Quantstamp Security Assessment Certificate

November 27th 2020 — Quantstamp Verified

Cryptex

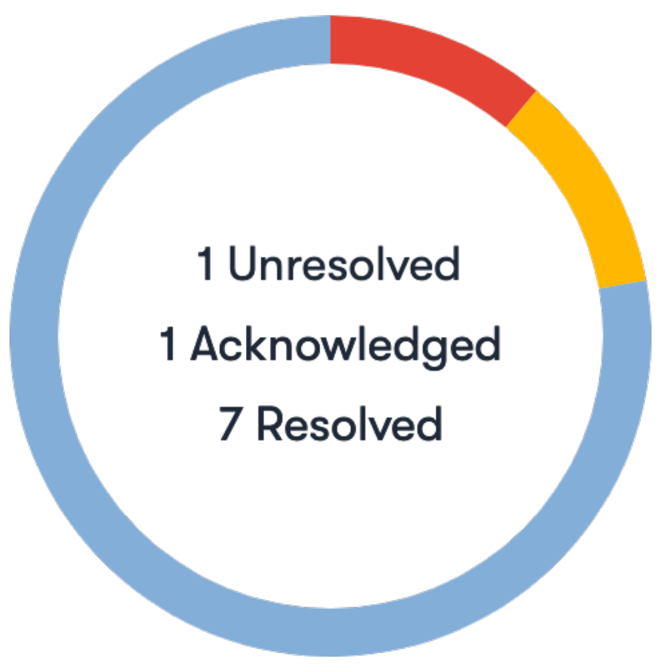
This security assessment was prepared by Quantstamp, the leader in blockchain security

Executive Summary

Type	DeFi
Auditors	Jake Goh Si Yuan, Senior Security Researcher Luís Fernando Schultz Xavier da Silveira, Security Consultant Joseph Xu, Technical R&D Advisor
Timeline	2020-10-21 through 2020-11-26
EVM	Muir Glacier
Languages	Solidity, Javascript
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	Web Documentation Whitepaper
Documentation Quality	Medium
Test Quality	Medium
Source Code	

Repository	Commit
contracts	31adfd2
None	9bd0481

Total Issues	9	(7 Resolved)
High Risk Issues	0	(0 Resolved)
Medium Risk Issues	3	(2 Resolved)
Low Risk Issues	1	(1 Resolved)
Informational Risk Issues	4	(4 Resolved)
Undetermined Risk Issues	1	(0 Resolved)



High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or is reasonably likely to lead to moderate financial impact.
Low Risk	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low-impact in view of the client's business circumstances.
Informational	The issue does not post an immediate risk, but is relevant to security best practices or Defence in Depth.
Undetermined	The impact of the issue is uncertain.

Unresolved	Acknowledged the existence of the risk, and decided to accept it without engaging in special efforts to control it.
Acknowledged	The issue remains in the code but is a result of an intentional business or design decision. As such, it is supposed to be addressed outside the programmatic means, such as: 1) comments, documentation, README, FAQ; 2) business processes; 3) analyses showing that the issue shall have no negative consequences in practice (e.g., gas analysis, deployment settings).
Resolved	Adjusted program implementation, requirements or constraints to eliminate the risk.
Mitigated	Implemented actions to minimize the impact or likelihood of the risk.

Summary of Findings

In this audit, we have identified 3 medium, 1 low, 4 informational and 1 undetermined risk severity issues, along with various best practices and documentation improvement suggestions, that we seriously encourage the Cryptex team to be addressing. We have found the codebase to generally be readable, although there could be further efforts to reduce duplication of code. At the same time, we have found documentation to be present in quantity, but missing key details, accuracy or depth when it came to important aspects of the platform and its formulae. This made it particularly difficult to audit at certain times, and we recommend that the Cryptex team to take some effort into expanding it.

