ABDK CONSULTING

VAULT SMART CONTRACT AUDIT

Gambit

Solidity

abdk.consulting

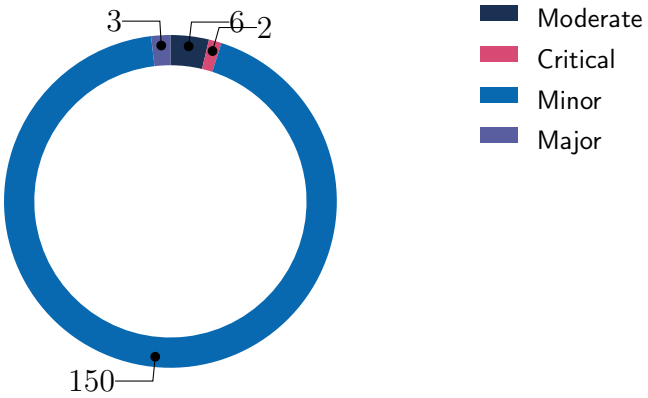
GAMBIT SMART CONTRACTS AUDIT CONCLUSION

by Mikhail Vladimirov and Dmitry Khovratovich
20th April 2021

We've been asked to review Gambit smart contracts in a GitHub repository. We have found several major issues and a few less important ones. The crucial ones are

- No access control for a critical function (CVF-42);
- Wrong argument passed (CVF-51);
- Incorrect whitelisting check (CVF-87);
- Function returning a constant (CVF-90);
- Too big loss incurred by users (CVF-130).

All those are easy to fix.



Findings

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

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

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

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

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

ID	Severity	Category	Status
CVF-148	Minor	Suboptimal	Opened
CVF-149	Minor	Procedural	Opened
CVF-150	Minor	Suboptimal	Opened
CVF-151	Minor	Bad naming	Opened
CVF-152	Minor	Suboptimal	Opened
CVF-153	Minor	Suboptimal	Opened
CVF-154	Minor	Unclear behavior	Opened
CVF-155	Minor	Suboptimal	Opened
CVF-156	Minor	Suboptimal	Opened
CVF-157	Minor	Suboptimal	Opened
CVF-158	Minor	Unclear behavior	Opened
CVF-159	Minor	Suboptimal	Opened
CVF-160	Minor	Procedural	Opened
CVF-161	Minor	Suboptimal	Opened

Contents

1 Document properties

Version

Version	Date	Author	Description
0.1	Apr. 20, 2021	D. Khovratovich	Initial Draft
0.2	Apr. 20, 2021	D. Khovratovich	Minor revision
1.0	Apr. 21, 2021	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 in the Gambit repository, commit 903531:

- Router.sol;
- Vault.sol;
- USDG.sol;
- YieldToken.sol.

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.

2.4 CVF-1

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** YieldToken.sol

Recommendation Should be "ô.6.0" according to a common best practice.

Listing 1:

```
3 solidity 0.6.12;
```

2.5 CVF-2

- **Severity** Minor
- **Status** Opened
- **Category** Procedural
- **Source** YieldToken.sol

Description We didn't review these files.

Listing 2:

```
5 "../libraries/math/SafeMath.sol";
  "../libraries/token/IERC20.sol";
  "../libraries/token/SafeERC20.sol";

9 "./interfaces/IYieldTracker.sol";
10 "./interfaces/IYieldToken.sol";
```

2.6 CVF-3

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** YieldToken.sol

Recommendation Turning these storage variables into constants would be make the contract more efficient.

Listing 3:

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

2.7 CVF-4

- **Severity** Minor
- **Status** Opened
- **Category** Bad naming
- **Source** YieldToken.sol

Recommendation In order to simplify the code, consider renaming this mapping to "balanceOf" and removing the "balanceOf" function.

Listing 4:

```
25 mapping (address => uint256) public balances;
```

2.8 CVF-5

- **Severity** Minor
- **Status** Opened
- **Category** Bad naming
- **Source** YieldToken.sol

Recommendation In order to simplify the code, consider renaming this mapping to "allowance" and removing the "allowance" function.

Listing 5:

```
25 mapping (address => uint256) public balances;
```

2.9 CVF-6

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** YieldToken.sol

Recommendation These mappings are public but there are also public getters for them. This seems redundant.

Listing 6:

```
25 mapping (address => uint256) public balances;
   mapping (address => mapping (address => uint256)) public
   ↪ allowances;
```

2.10 CVF-7

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** YieldToken.sol

Description The semantics of the keys in this mapping is unclear.
Recommendation Consider adding a documentation comment.

Listing 7:

```
26 mapping (address => mapping (address => uint256)) public
    ↪ allowances;
```

2.11 CVF-8

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** YieldToken.sol

Recommendation The type of this array should be "IYieldTracker[]".

Listing 8:

```
28 address[] public yieldTrackers;
```

2.12 CVF-9

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** YieldToken.sol

Recommendation If it is really necessary to make the name and the symbol customizable at the deployment time, consider making the corresponding storage variables immutable to save gas.

Listing 9:

```
36 constructor(string memory _name, string memory _symbol, uint256
    ↪ _initialSupply) public {
```

2.13 CVF-10

- **Severity** Minor
- **Status** Opened
- **Category** Unclear behavior
- **Source** YieldToken.sol

Recommendation This function should probably log an event.

Listing 10:

```
43 function setGov(address _gov) external onlyGov {

47 function setInfo(string memory _name, string memory _symbol)
    ↪ external onlyGov {
```

2.14 CVF-11

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** YieldToken.sol

Description Token meta information is usually considered immutable and some software does not recognize changes in it. Is it really necesary to have it mutable?

Listing 11:

```
47 function setInfo(string memory _name, string memory _symbol)
    ↪ external onlyGov {
```

2.15 CVF-12

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** YieldToken.sol

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

Listing 12:

```
65 function withdrawToken(address _token, address _account, uint256
    ↪  _amount) external onlyGov {
```

2.16 CVF-13

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** YieldToken.sol

Description Setting the whole list of yield trackers at once could consume lots of gas.
Recommendation Consider implementing some way to send the list of yield trackers to the contract in several transactions.

Listing 13:

```
69 function setYieldTrackers(address[] memory _yieldTrackers)
   ↪ external onlyGov {
```

2.17 CVF-14

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** YieldToken.sol

Description Claiming from all the trackers could consume lots of gas.
Recommendation Consider implementing some way split the claiming process across several transactions.

Listing 14:

```
74 for (uint256 i = 0; i < yieldTrackers.length; i++) {

81 for (uint256 i = 0; i < yieldTrackers.length; i++) {
```

2.18 CVF-15

- **Severity** Minor
- **Status** Opened
- **Category** Procedural
- **Source** YieldToken.sol

Description The code below this line looks like it is always executed, while actually it is executed only when 'nonStakingAccounts[_account]' is false.
Recommendation Consider putting the rest of the function explicitly into an "else" branch.

Listing 15:

```
98 }
```

2.19 CVF-16

- **Severity** Minor
- **Status** Opened
- **Category** Flaw
- **Source** YieldToken.sol

Description Here safe subtraction is used to enforce a business-level constraint, which is a bad practice.
Recommendation Consider explicitly checking that the allowance is sufficient.

Listing 16:

```
117 uint256 nextAllowance = allowances[_sender][msg.sender].sub(
    ↪ _amount, "YieldToken: transfer amount exceeds allowance");
```

2.20 CVF-17

- **Severity** Minor
- **Status** Opened
- **Category** Procedural
- **Source** YieldToken.sol

Description This emits the "Approval" event, which usually is not emitted on transfers.

Listing 17:

```
118 _approve(_sender, msg.sender, nextAllowance);
```

2.21 CVF-18

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** YieldToken.sol

Recommendation This check is redundant. It is anyway possible to mint to a dead address.

Listing 18:

```
124 require(_account != address(0), "YieldToken: mint to the zero
    ↪ address");
```

2.22 CVF-19

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** YieldToken.sol

Recommendation This check is redundant.

