Listing 259:

```
constructor(address _divider) {
```

3.260 CVF-260

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

Recommendation The type of the "adapter" arguments should be "Adapter".

Listing 260:

```
37 address adapter,

78 address adapter,

110 address adapter,
```

3.261 CVF-261

- **Severity** Moderate
- **Category** Flaw
- **Status** Fixed
- **Source** GClaimManager.sol

Description The "tBal" value is rounded down, i.e. towards the user. This could lead to a situation when the protocol will not be able to pay all the users in full. A good practice is ti always round towards the protocol, i.e. against a user.

Listing 261:

```
61 uint256 tBal = excess(adapter, maturity, uBal);

64     ERC20(Adapter(adapter).getTarget()).safeTransferFrom(msg.
    ↪ sender, address(this), tBal);
```

3.262 CVF-262

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

Description This call internally obtains the Claim address from the adapter, while this address was already obtained.
Recommendation Consider refactoring the code to not obtain the same address twice.

Listing 262:

```
uint256 tBal = excess(adapter, maturity, uBal);
```

3.263 CVF-263

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

Description The mscale value calcualted here is the maximum scale value across all the time moment when the "excess" function was called. The Divider may calculate the maximum scale at different time moment, so the maximum scale calculated here could differ from the maximum scale calculated in the Divider.

Listing 263:

```
117 uint256 mscale = mscales[claim];
    if (scale <= mscale) {
        scale = mscale;
120 } else {
        mscales[claim] = scale;
    }
```

3.264 CVF-264

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

Description This formula tries to emulate the Divider behavior, but doesn't take into account all the nuances. IT would be better to just collect, update the totals, and then divide the totals by the total supply.

Listing 264:

```
125 tBal = (uBal * scale) / (scale - initScale) / 10**18;
```

3.265 CVF-265

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

Description We didn't review this file.

Listing 265:

```
import { Trust } from "@rari-capital/solmate/src/auth/Trust.sol
  ↪ ";
```

3.266 CVF-266

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

Recommendation This internal reference should look like "../Divider.sol", otherwise it looks like an external reference.

Listing 266:

```
import { Divider } from "@sense-finance/v1-core/src/Divider.sol
  ↪ ";
```

3.267 CVF-267

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

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

Listing 267:

```
address public immutable divider;
```

3.268 CVF-268

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

Recommendation The type of the "divider" argument should be divider.

Listing 268:

```
constructor(address _divider) Trust(msg.sender) {
```

3.269 CVF-269

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

Description The "setPermissionless" function is called on the divider many times.
Recommendation Consider calling it only once.
Client Comment No longer there

Listing 269:

```
Divider(divider).setPermissionless(false);

    Divider(divider).setPermissionless(false);
```

3.270 CVF-270

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

Description In case the "adapters" array is empty, the function will emit no events, but still will set the divider in non-permissionless mode.
Recommendation Consider emitting some event event when the "adapters" array is empty.

Listing 270:

```
emit Stopped(adapters[i]);
```

3.271 CVF-271

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

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

Listing 271:

```
event Stopped(address indexed adapter);
```

3.272 CVF-272

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

Description There is no access level specified for these constants, so internal access will be used by default.
Recommendation Consider explicitly specifying an access level.

Listing 272:

```
string constant AlreadySettled = "Series has already been
    ↪ settled";
string constant CollectNotSettled = "Cannot collect if Series is
    ↪ at or after maturity and it has not been settled";
string constant Create2Failed = "ERC1167: create2 failed";
string constant DuplicateSeries = "Series has already been
    ↪ initialized";
string constant ExistingValue = "New value must be different
    ↪ than previous";
string constant FactoryNotSupported = "Factory is not supported
    ↪ ";
string constant FlashCallbackFailed = "FlashLender: Callback
    ↪ failed";
string constant FlashRepayFailed = "FlashLender: Repay failed";
string constant FlashTransferFailed = "FlashLender: Transfer
    ↪ failed";
string constant FlashUntrustedBorrower = "FlashBorrower:
    ↪ Untrusted lender";
string constant FlashUntrustedLoanInitiator = "FlashBorrower:
    ↪ Untrusted loan initiator";
string constant GuardCapReached = "Issuance cap reached";
string constant IssuanceFeeCapExceeded = "Issuance fee cannot
    ↪ exceed 10%";
string constant IssueOnSettled = "Cannot issue if Series is
    ↪ settled";
string constant InvalidAdapter = "Invalid adapter address or
    ↪ adapter is not enabled";
string constant InvalidMaturity = "Maturity date is not valid";
string constant InvalidMaturityOffsets = "Invalid maturity
    ↪ offsets";
string constant InvalidScaleValue = "Scale value is invalid";
string constant NotAuthorized = "UNTRUSTED"; // We copy the
    ↪ error message used by solmate's `Trust` auth lib
string constant NotEnoughClaims = "Not enough claims to collect
    ↪ given target balance";
string constant SeriesDoesntExists = "Series does not exist";
(... 30)
```

3.273 CVF-273

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

Recommendation Constants are usually named IN_UPPER_CASE.