ID	Description	Severity	Status
QSP-1	Liquidation may not succeed due to high penalty	^ Medium	Unresolved
QSP-2	Unsynchronized changes to vault state variables may lead to catastrophic aberrations	^ Medium	Fixed
QSP-3	Unchecked Return Value	^ Medium	Fixed
QSP-4	Vault Handlers can be reinitialized	˅ Low	Fixed
QSP-5	Repeated magic number in oracle calls	○ Informational	Fixed
QSP-6	Unlocked Pragma	○ Informational	Fixed
QSP-7	Liquidation Penalty validation	○ Informational	Fixed
QSP-8	Incorrect/Missing Visibility	○ Informational	Fixed
QSP-9	More tokens than `cap`	? Undetermined	Acknowledged

Quantstamp Audit Breakdown

Quantstamp's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.

Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow / underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Methodology

The Quantstamp auditing process follows a routine series of steps:

1. Code review that includes the following
   i. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
   ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Quantstamp describe.
2. Testing and automated analysis that includes the following:
   i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   ii. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, and actionable recommendations to help you take steps to secure your smart contracts.

Toolset

The notes below outline the setup and steps performed in the process of this audit.

Setup

Tool Setup:

- Slither v0.6.13

Steps taken to run the tools:

1. Installed the Slither tool: `pip install slither-analyzer`
2. Run Slither from the project directory: `slither .`

Findings

QSP-1 Liquidation may not succeed due to high penalty

Severity: Medium Risk

Status: Unresolved

File(s) affected: `IVaultHandler.sol`

Description: The current liquidation procedure in the smart contract assumes sufficient collateral in the vault to cover both the debt and the liquidation penalty at all times. However, liquidation may not be successful depending on the market environment and if the liquidation penalty parameter is too high. This is not exactly a smart contract issue since the risk of an undercollateralized vault will always be present. One could wait for the market condition to improve, but if the vault is not liquidated reasonably quickly, then there will be risk of the vault's collaterialization ratio going below 100% at some point. This risk can be quite serious if certain ERC20 tokens are accepted as collateral later on.

Exploit Scenario: Consider the following scenario: there exists an `ETH` vault with minimum ratio of 150% and liquidation penalty of 30%. Lets say the `TCAP` price is $50 and `ETH` price is at $400. With these figures, let's imagine that this vault originally generates 300 `TCAP` debt using 100 `ETH` collateral, creating a ratio of 267%.
Let's say some event takes place, such as a major security incident, and the `ETH` price drops to $150 whilst the `TCAP` is at $40. The ratio is now 125% and the vault is at risk of being liquidated. In the `requiredLiquidationCollateral()` calculation, the dividend (missing `TCAP` equivalent from the vault) is 75 `TCAP` tokens worth of `ETH` times 100. The divisor is $150 - (30 + 100) = 20$ therefore making 375 `TCAP` tokens required to liquidate the vault. However, since the vault only has 300 `TCAP` debt, it will not be able to `_burn` the tokens. At the same time, the liquidation reward of 130 `ETH` is also more than the collateral available in the vault.

Recommendation: A few approaches to mitigating this risk:

- Allow an emergency mechanism to reduce the liquidation penalty with less than 3 days timelock.
- A mechanism to allow users to accept a lower liquidation reward if liquidating the full vault for less than the posted liquidation penalty (possibly introducing some form of insurance fee or minting TCAP to provide additional incentives here).
- Ensure a sufficiently high collateralization ratio in the system to buy some time between a vault being at risk for liquidation and being completely under water.

Update: The emergency functions will enable easier liquidation and mitigate the risk of the entire system being under-collateralized. One item to note, however, is that the 3 days timelock would apply for reverting the system back to positive burn fee and liquidation penalty, so this is somewhat of a "nuclear" option.
As a general risk management best practice, it is recommended for the team to develop a specific, quantitative set of rules that would trigger these emergency functions (e.g., x% of vaults are within y% of collateralization limit after z% adverse moves in the market within the last t days). A dashboard should be in place to monitor the overall health of the system. Ideally, such policy is made public for user awareness but at the very least the quantitative criteria for invoking risk management protocols should be communicated internally within the team.