Listing 19:

```
139 require(_account != address(0), "YieldToken: burn from the zero
    ↪ address");
```

2.23 CVF-20

- **Severity** Minor
- **Status** Opened
- **Category** Flaw
- **Source** YieldToken.sol

Description Here safe subtraction is used to enforce a business-level constraint.
Recommendation Consider explicitly checking that the current balance is sufficient.

Listing 20:

```
144 balances[_account] = balances[_account].sub(_amount);
```

2.24 CVF-21

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** YieldToken.sol

Recommendation These checks are redundant.

Listing 21:

```
154 require(_sender != address(0), "YieldToken: transfer from the
    ↪ zero address");
    require(_recipient != address(0), "YieldToken: transfer to the
    ↪ zero address");
```

2.25 CVF-22

- **Severity** Minor
- **Status** Opened
- **Category** Flaw
- **Source** YieldToken.sol

Description Here safe subtraction is used to enforce a business-level constraint.
Recommendation Consider explicitly checking that the current balance is sufficient.

Listing 22:

```
160 balances[_sender] = balances[_sender].sub(_amount, "YieldToken:
    ↪ transfer amount exceeds balance");
```

2.26 CVF-23

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** YieldToken.sol

Recommendation It would be more efficient to combine these two writes into a single one.

Listing 23:

```
164 nonStakingSupply = nonStakingSupply.sub(_amount);

167 nonStakingSupply = nonStakingSupply.add(_amount);
```

2.27 CVF-24

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** YieldToken.sol

Recommendation These checks are redundant.

Listing 24:

```
174 require(_owner != address(0), "YieldToken: approve from the zero
    ↪  address");
    require(_spender != address(0), "YieldToken: approve to the zero
    ↪  address");
```

2.28 CVF-25

- **Severity** Minor
- **Status** Opened
- **Category** Procedural
- **Source** Router.sol

Recommendation SPDX license identifier should be in the first line of the file.

Listing 25:

```
2 SPDX-License-Identifier: MIT
```

2.29 CVF-26

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** Router.sol

Recommendation Should be "ô.6.0" according to a common best practice.

Listing 26:

```
4 solidity 0.6.12;
```

2.30 CVF-27

- **Severity** Minor
- **Status** Opened
- **Category** Procedural
- **Source** Router.sol

Description We didn't review these files.

Listing 27:

```
6 "../libraries/math/SafeMath.sol";
  "../libraries/token/IERC20.sol";
  "../libraries/token/SafeERC20.sol";
  "../libraries/utils/Address.sol";

11 "../tokens/interfaces/IWETH.sol";
```

2.31 CVF-28

- **Severity** Minor
- **Status** Opened
- **Category** Bad naming
- **Source** Router.sol

Recommendation Enum constants are usually named IN_UPPER_CASE.

Listing 28:

```
19 enum TriggerType { None, AboveTriggerPrice, BelowTriggerPrice,
     ↪ Trailing }
```

2.32 CVF-29

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** Router.sol

Recommendation The type of the "path" field should be "IERC20[]".

Listing 29:

```
23 address[] path;

32 address[] path;
```

2.33 CVF-30

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** Router.sol

Recommendation The type of this storage variable should be "IWETH".

Listing 30:

```
57 address public weth;
```

2.34 CVF-31

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** Router.sol

Recommendation The type of this storage variable should be "IUSDG".

Listing 31:

```
58 address public usdg;
```

2.35 CVF-32

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** Router.sol

Recommendation The type of this storage variable should be "IVault".

Listing 32:

```
59 address public vault;
```

2.36 CVF-33

- **Severity** Moderate
- **Status** Opened
- **Category** Unclear behavior
- **Source** Router.sol

Description There is no way to cancel a decrease position order.
Recommendation Consider adding a function for this purpose.

Listing 33:

```
63 mapping (bytes32 => DecreasePositionOrder) public
   ↪ decreasePositionOrders;
```

2.37 CVF-34

- **Severity** Minor
- **Status** Opened
- **Category** Documentation
- **Source** Router.sol

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

Listing 34:

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

2.38 CVF-35

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** Router.sol

Description There is not range check for the length of this array. It seems that only arrays of at least two elements do make sense here, maybe only lengths 2 or 3 are valid according to ‘_swap‘ function.
Recommendation Consider adding an explicit range check.

Listing 35:

```
74 address[] memory _path,
```

2.39 CVF-36

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** Router.sol

Description There is no range check for this parameters. It seems that only non-zero values do make sense.
Recommendation Consider adding an explicit range check.

Listing 36:

```
75 uint256 _amountIn,

110 uint256 _amountIn,
```

2.40 CVF-37

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** Router.sol

Description The "_sender" function is called twice in the same function.
Recommendation Consider calling once and reusing the returned value.

Listing 37:

```
81  bytes32 id = getId(_sender(), _nonce);

83      _sender(),

117 bytes32 id = getId(_sender(), _nonce);

119     _sender(),

161 bytes32 id = getId(_sender(), _nonce);

163     _sender(),
```

2.41 CVF-38

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** Router.sol

Description It is not checked that the order with such ID does exist.
Recommendation Consider adding an explicit check based on some order field that cannot be zero.

Listing 38:

```
93  SwapOrder memory order = swapOrders[_id];

131 IncreasePositionOrder memory order = increasePositionOrders[_id
    ↪ ];

179 DecreasePositionOrder memory order = decreasePositionOrders[_id
    ↪ ];
```

2.42 CVF-39

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** Router.sol

Description The expression "order.path[order.path.length - 1]" is calculated twice.
Recommendation Consider calculating once and reusing the value.

Listing 39:

```
97 IERC20(order.path[order.path.length - 1]).safeTransfer(msg.
   ↪ sender, order.relayerFee);
IERC20(order.path[order.path.length - 1]).safeTransfer(order.
   ↪ receiver, amountOut.sub(order.relayerFee));
```

2.43 CVF-40

- **Severity** Minor
- **Status** Opened
- **Category** Flaw
- **Source** Router.sol

Description SafeMath is used here to enforce a business-level constraint, which is a bad practice.
Recommendation Consider explicitly checking that the relayer fee doesn't exceed the output amount.

Listing 40:

```
98 IERC20(order.path[order.path.length - 1]).safeTransfer(order.
   ↪ receiver, amountOut.sub(order.relayerFee));
```

2.44 CVF-41

- **Severity** Moderate
- **Status** Opened
- **Category** Suboptimal
- **Source** Router.sol

Description Deleting the order at the end of the function makes the function vulnerable to reentrancy attacks. The order should be deleted right after being read from the storage.

Listing 41:

```
100 delete swapOrders[_id];

141 delete increasePositionOrders[_id];

208 delete increasePositionOrders[_id];
```

2.45 CVF-42

- **Severity** Critical
- **Status** Opened
- **Category** Suboptimal
- **Source** Router.sol

Description This function could be called by anyone. A malicious actor may use this function to delete swaps, either to prevent them from being executed or just to collect gas refund, that the attacker may use to finance his own transaction.
Recommendation Consider making this function callable only by the initiator of a swap, or protecting this function in some other way.

Listing 42:

```
103 function cancelSwap(bytes32 _id) external {
        delete swapOrders[_id];

144 function cancelIncreasePosition(bytes32 _id) external {
        delete increasePositionOrders[_id];
```

2.46 CVF-43

- **Severity** Minor
- **Status** Opened
- **Category** Flaw
- **Source** Router.sol

Description There is no range check for the length of this array. It seems that only arrays of at least one element do make sense here.
Recommendation Consider adding an explicit range check.

Listing 43:

```
108 address[] memory _path,
```

2.47 CVF-44

- **Severity** Moderate
- **Status** Opened
- **Category** Flaw
- **Source** Router.sol

Description In case the relayer fee exceeds the input amount, the user may spend more tokens then specified in the "amountIn" parameter.
Recommendation Consider adding an explicit check that the relayer fee doesn't exceed the input amount.