Listing 273:

```
string constant AlreadySettled = "Series has already been
    ↪ settled";
string constant CollectNotSettled = "Cannot collect if Series is
    ↪ at or after maturity and it has not been settled";
string constant Create2Failed = "ERC1167: create2 failed";
string constant DuplicateSeries = "Series has already been
    ↪ initialized";
string constant ExistingValue = "New value must be different
    ↪ than previous";
string constant FactoryNotSupported = "Factory is not supported
    ↪ ";
string constant FlashCallbackFailed = "FlashLender: Callback
    ↪ failed";
string constant FlashRepayFailed = "FlashLender: Repay failed";
string constant FlashTransferFailed = "FlashLender: Transfer
    ↪ failed";
string constant FlashUntrustedBorrower = "FlashBorrower:
    ↪ Untrusted lender";
string constant FlashUntrustedLoanInitiator = "FlashBorrower:
    ↪ Untrusted loan initiator";
string constant GuardCapReached = "Issuance cap reached";
string constant IssuanceFeeCapExceeded = "Issuance fee cannot
    ↪ exceed 10%";
string constant IssueOnSettled = "Cannot issue if Series is
    ↪ settled";
string constant InvalidAdapter = "Invalid adapter address or
    ↪ adapter is not enabled";
string constant InvalidMaturity = "Maturity date is not valid";
string constant InvalidMaturityOffsets = "Invalid maturity
    ↪ offsets";
string constant InvalidScaleValue = "Scale value is invalid";
string constant NotAuthorized = "UNTRUSTED"; // We copy the
    ↪ error message used by solmate's `Trust` auth lib
string constant NotEnoughClaims = "Not enough claims to collect
    ↪ given target balance";
string constant SeriesDoesntExists = "Series does not exist";
string constant NotSettled = "Series must be settled";
(... 30)
```

3.274 CVF-274

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

Description Strings are inefficient.
Recommendation Consider using an enum.

Listing 274:

```
string constant AlreadySettled = "Series has already been
    ↪ settled";
string constant CollectNotSettled = "Cannot collect if Series is
    ↪ at or after maturity and it has not been settled";
string constant Create2Failed = "ERC1167: create2 failed";
string constant DuplicateSeries = "Series has already been
    ↪ initialized";
string constant ExistingValue = "New value must be different
    ↪ than previous";
string constant FactoryNotSupported = "Factory is not supported
    ↪ ";
string constant FlashCallbackFailed = "FlashLender: Callback
    ↪ failed";
string constant FlashRepayFailed = "FlashLender: Repay failed";
string constant FlashTransferFailed = "FlashLender: Transfer
    ↪ failed";
string constant FlashUntrustedBorrower = "FlashBorrower:
    ↪ Untrusted lender";
string constant FlashUntrustedLoanInitiator = "FlashBorrower:
    ↪ Untrusted loan initiator";
string constant GuardCapReached = "Issuance cap reached";
string constant IssuanceFeeCapExceeded = "Issuance fee cannot
    ↪ exceed 10%";
string constant IssueOnSettled = "Cannot issue if Series is
    ↪ settled";
string constant InvalidAdapter = "Invalid adapter address or
    ↪ adapter is not enabled";
string constant InvalidMaturity = "Maturity date is not valid";
string constant InvalidMaturityOffsets = "Invalid maturity
    ↪ offsets";
string constant InvalidScaleValue = "Scale value is invalid";
string constant NotAuthorized = "UNTRUSTED"; // We copy the
    ↪ error message used by solmate's `Trust` auth lib
string constant NotEnoughClaims = "Not enough claims to collect
    ↪ given target balance";
string constant SeriesDoesntExists = "Series does not exist";
string constant NotSettled = "Series must be settled";
(... 30)
```

3.275 CVF-275

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

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

Listing 275:

```
address public immutable divider;
```

3.276 CVF-276

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

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

Listing 276:

```
address public immutable adapter;
```

3.277 CVF-277

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

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

Listing 277:

```
address _divider,
```

3.278 CVF-278

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

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

Listing 278:

```
address _adapter,
```

3.279 CVF-279

- **Severity** Major
- **Category** Flaw
- **Status** Info
- **Source** Claim.sol

Description The "collect" call is executed even in case the "transfer" call returns false.
Recommendation Consider changing "return super.transfer(..)" into "require (super.transfer (...))".
Client Comment Solmate's ERC20 never returns false, so the issue is incorrect.

Listing 279:

```
Divider(divider).collect(msg.sender, adapter, maturity, value,
    ↪ to);
return super.transfer(to, value);
```

3.280 CVF-280

- **Severity** Major
- **Category** Flaw
- **Status** Info
- **Source** Claim.sol

Description The "collect" call is executed even in case the "transferFrom" call returns false.
Recommendation Consider changing "return super.transferFrom(..)" into "require (super.transferFrom (...))".
Client Comment Solmate's ERC20 never returns false, so the issue is incorrect.

Listing 280:

```
Divider(divider).collect(from, adapter, maturity, value, to);
return super.transferFrom(from, to, value);
```

3.281 CVF-281

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

Description We didn't review these files.