QSP-2 Unsynchronized changes to vault state variables may lead to catastrophic aberrations

Severity: Medium Risk

Status: Fixed

File(s) affected: `IVaultHandler.sol`, `Orchestrator.sol`

Description: The `onlyOwner` setters in `IVaultHandler` and the `Functions` enum in `Orchestrator` illustrates the intention to have the state variables in `IVaultHandler`be changed by the `Orchestrator`. The `Orchestrator` uses a timelocked commit-reveal scheme that works by having the `owner` of the contract call `unlockFunction` which sets a timelock which can be opened a constant 3 days later by a second call.
Theoretically, the 3 days period allows all users of the platform to have time to become aware of an upcoming change to a particular variable and vault, although not on what the actual change would be. These vault variable changes are pretty significant in terms of functionalities : the liquidation ratio or the divisor can lead to "premature" liquidation, whilst changes to any of the address variables can lead to erroneous behaviour and results.
Some of these changes require significant migration efforts. For instance consider a change to `collateralContract`. The existing vaults have a non-zero amount of `Vault.Collateral` and this was the accounting done when the `collateralContract` was pointing to the previous token. After such a change, there are questions remaining unanswered -- for instance, would users first get refunded their previous tokens, or would everyone be automatically switched over to the new one? Or, would there be a need to switch and synchronize `collateralPriceOracle` -- and requires much more significant documenting and planning.

Recommendation: Either remove the functionality to switch some or all of the variables after it has already been set, AND/OR, document and plan this process much more thoroughly and recognize that some of these changes may have non-trivial side-effects.

QSP-3 Unchecked Return Value

Severity: Medium Risk

Status: Fixed

File(s) affected: `IVaultHandler.sol`

Related Issue(s): SWC-104

Description: Most functions will return a `true` or `false` value upon success. Some functions, like `send()`, are more crucial to check than others. It's important to ensure that every necessary function is checked. In particular, these functions relate to the IERC20 methods which may fail silently and rely on passing boolean status to inform.
The instances we have spotted are as follows :

1. In IVaultHandler.sol::L322 `transferFrom` return value ignored.
2. In IVaultHandler.sol::L354 `transfer` return value ignored.
3. In IVaultHandler.sol::L426 `transfer` return value ignored.

Recommendation: Check and validate the return values of these functions.

QSP-4 Vault Handlers can be reinitialized

Severity: Low Risk

Status: Fixed

File(s) affected: `IVaultHandler.sol`

Description: All vault handlers inherit the IVaultHandler contract, which has the `initialize` method that all other contracts inherit and use. It is currently commented in L170 that the method `Can only be called once`. However, that is not true at all, and can be called again even after initialization has taken place. The lowered chance and severity is due to the fact that this is an `onlyOwner` method. However, given that there is an ambigious looking comment for this method, this issue has to be highlighted.

Recommendation: Introduce a state variable that can be checked for before initialization to ensure that only one takes place.

QSP-5 Repeated magic number in oracle calls

Severity: Informational

Status: Fixed

File(s) affected: `IVaultHandler.sol`

Description: The calls to the Chainlink oracles for the collateral price and `TCAP` price have the usage pattern of

```
xxxPriceOracle
      .getLatestAnswer()
      .toUint256()
      .mul(10000000000)
```

which can be inferred to be a decimals normalization from retrieving oracle results. However, instead of using `10000000000` as a constant, it is simply duplicated across different usages. This may be a potential source of bugs in the future as any changes will have to be similarly and precisely duplicated. It may also be cleaner still to refactor it such that either the normalisation digits are available as a public method, or the entire call and it's post-treatment can be packaged into a public view method.