Listing 44:

```
132 IERC20(order.path[0]).safeTransferFrom(order.account, msg.sender
    ↪ , order.relayerFee);
```

2.48 CVF-45

- **Severity** Moderate
- **Status** Opened
- **Category** Flaw
- **Source** Router.sol

Description Passing zero as the value of "_minOut" parameter here makes this function vulnerable to front run attacks. In case of a significant price shift between storing and executing an increase position order, the user may get position increase that is smaller than expected.
Recommendation Consider adding an extra parameter for the minimum position increase.

Listing 45:

```
137 _swap(order.path, 0, vault);
```

2.49 CVF-46

- **Severity** Moderate
- **Status** Opened
- **Category** Unclear behavior
- **Source** Router.sol

Description It looks quite weird that in case the output amount is not enough to cover the relayer fee, no fee is subtracted from the output amount at all. It would be more logical in such case to send the whole output amount to the relayer as a fee, and then try to take the rest of the fee from the creator of the order.

Listing 46:

```
205 IERC20(order.collateralToken).safeTransfer(order.receiver,
    ↪ amountOut);
```

2.50 CVF-47

- **Severity** Minor
- **Status** Opened
- **Category** Bad datatype
- **Source** Router.sol

Description The type of the "_path" parameter should be "IERC20[] memory" rather than just "address[] memory".

Listing 47:

```
211 function swap(address[] memory _path, uint256 _amountIn, uint256
    ↪  _minOut, address _receiver) public {

216 function swapETHToTokens(address[] memory _path, uint256 _minOut
    ↪ , address _receiver) external payable {

222 function swapTokensToETH(address[] memory _path, uint256
    ↪ _amountIn, uint256 _minOut, address payable _receiver)
    ↪ external {

229 function increasePosition(address[] memory _path, address
    ↪ _indexToken, uint256 _amountIn, uint256 _minOut, uint256
    ↪ _sizeDelta, bool _isLong, uint256 _price) external {

237 function increasePositionETH(address[] memory _path, address
    ↪ _indexToken, uint256 _minOut, uint256 _sizeDelta, bool
    ↪ _isLong, uint256 _price) external payable {
```

2.51 CVF-48

- **Severity** Minor
- **Status** Opened
- **Category** Unclear behavior
- **Source** Router.sol

Recommendation These function should return the actual output amount.

Listing 48:

```
211 function swap(address[] memory _path, uint256 _amountIn, uint256
      ↪  _minOut, address _receiver) public {

216 function swapETHToTokens(address[] memory _path, uint256 _minOut
      ↪ , address _receiver) external payable {

222 function swapTokensToETH(address[] memory _path, uint256
      ↪ _amountIn, uint256 _minOut, address payable _receiver)
      ↪ external {

246 function decreasePosition(address _collateralToken, address
      ↪ _indexToken, uint256 _collateralDelta, uint256 _sizeDelta,
      ↪  bool _isLong, address _receiver, uint256 _price) external
      ↪  {

250 function decreasePositionETH(address _collateralToken, address
      ↪ _indexToken, uint256 _collateralDelta, uint256 _sizeDelta,
      ↪  bool _isLong, address payable _receiver, uint256 _price)
      ↪ external {
```

2.52 CVF-49

- **Severity** Minor
- **Status** Opened
- **Category** Flaw
- **Source** Router.sol

Description There are not range checks for the lengths of the "_path" arguments.
Recommendation Consider adding proper range checks.

Listing 49:

```
211 function swap(address[] memory _path, uint256 _amountIn, uint256
    ↪  _minOut, address _receiver) public {

216 function swapETHToTokens(address[] memory _path, uint256 _minOut
    ↪ , address _receiver) external payable {

222 function swapTokensToETH(address[] memory _path, uint256
    ↪ _amountIn, uint256 _minOut, address payable _receiver)
    ↪ external {

229 function increasePosition(address[] memory _path, address
    ↪ _indexToken, uint256 _amountIn, uint256 _minOut, uint256
    ↪ _sizeDelta, bool _isLong, uint256 _price) external {

237 function increasePositionETH(address[] memory _path, address
    ↪ _indexToken, uint256 _minOut, uint256 _sizeDelta, bool
    ↪ _isLong, uint256 _price) external payable {
```

2.53 CVF-50

- **Severity** Minor
- **Status** Opened
- **Category** Unclear behavior
- **Source** Router.sol

Recommendation These functions should accept an extra argument specifying the minimum allowed position increase. Also these functions should return the actual position increase.

Listing 50:

```
229 function increasePosition(address[] memory _path, address
    ↪ _indexToken, uint256 _amountIn, uint256 _minOut, uint256
    ↪ _sizeDelta, bool _isLong, uint256 _price) external {

237 function increasePositionETH(address[] memory _path, address
    ↪ _indexToken, uint256 _minOut, uint256 _sizeDelta, bool
    ↪ _isLong, uint256 _price) external payable {
```

2.54 CVF-51

- **Severity** Critical
- **Status** Opened
- **Category** Flaw
- **Source** Router.sol

Recommendation The "address(this)" should be passed to the "_decreasePosition" function instead of "_recevier".

Listing 51:

```
251 uint256 amountOut = _decreasePosition(_collateralToken,
    ↪ _indexToken, _collateralDelta, _sizeDelta, _isLong,
    ↪ _receiver, _price);
```

2.55 CVF-52

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** Router.sol

Recommendation The minimum output check could be performed by the caller. No need to pass the "_minOut" parameter to the "_swap" function.

Listing 52:

```
292 function _swap(address[] memory _path, uint256 _minOut, address
    ↪ _receiver) private returns (uint256) {
```

2.56 CVF-53

- **Severity** Minor
- **Status** Opened
- **Category** Procedural
- **Source** Router.sol

Description The code after these lines looks like it is always executed, while actually it is executed only when the corresponding condition is false.
Recommendation Consider using explicit "else" branches to make the code more readable.

Listing 53:

```
295 }

299 }
```

2.57 CVF-54

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** Router.sol

Recommendation Consider separately handling or explicitly forbidding the special case when '_tokenIn == _tokenOut'.

Listing 54:

```
307 if (_tokenOut == usdg) { // buyUSDG
```

2.58 CVF-55

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** Router.sol

Recommendation This check could be made by the caller, no need to pass the "_minOut" parameter to the "_valutSwap" function.

Listing 55:

```
315 require(amountOut >= _minOut, "Router: insufficient amountOut");
```

2.59 CVF-56

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** Router.sol

Description This function is redundant.
Recommendation Consider removing it.

Listing 56:

```
319 function _sender() private view returns (address) {
```

2.60 CVF-57

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** USDG.sol

Recommendation Should be "ĝ0.6.0" according to a common best practice.

Listing 57:

```
3 solidity 0.6.12;
```

2.61 CVF-58

- **Severity** Minor
- **Status** Opened
- **Category** Procedural
- **Source** USDG.sol

Description We didn't review this file.

Listing 58:

```
5 "./interfaces/IUSDG.sol";
```

2.62 CVF-59

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** USDG.sol

Recommendation As there could be many vaults, it would be more logical to either pass no vaults into the constructor at all and add all of them later, or pass an array of all existing vaults.

Listing 59:

```
17 constructor(address _vault) public YieldToken("USD Gambit", "
   ↪ USDG", 0) {
```

2.63 CVF-60

- **Severity** Minor
- **Status** Opened
- **Category** Suboptimal
- **Source** USDG.sol

Recommendation This function probably should log some event.

Listing 60:

```
21 function addVault(address _vault) external onlyGov {

25 function removeVault(address _vault) external onlyGov {
```

2.64 CVF-61

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

Recommendation Should be "ô.6.0" according to a common best practice.

Listing 61:

```
3 solidity 0.6.12;
```

2.65 CVF-62

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

Description We didn't review these files.

Listing 62:

```
5 "../libraries/math/SafeMath.sol";
  "../libraries/token/IERC20.sol";
  "../libraries/token/SafeERC20.sol";
  "../libraries/utils/ReentrancyGuard.sol";

10 "../oracle/interfaces/IPriceFeed.sol";
```

2.66 CVF-63

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

Description The logic of this contract uses "decimals" property of tokens. This is a bad practice that makes the contract more complicated.
Recommendation Consider ignoring decimals and do all the calculations in basic units, rather then in full tokens.