Listing 281:

```
import { ERC20 } from "@rari-capital/solmate/src/erc20/ERC20.sol
    ↪ ";
import { Trust } from "@rari-capital/solmate/src/auth/Trust.sol
    ↪ ";
```

3.282 CVF-282

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

Description We didn't review the "Trust" base contract.

Listing 282:

```
contract Token is ERC20, Trust {
```

3.283 CVF-283

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

Description This function allows the trusted address to burn tokens owned by anybody, effectively stealing tokens from them. More common approach would be to allow the trusted address to burn only its own tokens, or tokens explicitly approved to it.

Listing 283:

```
function burn(address usr, uint256 amount) public requiresTrust
    ↪ {
```

3.284 CVF-284

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

Recommendation This interface should be in a file named "BalancerPool.sol".

Listing 284:

```
interface BalancerPool {
```

3.285 CVF-285

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

Description The semantics and the format of the returned data is unclear.
Recommendation Consider adding a documentation comment.

Listing 285:

```
returns (uint256[] memory results);
```

3.286 CVF-286

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

Description The semantics of these fields is unclear.
Recommendation Consider adding documentation comments.

Listing 286:

```
Variable variable;
uint256 secs;
uint256 ago;
```

3.287 CVF-287

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

Description The formats of these number is unclear.
Recommendation Consider adding a documentation comment.

Listing 287:

```
int256 logPairPrice,
int256 accLogPairPrice,
int256 logBptPrice,
int256 accLogBptPrice,
int256 logInvariant,
int256 accLogInvariant,
```

3.288 CVF-288

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

Recommendation The return type could be more specific.

Listing 288:

```
function getVault() external view returns (address);
```

3.289 CVF-289

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

Description We didn't review this file.

Listing 289:

```
import { ERC20 } from "@rari-capital/solmate/src/erc20/SafeERC20
    ↪ .sol";
```

3.290 CVF-290

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

Recommendation This interface should be moved to a separate file named "IAsset.sol".

Listing 290:

```
interface IAsset {}
```

3.291 CVF-291

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

Recommendation This interface should be in a file named "BalancerVault.sol".

Listing 291:

```
interface BalancerVault {
```

3.292 CVF-292

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

Description The semantics of the first returned values is unclear.
Recommendation Consider giving it a descriptive name, adding a documentation comment, and/or making its type more specific.

Listing 292:

```
function getPool(bytes32 poolId) external view returns (address,
    ↪ PoolSpecialization);
```

3.293 CVF-293

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

Description The semantics of the returned value is unclear.
Recommendation Consider giving it a descriptive name and/or adding a documentation comment.

Listing 293:

```
) external payable returns (uint256);
```

3.294 CVF-294

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

Description Phantom overflow is possible here, i.e. a situation when the final calculation result would fit into the destination type, but some intermediate calculations overflow.
Recommendation Consider using the "muldiv" function described here: https://2π.com/21/muldiv/index.html or some tricks described here: https://medium.com/coinmonks/math-in-solidity-part-3-percents-and-proportions-4db014e080b1 or some other approaches that prevent phantom overflows.
Client Comment NOTE: We've tried adding "muldiv" but we've realised that, on one hand, it increased the gas cost and, on the other hand, we would need to understand how it works so as to also have the ‘mulUp‘ and ‘divUp‘. As this is marked as a "minor" issue by ADBK and considering also that "ds-math" (which is the same as what we are using) has been in production on projects like Maker and this overflow issue could happen on extreme large numbers (rare use case) and we've also seen that some similar projects (Pendle) are not implementing these tricks to be covered by phantom overflow, we've decided to put this on hold. Is this a good approach/reasoning?

Listing 294:

```
z = x * y;

z = x * y + baseUnit - 1; // Rounds up. So (again imagining 2
    ↪ decimal places):

z = (x * baseUnit) / y;

z = x * baseUnit + y; // 101 (1.01) / 1000 (10) -> (101 * 100 +
    ↪ 1000 - 1) / 1000 -> 11 (0.11 = 0.101 rounded up).
```

3.295 CVF-295

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

Recommendation It would be better to write: x * y + (baseUnit - 1) to prevent overflow is case x * y + baseUnit is exactly 2^256.

Listing 295:

```
27 z = x * y + baseUnit − 1; // Rounds up. So (again imagining 2
   ↪ decimal places):
```

3.296 CVF-296

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

Recommendation Consider calculating the rounded down result first, using some tricks that prevent phantom overflows, and then use the "mulmod" function to know whether the result ought to be incremented.
Client Comment Same as CVF 294

Listing 296:

```
27 z = x * y + baseUnit − 1; // Rounds up. So (again imagining 2
   ↪ decimal places):

48 z = x * baseUnit + y; // 101 (1.01) / 1000 (10) −> (101 * 100 +
   ↪ 1000 − 1) / 1000 −> 11 (0.11 = 0.101 rounded up).
```

3.297 CVF-297

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

Recommendation Brackets are redundant here.

Listing 297:

```
29     z /= (baseUnit);

40 z = (x * baseUnit) / y;
```