Listing 63:

```
15 Vault is ReentrancyGuard, IVault {
```

2.67 CVF-64

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

Description The meaning of these fields is unclear, and especially unclear are their measurement units.
Recommendation Consider adding documentation comments.

Listing 64:

```
20 uint256 size;
   uint256 collateral;
   uint256 averagePrice;
   uint256 entryFundingRate;
   uint256 reserveAmount;
   int256 realisedPnl;
```

2.68 CVF-65

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

Description No access level specified for these constants, so the internal access level will be used by default.
Recommendation Consider explicitly specifying an access level.

Listing 65:

```
28 uint256 constant BASIS_POINTS_DIVISOR = 10000;
   uint256 constant FUNDING_RATE_PRECISION = 1000000;
30 uint256 constant PRICE_PRECISION = 10 ** 30;
   uint256 constant MIN_LEVERAGE = 10000; // 1x
   uint256 constant USDG_DECIMALS = 18;
   uint256 constant MAX_FEE_BASIS_POINTS = 500; // 5%
   uint256 constant MAX_LIQUIDATION_FEE_USD = 100 * PRICE_PRECISION
     ↪ ; // 100 USD
   uint256 constant MIN_FUNDING_RATE_INTERVAL = 1 hours;
   uint256 constant MAX_FUNDING_RATE_FACTOR = 10000; // 1%
```

2.69 CVF-66

- **Severity** Minor
- **Status** Opened
- **Category** Bad naming
- **Source** Vault.sol

Recommendation "1e30" would be more readable.

Listing 66:

```
30 uint256 constant PRICE_PRECISION = 10 ** 30;
```

2.70 CVF-67

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

Recommendation The type of this storage variable should be "Router".

Listing 67:

```
40 address public router;
```

2.71 CVF-68

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

Recommendation The type of this storage variable should be "IUSDG".

Listing 68:

```
42 address public usdg;
```

2.72 CVF-69

- **Severity** Minor
- **Status** Opened
- **Category** Bad naming
- **Source** Vault.sol

Description In identifiers, the acronyms "USD" and "USDG" are used either in ALL_CAPITALS or in CamelCase.
Recommendation Consider using consistent style across the code.

Listing 69:

```
45 uint256 public maxUsdg;

49 uint256 public liquidationFeeUsd;

84 mapping (address => uint256) public guaranteedUsd;

93 event BuyUSDG(address token, uint256 tokenAmount, uint256
   ↪ usdgAmount);
   event SellUSDG(address token, uint256 usdgAmount, uint256
   ↪ tokenAmount);

145 event IncreaseGuaranteedUsd(address token, uint256 amount);
    event DecreaseGuaranteedUsd(address token, uint256 amount);
```

2.73 CVF-70

- **Severity** Minor
- **Status** Opened
- **Category** Bad naming
- **Source** Vault.sol

Recommendation 50e4' would be more readable.

Listing 70:

```
46 uint256 public maxLeverage = 50 * 10000; // 50x
```

2.74 CVF-71

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

Description The meaning of the keys in these mapping is unclear.
Recommendation Consider adding a documentation comment.

Listing 71:

```
56 mapping (address => mapping (address => bool)) public
     ↪ approvedRouters;

58 mapping (address => bool) public whitelistedTokens;
   mapping (address => address) public priceFeeds;
60 mapping (address => uint256) public priceDecimals;
   mapping (address => uint256) public tokenDecimals;
   mapping (address => uint256) public redemptionBasisPoints;
   mapping (address => uint256) public minProfitBasisPoints;
   mapping (address => bool) public stableTokens;

67 mapping (address => uint256) public tokenBalances;

74 mapping (address => uint256) public override poolAmounts;

77 mapping (address => uint256) public override reservedAmounts;

84 mapping (address => uint256) public guaranteedUsd;

86 mapping (address => uint256) public cumulativeFundingRates;
   mapping (address => uint256) public lastFundingTimes;

89 mapping (bytes32 => Position) public positions;

91 mapping (address => uint256) public feeReserves;
```

2.75 CVF-72

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

Recommendation The key type for these mappings should be "IERC20".

Listing 72:

```
58 mapping (address => bool) public whitelistedTokens;
   mapping (address => address) public priceFeeds;
60 mapping (address => uint256) public priceDecimals;
   mapping (address => uint256) public tokenDecimals;
   mapping (address => uint256) public redemptionBasisPoints;
   mapping (address => uint256) public minProfitBasisPoints;
   mapping (address => bool) public stableTokens;
```

2.76 CVF-73

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

Recommendation It would be more efficient to have a single mapping whose keys are token contract addresses and values are structs of all the token-related fields.

Listing 73:

```
58  mapping (address => bool) public whitelistedTokens;
    mapping (address => address) public priceFeeds;
60  mapping (address => uint256) public priceDecimals;
    mapping (address => uint256) public tokenDecimals;
    mapping (address => uint256) public redemptionBasisPoints;
    mapping (address => uint256) public minProfitBasisPoints;
    mapping (address => bool) public stableTokens;

67  mapping (address => uint256) public tokenBalances;

70  mapping (address => uint256) public override usdgAmounts;

74  mapping (address => uint256) public override poolAmounts;

77  mapping (address => uint256) public override reservedAmounts;

84  mapping (address => uint256) public guaranteedUsd;

86  mapping (address => uint256) public cumulativeFundingRates;
    mapping (address => uint256) public lastFundingTimes;

91  mapping (address => uint256) public feeReserves;
```

2.77 CVF-74

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

Recommendation The key types for these mappings should be "IERC20".

Listing 74:

```
58 mapping (address => bool) public whitelistedTokens;
   mapping (address => address) public priceFeeds;
60 mapping (address => uint256) public priceDecimals;
   mapping (address => uint256) public tokenDecimals;
   mapping (address => uint256) public redemptionBasisPoints;
   mapping (address => uint256) public minProfitBasisPoints;
   mapping (address => bool) public stableTokens;

67 mapping (address => uint256) public tokenBalances;

70 mapping (address => uint256) public override usdgAmounts;

74 mapping (address => uint256) public override poolAmounts;

77 mapping (address => uint256) public override reservedAmounts;

84 mapping (address => uint256) public guaranteedUsd;

86 mapping (address => uint256) public cumulativeFundingRates;
   mapping (address => uint256) public lastFundingTimes;

91 mapping (address => uint256) public feeReserves;
```

2.78 CVF-75

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

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

Listing 75:

```
59 mapping (address => address) public priceFeeds;
```

2.79 CVF-76

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

Recommendation The types of all token parameters should be "IERC20".

Listing 76:

```
 93 event BuyUSDG(address token, uint256 tokenAmount, uint256
      ↪ usdgAmount);
    event SellUSDG(address token, uint256 usdgAmount, uint256
      ↪ tokenAmount);
    event Swap(address tokenIn, address tokenOut, uint256 amountIn,
      ↪ uint256 amountOut);

100     address collateralToken,
        address indexToken,

108     address collateralToken,
        address indexToken,

117     address collateralToken,
        address indexToken,

133 event UpdateFundingRate(address token, uint256 fundingRate);

136 event CollectSwapFees(address token, uint256 feeAmount);
    event CollectMarginFees(address token, uint256 feeUsd, uint256
      ↪ feeTokens);

139 event IncreasePoolAmount(address token, uint256 amount);
140 event DecreasePoolAmount(address token, uint256 amount);
    event IncreaseUsdgAmount(address token, uint256 amount);
    event DecreaseUsdgAmount(address token, uint256 amount);
    event IncreaseReservedAmount(address token, uint256 amount);
    event DecreaseReservedAmount(address token, uint256 amount);
    event IncreaseGuaranteedUsd(address token, uint256 amount);
    event DecreaseGuaranteedUsd(address token, uint256 amount);
```

2.80 CVF-77

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

Recommendation Token and account parameters should be indexed.

Listing 77:

```
 93 event BuyUSDG(address token, uint256 tokenAmount, uint256
      ↪ usdgAmount);
    event SellUSDG(address token, uint256 usdgAmount, uint256
      ↪ tokenAmount);
    event Swap(address tokenIn, address tokenOut, uint256 amountIn,
      ↪ uint256 amountOut);

 99     address account,
100     address collateralToken,
        address indexToken,

107     address account,
        address collateralToken,
        address indexToken,

116     address account,
        address collateralToken,
        address indexToken,

133 event UpdateFundingRate(address token, uint256 fundingRate);

136 event CollectSwapFees(address token, uint256 feeAmount);
    event CollectMarginFees(address token, uint256 feeUsd, uint256
      ↪ feeTokens);

139 event IncreasePoolAmount(address token, uint256 amount);
140 event DecreasePoolAmount(address token, uint256 amount);
    event IncreaseUsdgAmount(address token, uint256 amount);
    event DecreaseUsdgAmount(address token, uint256 amount);
    event IncreaseReservedAmount(address token, uint256 amount);
    event DecreaseReservedAmount(address token, uint256 amount);
    event IncreaseGuaranteedUsd(address token, uint256 amount);
    event DecreaseGuaranteedUsd(address token, uint256 amount);
```

2.81 CVF-78

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

Recommendation The type of this parameter should be "Router".

Listing 78:

```
160 address _router,
```

2.82 CVF-79

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

Recommendation The type of this parameters should be "IUSDG".

Listing 79:

```
161 address _usdg,
```

2.83 CVF-80

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

Description There are no range checks for these parameters, while not all the possible values seem to be valid.
Recommendation Consider adding range checks.

Listing 80:

```
162 uint256 _maxUsdg,
    uint256 _liquidationFeeUsd,
    uint256 _fundingRateFactor
```

2.84 CVF-81

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

Recommendation These functions should probably log some events.

Listing 81:

```
180 function setMaxUsdg(uint256 _maxUsdg) external nonReentrant
      ↪ onlyGov {

184 function setMaxLeverage(uint256 _maxLeverage) external
      ↪ nonReentrant onlyGov {

189 function setPriceSampleSpace(uint256 _priceSampleSpace) external
      ↪  nonReentrant onlyGov {

194 function setFees(uint256 _liquidationFeeUsd, uint256
      ↪ _swapFeeBasisPoints, uint256 _marginFeeBasisPoints)
      ↪ external nonReentrant onlyGov {

203 function setFundingRate(uint256 _fundingInterval, uint256
      ↪ _fundingRateFactor) external nonReentrant onlyGov {

210 function setTokenConfig(

231 function clearTokenConfig(address _token) external nonReentrant
      ↪ onlyGov {
```

2.85 CVF-82

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

Recommendation Should probably be ">=" here.

Listing 82:

```
185 require(_maxLeverage > MIN_LEVERAGE, "Vault: invalid
      ↪ _maxLeverage");
```

2.86 CVF-83

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

Recommendation Should probably be ">=" here.

Listing 83:

```
204 require(_fundingInterval > MIN_FUNDING_RATE_INTERVAL, "Vault:
    ↪ invalid _fundingInterval");
```

2.87 CVF-84

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

Recommendation The type of the "_token" arguments should be "IERC20".

Listing 84:

```
211     address _token,

231 function clearTokenConfig(address _token) external nonReentrant
    ↪ onlyGov {

242 function withdrawFees(address _token, address _receiver)
    ↪ external nonReentrant onlyGov returns (uint256) {

257 function buyUSDG(address _token, address _receiver) external
    ↪ override nonReentrant returns (uint256) {

283 function sellUSDG(address _token, address _receiver) external
    ↪ override nonReentrant returns (uint256) {

314 function swap(address _tokenIn, address _tokenOut, address
    ↪ _receiver) external override nonReentrant returns (uint256
    ↪ ) {

524 function getMaxPrice(address _token) public override view
    ↪ returns (uint256) {

528 function getMinPrice(address _token) public override view
    ↪ returns (uint256) {

532 function getSinglePrice(address _token) public override view
    ↪ returns (uint256) {

538 function getRoundId(address _token) public override view returns
    ↪  (uint256) {

544 function getRoundPrice(address _token, uint256 _roundId) public
    ↪ override view returns (uint256) {

552 function getPrice(address _token, bool _maximise) public view
    ↪ returns (uint256) {
(...)
979 function _decreaseGuaranteedUsd(address _token, uint256
    ↪ _usdAmount) private {
```

2.88 CVF-85

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

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

Listing 85:

```
212 address _priceFeed,
```

2.89 CVF-86

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

Description There are no range checks for these parameters, while not all the possible values seem to be valid.
Recommendation Consider adding explicit range checks.

Listing 86:

```
213 uint256 _priceDecimals,
    uint256 _tokenDecimals,
    uint256 _redemptionBps,
    uint256 _minProfitBps,
```

2.90 CVF-87

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

Description It is not checked whether the token is already whitelisted.
Recommendation Consider forbidding to configure tokens that are already whitelisted.

Listing 87:

```
219 whitelistedTokens[_token] = true;
```

2.91 CVF-88

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

Description Overflow is possible here, neither token decimals nor price decimals are limited.
Recommendation Consider limiting decimals values and/or using a safe power implementation.

Listing 88:

```
222 tokenDecimals[_token] = _tokenDecimals;

587 return 10 ** decimals;

640 return _amount.mul(10 ** decimalsMul).div(10 ** decimalsDiv);

652 return _tokenAmount.mul(price).div(10 ** decimals);

659 return _tokenAmount.mul(price).div(10 ** decimals);

675 return _usdAmount.mul(10 ** decimals).div(_price);
```

2.92 CVF-89

- **Severity** Minor
- **Status** Opened
- **Category** Flaw
- **Source** Vault.sol

Description This function deletes the configuration for a token, but doesn't delete the dynamic state for the token stored in the "tokenBalances", "usdgAmounts", "poolAmounts", reservedAmounts", guaranteedUsd", "cumulativeFunctionRates", "lastFunctingTimes", and "feeReserves" mappings. Probably not an issue.

Listing 89:

```
231 function clearTokenConfig(address _token) external nonReentrant
    ↪ onlyGov {
```

2.93 CVF-90

- **Severity** Major
- **Status** Opened
- **Category** Unclear behavior
- **Source** Vault.sol

Description This function always returns zero. Probably it should returns the amount withdrawn.

Listing 90:

```
242 function withdrawFees(address _token, address _receiver)
    ↪ external nonReentrant onlyGov returns (uint256) {
```

2.94 CVF-91

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

Description The code after this line looks like it is always executed, while actually it is executed only when amount != 0.
Recommendation Consider explicitly putting the rest of the function into an "else" branch.

Listing 91:

```
244 if(amount == 0) { return 0; }
```

2.95 CVF-92

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

Description This function is executed even if the router has been already added.

Listing 92:

```
249 function addRouter(address _router) external {
```

2.96 CVF-93

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

Description This function is executed even if the router has been already removed.

Listing 93:

```
253 function removeRouter(address _router) external {
```

2.97 CVF-94

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

Description A phantom overflow is possible here, i.e. a situation when the final output would fit into the destination type, while some intermediary calculations overflow.
Recommendation Consider using a muldiv implementation that is resistant to phantom overflows.

Listing 94:

```
268 uint256 usdgAmount = amountAfterFees.mul(price).div(
      ↪ PRICE_PRECISION);

326 uint256 amountOut = amountIn.mul(priceIn).div(priceOut);

331 uint256 usdAmount = amountIn.mul(priceIn).div(PRICE_PRECISION);

413 uint256 reserveDelta = position.reserveAmount.mul(_sizeDelta).
      ↪ div(position.size);

582 return price.mul(PRICE_PRECISION).div(getPricePrecision(_token))
      ↪ ;

593 uint256 redemptionAmount = _usdgAmount.mul(PRICE_PRECISION).div(
      ↪ price);

611 uint256 cappedAmount = _usdgAmount.mul(redemptionCollateral).div
      ↪ (totalUsdgAmount);

613 cappedAmount = cappedAmount.mul(basisPoints).div(
      ↪ BASIS_POINTS_DIVISOR);

640 return _amount.mul(10 ** decimalsMul).div(10 ** decimalsDiv);

652 return _tokenAmount.mul(price).div(10 ** decimals);

659 return _tokenAmount.mul(price).div(10 ** decimals);

675 return _usdAmount.mul(10 ** decimals).div(_price);

718 return fundingRateFactor.mul(reservedAmounts[_token]).mul(
      ↪ intervals).div(poolAmount);

    (...)

893 uint256 afterFeeAmount = _amount.mul(BASIS_POINTS_DIVISOR.sub(
      ↪ swapFeeBasisPoints)).div(BASIS_POINTS_DIVISOR);
```

2.98 CVF-95

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

Description The token parameters should have type "IERC20".

Listing 95:

```
314 function swap(address _tokenIn, address _tokenOut, address
    ↪ _receiver) external override nonReentrant returns (uint256
    ↪ ) {

348 function increasePosition(address _account, address
    ↪ _collateralToken, address _indexToken, uint256 _sizeDelta,
    ↪  bool _isLong) external override nonReentrant {

400 function decreasePosition(address _account, address
    ↪ _collateralToken, address _indexToken, uint256
    ↪ _collateralDelta, uint256 _sizeDelta, bool _isLong,
    ↪ address _receiver) external override nonReentrant returns
    ↪ (uint256) {

454 function liquidatePosition(address _account, address
    ↪ _collateralToken, address _indexToken, bool _isLong,
    ↪ address _feeReceiver) external nonReentrant {

487 function validateLiquidation(address _account, address
    ↪ _collateralToken, address _indexToken, bool _isLong, bool
    ↪ _raise) public view returns (bool, uint256) {

678 function getPosition(address _account, address _collateralToken,
    ↪  address _indexToken, bool _isLong) public override view
    ↪ returns (uint256, uint256, uint256, uint256, uint256,
    ↪ uint256, bool) {

685 function getPositionKey(address _account, address
    ↪ _collateralToken, address _indexToken, bool _isLong)
    ↪ public pure returns (bytes32) {

728 function getPositionLeverage(address _account, address
    ↪ _collateralToken, address _indexToken, bool _isLong)
    ↪ public view returns (uint256) {
(...)
788 function _reduceCollateral(address _account, address
    ↪ _collateralToken, address _indexToken, uint256
    ↪ _collateralDelta, uint256 _sizeDelta, bool _isLong)
    ↪ private returns (uint256, uint256) {
```

2.99 CVF-96

- **Severity** Moderate
- **Status** Opened
- **Category** Unclear behavior
- **Source** Vault.sol

Description Here "usdAmount" is calculated from amountId whose number of decimals matches _tokenIn. So adjustForDecimals here should use "_tokenIn" rather than "_tokenOut".

Listing 96:

```
333 uint256 usdOut = adjustForDecimals(usdAmount, _tokenOut, usdg);
```

2.100 CVF-97

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

Description There should be a function that returns a position by position key components. Currently this logic is repeated many times in the code.

Listing 97:

```
353 bytes32 key = getPositionKey(_account, _collateralToken,
      ↪ _indexToken, _isLong);
    Position storage position = positions[key];

405 bytes32 key = getPositionKey(_account, _collateralToken,
      ↪ _indexToken, _isLong);
    Position storage position = positions[key];

458 bytes32 key = getPositionKey(_account, _collateralToken,
      ↪ _indexToken, _isLong);
    Position memory position = positions[key];

488 bytes32 key = getPositionKey(_account, _collateralToken,
      ↪ _indexToken, _isLong);
    Position memory position = positions[key];

679 bytes32 key = getPositionKey(_account, _collateralToken,
      ↪ _indexToken, _isLong);
680 Position memory position = positions[key];

729 bytes32 key = getPositionKey(_account, _collateralToken,
      ↪ _indexToken, _isLong);
730 Position memory position = positions[key];

743 bytes32 key = getPositionKey(_account, _collateralToken,
      ↪ _indexToken, _isLong);
    Position memory position = positions[key];

789 bytes32 key = getPositionKey(_account, _collateralToken,
      ↪ _indexToken, _isLong);
790 Position storage position = positions[key];
```

2.101 CVF-98

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

Recommendation Should be "} else if (...".

Listing 98:

```
360 }

362 if (position.size > 0 && _sizeDelta > 0) {
```

2.102 CVF-99

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

Description A pair of increase+decrease guaranteed USD calls could be merged into a single call to either increase or decrease.

Listing 99:

```
428 _increaseGuaranteedUsd(_collateralToken, collateral.sub(position
    ↪ .collateral));
    _decreaseGuaranteedUsd(_collateralToken, _sizeDelta);

433 _increaseGuaranteedUsd(_collateralToken, collateral);
    _decreaseGuaranteedUsd(_collateralToken, _sizeDelta);
```

2.103 CVF-100

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

Description The code below looks like it is always executed, while it is actually executed only when 'usdOut' is zero.
Recommendation Consider putting the rest of the function explicitly into an "else" branch.

Listing 100:

```
449 }
```

2.104 CVF-101

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

Description These two calls could be merged into one call that increases/decreases the pool amount by the net pool amount change.

Listing 101:

```
478     _increasePoolAmount(_collateralToken, usdToTokenMin(
            ↪ _collateralToken, remainingCollateral));

483 _decreasePoolAmount(_collateralToken, usdToTokenMin(
        ↪ _collateralToken, liquidationFeeUsd));
```

2.105 CVF-102

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

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

Listing 102:

```
487 function validateLiquidation(address _account, address
        ↪ _collateralToken, address _indexToken, bool _isLong, bool
        ↪ _raise) public view returns (bool, uint256) {
```

2.106 CVF-103

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

Description The code below looks like it is always executed, while it is actually executed only when the hasProfit flag is true or position 'collateral >= delta'.
Recommendation Consider putting the rest of the function explicitly into an "else" branch.

Listing 103:

```
498 }
```

2.107 CVF-104

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

Description Should the delta to be added to the collateral in case the "hasProfit" flag is true?

Listing 104:

```
503 }
```

2.108 CVF-105

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

Recommendation It should be "<=" here as the type(uint80).max" value is a valid value for the uint80 type.

Listing 105:

```
547 require(_roundId < type(uint80).max, "Vault: invalid _roundId");
```

2.109 CVF-106

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

Recommendation This variable is redundant. Just assign the value to the "priceFeed" variable.

Listing 106:

```
553 address priceFeedAddress = priceFeeds[_token];
```

2.110 CVF-107

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

Description These two lines are equivalent to: if (roundId <= i) break; as in case roundId - i == 0, at teh next round roundId < i will be true and the loop execution will be terminated.

Listing 107:

```
561 if (roundId < i) { break; }
    if (roundId - i == 0) { continue; }
```

2.111 CVF-108

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

Description The code below looks like it is always executed, while it is actually executed only when price != 0.
Recommendation Consider putting the resot of the loop body explicitly into an "else" branch.

Listing 108:

```
569 }
```

2.112 CVF-109

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

Description It would be more efficient to store the 10ĉdecimals value instead of just decimals.

Listing 109:

```
587 return 10 ** decimals;

640 return _amount.mul(10 ** decimalsMul).div(10 ** decimalsDiv);

652 return _tokenAmount.mul(price).div(10 ** decimals);

659 return _tokenAmount.mul(price).div(10 ** decimals);

675 return _usdAmount.mul(10 ** decimals).div(_price);
```

2.113 CVF-110

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

Description The code below looks it is always executed, while actually it is executed only when the token is not stable.
Recommendation Consider putting the rest of the function explicitly into an "else" branch.

Listing 110:

```
597 }
```

2.114 CVF-111

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

Description The code below looks it is always executed, while actually it is executed only when 'redemptionCollateral' is not zero.
Recommendation Consider putting the rest of the function explicitly into an "else" branch.

Listing 111:

```
600 if (redemptionCollateral == 0) { return 0; }
```

2.115 CVF-112

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

Description The code below looks it is always executed, while actually it is executed only when totalUsdgAmount is not zero.
Recommendation Consider putting the rest of the function explicitly into an "else" branch.

Listing 112:

```
607 }
```

2.116 CVF-113

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

Description The intermediary capped amount value, that is already not precise due to division, is then passed to the "adjustForDecimals" function where it could be multiplied, thus amplifying any possible errors.
Recommendation Consider refactoring to make the division the very last operation.

Listing 113:

```
613 cappedAmount = cappedAmount.mul(basisPoints).div(
    ↪ BASIS_POINTS_DIVISOR);
cappedAmount = adjustForDecimals(cappedAmount, usdg, _token);
```

2.117 CVF-114

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

Description The code below looks it is always executed, while actually it is executed only when the token is not stable.
Recommendation Consider putting the rest of the function explicitly into an "else" branch.

Listing 114:

```
622 }
```

2.118 CVF-115

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

Recommendation This could be implemented in a more reliable way like this: if (decimalsMul > decimalsDiv) return _amount.mul (10 ** (decimalsMul - decimalsDiv)); else return _amount.div (10 ** (decimalsDiv - decimalsMul));.

Listing 115:

```
640 return _amount.mul(10 ** decimalsMul).div(10 ** decimalsDiv);
```

2.119 CVF-116

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

Recommendation These conditional statements are redundant. They optimize rare cases when "_tokenAmount" is zero, but make all the other cases more expensive.

Listing 116:

```
649 if (_tokenAmount == 0) { return 0; }

656 if (_tokenAmount == 0) { return 0; }
```

2.120 CVF-117

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

Recommendation These conditional statements are redundant. They optimize rare cases when "_usdAmount is zero", but make all the other cases more expensive.

Listing 117:

```
663 if (_usdAmount == 0) { return 0; }

668 if (_usdAmount == 0) { return 0; }

673 if (_usdAmount == 0) { return 0; }
```

2.121 CVF-118

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

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

Listing 118:

```
678 function getPosition(address _account, address _collateralToken,
    ↪  address _indexToken, bool _isLong) public override view
    ↪ returns (uint256, uint256, uint256, uint256, uint256,
    ↪ uint256, bool) {
```

2.122 CVF-119

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

Description The absolute value and the sign of the realized PNL are returned separately.
Recommendation Consider returning as a single signed value.

Listing 119:

```
682 return (position.size, position.collateral, position.
    ↪ averagePrice, position.entryFundingRate, position.
    ↪ reserveAmount, realisedPnl, position.realisedPnl >= 0);
```

2.123 CVF-120

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

Description The value "lastFundingTimes[_token]" is calculated three times.
Recommendation Consider calculating once and reusing.

Listing 120:

```
695 if (lastFundingTimes[_token] == 0) {

700 if (lastFundingTimes[_token].add(fundingInterval) > block.
    ↪ timestamp) {

706 lastFundingTimes[_token] = block.timestamp.div(fundingInterval).
    ↪ mul(fundingInterval);
```

2.124 CVF-121

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

Recommendation This could be calculated in a more efficient way as: block.timestamp - block.timestamp % fundingInterval.

Listing 121:

```
696     lastFundingTimes[_token] = block.timestamp.div(
        ↪ fundingInterval).mul(fundingInterval);

706 lastFundingTimes[_token] = block.timestamp.div(fundingInterval).
    ↪ mul(fundingInterval);
```

2.125 CVF-122

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

Recommendation Should be "} else if (" for readability.

Listing 122:

```
698 }

700 if (lastFundingTimes[_token].add(fundingInterval) > block.
    ↪ timestamp) {
```

2.126 CVF-123

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

Description The code could be made more readable if the condition for this conditional statement would be inverted and the rest of the function would be put into an explicit "then" branch.

Listing 123:

```
700 if (lastFundingTimes[_token].add(fundingInterval) > block.
    ↪ timestamp) {
```

2.127 CVF-124

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

Description This condition will be checked again in the "getNextFundingRate" function.
Recommendation Consider calling the function first and then checking for whether the returned value is zero.

Listing 124:

```
700 if (lastFundingTimes[_token].add(fundingInterval) > block.
    ↪ timestamp) {
```

2.128 CVF-125

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

Description The value "cumulativeFundingRates[_token]" that is read from the storage here, was written to the storage a few lines above.
Recommendation Consider caching the value in a local variable and reusing.

Listing 125:

```
708 emit UpdateFundingRate(_token, cumulativeFundingRates[_token]);
```

2.129 CVF-126

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

Description The code after this line looks like it is always executed, while it is executed only when lastFundingTimes[_token].add(fundingInterval) <= block.timestamp.
Recommendation Consider putting the rest of the function explicitly into an "else" branch.

Listing 126:

```
712 if (lastFundingTimes[_token].add(fundingInterval) > block.
    ↪ timestamp) { return 0; }
```

2.130 CVF-127

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

Description The value "lastfundingTimes[_token]" is calcualted twice.
Recommendation Consider calculating once and reusing.

Listing 127:

```
712 if (lastFundingTimes[_token].add(fundingInterval) > block.
    ↪ timestamp) { return 0; }

714 uint256 intervals = block.timestamp.sub(lastFundingTimes[_token
    ↪ ]).div(fundingInterval);
```

2.131 CVF-128

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

Description The code after this line looks like it is always executed, while it is executed only when poolAmount != 0.
Recommendation Consider putting the rest of the function explicitly into an "else" branch.

Listing 128:

```
716 if (poolAmount == 0) { return 0; }
```

2.132 CVF-129

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

Description In case the pool amount is zero, but the reserved amount is not, the utilization would not be zero, but rather infinity, or undefined.
Recommendation Consider reverting in case the pool amount is zero.

Listing 129:

```
723 if (poolAmount == 0) { return 0; }
```

2.133 CVF-130

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

Description When '_isLong' is true, for the purpose of the next average price calculation, the "getDelta" function should use the max token price instead of the min price. Using min price makes the result even worse for the user, then when max price is used, while even max price guarantees that the price error would be towards the protocol' benefit. When '_isLong' is false, "getDelta" should use the min token price.

Listing 130:

```
736 (bool hasProfit, uint256 delta) = getDelta(_indexToken, _size,
    ↪ _averagePrice, _isLong);
```

2.134 CVF-131

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

Recommendation These functions basically return the sign and the absolute value of the delta as two separate values. Returning a single signed value would make code simpler and easier to read.

Listing 131:

```
742 function getPositionDelta(address _account, address
    ↪ _collateralToken, address _indexToken, bool _isLong)
    ↪ public view returns (bool, uint256) {

748 function getDelta(address _indexToken, uint256 _size, uint256
    ↪ _averagePrice, bool _isLong) public override view returns
    ↪ (bool, uint256) {
```

2.135 CVF-132

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

Description It is suboptimal to read the whole position structure into the memory when only two fields from it are actually needed.
Recommendation Consider changing the "position" variable from "memory" to "storage".

Listing 132:

```
744 Position memory position = positions[key];
```

2.136 CVF-133

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

Recommendation The "minBps" should be read from the storage only when 'hasProfit' is true.

Listing 133:

```
764 uint256 minBps = minProfitBasisPoints[_indexToken];
```

2.137 CVF-134

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

Recommendation This line is redundant. It optimizes the rare case when _size == 0, but makes all the other cases more expensive.

Listing 134:

```
773 if (_size == 0) { return 0; }
```

2.138 CVF-135

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

Recommendation This line is redundant. It optimizes the rare case when fundingRate == 0, but makes all the other cases more expensive.

Listing 135:

```
776 if (fundingRate == 0) { return 0; }
```

2.139 CVF-136

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

Recommendation The "fundingFee" variable is redundant. Just return the calculated value.

Listing 136:

```
778 uint256 fundingFee = _size.mul(fundingRate).div(
    ↪ FUNDING_RATE_PRECISION);
```

2.140 CVF-137

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

Recommendation This line is redundant. It optimizes the rare case when _sizeDelta == 0, but makes all the other cases more expensive.

Listing 137:

```
783 if (_sizeDelta == 0) { return 0; }
```

2.141 CVF-138

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

Recommendation The subtraction here would not be necessary if the value "BASIS_POINTS_DIVISOR - marginFeeBasisPoints" would be stored instead of just "marginFeeBasisPoints".

Listing 138:

```
784 uint256 afterFeeUsd = _sizeDelta.mul(BASIS_POINTS_DIVISOR.sub(
    ↪ marginFeeBasisPoints)).div(BASIS_POINTS_DIVISOR);
```

2.142 CVF-139

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

Recommendation The "_hasProfit" variable is redundant, just use "hasProfit" instead.

Listing 139:

```
798 (bool _hasProfit, uint256 delta) = getDelta(_indexToken,
    ↪ position.size, position.averagePrice, _isLong);
```

2.143 CVF-140

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

Description When "_isLong" is true, for the purpose of the reduce collateral execution, the "getDelta" function should use the max token price instead of the min price. Using min price makes the result even worse for the user, then when max price is used, while even max price guarantees that the price error would be towards the protocol' benefit. When "_isLong" is false, "getDelta" should use the min token price.

Listing 140:

```
798 (bool _hasProfit, uint256 delta) = getDelta(_indexToken,
    ↪ position.size, position.averagePrice, _isLong);
```

2.144 CVF-141

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

Recommendation These two conditional statements could be restructured as: if (adjustedDelta > 0) { if (hasProfit) { ... } else { ... } }

Listing 141:

```
806 if (hasProfit && adjustedDelta > 0) {

817 if (!hasProfit && adjustedDelta > 0) {
```

2.145 CVF-142

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

Recommendation This code should be executed only when position.size > _sizeDelta, as in the opposite case the whole collateral will be sent to the user anyways, regardless of the "_collateralDelta" value of.

Listing 142:

```
833 if (_collateralDelta > 0) {
        usdOut = usdOut.add(_collateralDelta);
        position.collateral = position.collateral.sub(
            ↪ _collateralDelta);
    }
```

2.146 CVF-143

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

Description What if both, the "usdOut" value and the collateral are less then the fee, but their sum is greater? Why not to deduct as much fee as possible from the "usdOut" value, then deduct the rest from the collateral?

Listing 143:

```
844 // if the usdOut is more than the fee then deduct the fee from
        ↪ the usdOut directly
    // else deduct the fee from the position's collateral
```

2.147 CVF-144

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

Description This condition is always false, as in both cases, where the "_validatePosition" function is called, it is guaranteed, that _size > 0.
Recommendation Consider removing this conditional statement, or replacing is with require (_size > 0).

Listing 144:

```
863 if (_size == 0) {
```

2.148 CVF-145

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

Description The code below looks like it is always executed, while it is executed only when _size != 0.
Recommendation Consider putting the rest of the function explicitly into an "else" branch.

Listing 145:

```
866 }
```

2.149 CVF-146

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

Recommendation This could be rewritten as: require (msg.sender == _account || msg.sender == router || apporovedRouters [_account][msg.sender], ...);

Listing 146:

```
871 if (msg.sender == _account) { return; }
    if (msg.sender == router) { return; }
    require(approvedRouters[_account][msg.sender], "Vault: invalid
       ↪ msg.sender");
```

2.150 CVF-147

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

Recommendation This check should be done once before the "if" statement.

Listing 147:

```
879     require(whitelistedTokens[_collateralToken], "Vault:
          ↪ _collateralToken not whitelisted");

887 require(whitelistedTokens[_collateralToken], "Vault:
      ↪ _collateralToken not whitelisted");
```

2.151 CVF-148

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

Description These checks are basically equivalent, as inside the "if" statement the collateral token and the index token are the same. So, this check should be done once before the "if" statement in the form: require (!stableToken[_indexToken]);.

Listing 148:

```
880     require(!stableTokens[_collateralToken], "Vault:
          ↪ _collateralToken must not be a stableToken");

889 require(!stableTokens[_indexToken], "Vault: _indexToken must not
      ↪  be a stableToken");
```

2.152 CVF-149

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

Description The code below looks like it is always executed while it is only executed when '_isLong' is false.
Recommendation Consider putting the rest of the function explicitly into an "else" branch.

Listing 149:

```
882 }
```

2.153 CVF-150

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

Recommendation The subtraction here would not be needed if the value "BASIS_POINTS_DIVISOR - swapFeeBasisPoints" would be stored instead of just "swapFeeBasisPoints".

Listing 150:

```
893 uint256 afterFeeAmount = _amount.mul(BASIS_POINTS_DIVISOR.sub(
    ↪ swapFeeBasisPoints)).div(BASIS_POINTS_DIVISOR);
```

2.154 CVF-151

- **Severity** Minor
- **Status** Opened
- **Category** Bad naming
- **Source** Vault.sol

Description The name is confusing, as this function actually doesn't transfer any tokens.
Recommendation Consider renaming.

Listing 151:

```
913 function _transferIn(address _token) private returns (uint256) {
```

2.155 CVF-152

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

Recommendation It would be possible to replace these lines with a call to "_updateTokenBalance" if it would return the updated balance.

Listing 152:

```
915 uint256 nextBalance = IERC20(_token).balanceOf(address(this));
tokenBalances[_token] = nextBalance;
```

2.156 CVF-153

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

Description This line duplicates the logic of the "_updateTokenBalance" function.

Listing 153:

```
923 tokenBalances[_token] = IERC20(_token).balanceOf(address(this));
```

2.157 CVF-154

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

Description This function should return the updated token balance.

Listing 154:

```
926 function _updateTokenBalance(address _token) private {
```

2.158 CVF-155

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

Description In many cases, the actual token balance is already stored in the "tokenBalances" mapping. In such cases, it is suboptimal to query the balance from the token contract again. **Recommendation** Consider refactoring.

Listing 155:

```
933 uint256 balance = IERC20(_token).balanceOf(address(this));
```

2.159 CVF-156

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

Description Here the value "poolAmounts[_token]", written into the storage two lines above, is read from the storage again. This is suboptimal.
Recommendation Consider caching this value in a local variable.

Listing 156:

```
934 require(poolAmounts[_token] <= balance, "Vault: invalid increase
    ↪ ");
```

2.160 CVF-157

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

Description Here the value "poolAmounts[_token]", written into the storage in the previous line, is read from the storage again. This is suboptimal.
Recommendation Consider caching this value in a local variable.

Listing 157:

```
940 require(reservedAmounts[_token] <= poolAmounts[_token], "Vault:
    ↪ reserve exceeds pool");
```

2.161 CVF-158

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

Recommendation The capping means that is USDG amount for a token is decreased by X and then increased be X, the amount will not necessary return to the initial value.

Listing 158:

```
951 // since USDG can be minted using multiple assets
    // it is possible for the USDG debt for a single asset to be
    ↪ less than zero
    // the USDG debt is capped to zero for this case
```

2.162 CVF-159

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

Recommendation These likes could be rewritten as: usdgAmounts [_token] = value <= _amount ? 0 : value - _amount;.

Listing 159:

```
954 if (value <= _amount) {
        usdgAmounts[_token] = 0;
        emit DecreaseUsdgAmount(_token, value);
        return;
    }
    usdgAmounts[_token] = value.sub(_amount);
```

2.163 CVF-160

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

Description The code below this line looks like it is always executed, while it is executed only when value > _amount.
Recommendation Consider putting the rest of the function explicitly into an "else" branch.

Listing 160:

```
958 }
```

2.164 CVF-161

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

Description Here the value "reservedAmounts[_token]", written to the storage in the previous line, is read from the storage again. This is suboptimal.
Recommendation Consider caching this value in a local variable.

Listing 161:

```
965 require(reservedAmounts[_token] <= poolAmounts[_token], "Vault:
    ↪ reserve exceeds pool");
```