Recommendation: Move the magic number into its own constant and consider refactoring such that you have methods of the signature `xxxPrice() public view returns(uint256 price)` that contains

```
.getLatestAnswer()
      .toUint256()
      .mul(10000000000);
```

for any oracle call.
Update: The recommended change has been implemented by moving the magic number to a `uint256 public constant oracleDigits` and `function getOraclePrice()`. We note that the assumption is that the `oracleDigits` constant will not change in the Chainlink aggregator contract.

QSP-6 Unlocked Pragma

Severity: Informational

Status: Fixed

File(s) affected: `All contracts`

Related Issue(s): SWC-103

Description: Every Solidity file specifies in the header a version number of the format `pragma solidity (^)0.6.8`. The caret (^) before the version number implies an unlocked pragma, meaning that the compiler will use the specified version *and above*, hence the term "unlocked".

Recommendation: For consistency and to prevent unexpected behavior in the future, it is recommended to remove the caret to lock the file onto a specific Solidity version.

QSP-7 Liquidation Penalty validation

Severity: Informational

Status: Fixed

File(s) affected: `IVaultHandler.sol`

Description: It may be helpful to add a check on `setLiquidationPenalty()` so that liquidationPenalty + 100 < ratio. This will prevent `requiredLiquidationCollateral()` from breaking due to the divisor <= 0.

QSP-8 Incorrect/Missing Visibility

Severity: Informational

Status: Fixed

File(s) affected: `IVaultHandler.sol`

Related Issue(s): SWC-100, SWC-108

Description: The visibility of a function or field changes determines how it can be accessed by others. Using the right visibility ensures optimal gas costs and reduces possibilities of attacks. The four types of visibility are:

- `external` - can be called by any other contract (but not within the contract itself)
- `public` - can be called anywhere
- `internal` - can only be called within contract, and inherited contracts will inherit this functionality
- `private` - can only be called within contract, cannot be inherited

1. In IVaultHandler.sol::L78, `counter` does not have any visibility explictly stated.
2. In IVaultHandler.sol `initialize` should be set to `external`.
3. In IVaultHandler.sol `setTCAPContract` should be set to `external.`
4. In IVaultHandler.sol `setTCAPOracle` should be set to `external`.
5. In IVaultHandler.sol `setCollateralContract` should be set to `external`.
6. In IVaultHandler.sol `setCollateralPriceOracleContract` should be set to `external.`
7. In IVaultHandler.sol `setETHPriceOracle` should be set to `external`.
8. In IVaultHandler.sol `setDivisor` should be set to `external`.
9. In IVaultHandler.sol `setRatio` should be set to `external`.
10. In IVaultHandler.sol `setBurnFee` should be set to `external`.
11. In IVaultHandler.sol `setLiquidationPenalty` should be set to `external.`
12. In IVaultHandler.sol `createVault` should be set to `external.`

13. In IVaultHandler.sol `addCollateral` should be set to `external`.
14. In IVaultHandler.sol `removeCollateral` should be set to `external`.
15. In IVaultHandler.sol `mint` should be set to `external`.
16. In IVaultHandler.sol `burn` should be set to `external`.
17. In IVaultHandler.sol `liquidateVault` should be set to `external`.
18. In IVaultHandler.sol `pause` should be set to `external`.
19. In IVaultHandler.sol `unpause` should be set to `external`.
20. In IVaultHandler.sol `retrieveFees` should be set to `external`.
21. In IVaultHandler.sol `getVault` should be set to `external`.
22. In Orchestrator.sol `initializeVault` should be set to `external`.
23. In Orchestrator.sol `unlockFunction` should be set to `external`.
24. In Orchestrator.sol `lockVaultFunction` should be set to `external`.
25. In Orchestrator.sol `setDivisor` should be set to `external`.
26. In Orchestrator.sol `setRatio` should be set to `external`.
27. In Orchestrator.sol `setBurnFee` should be set to `external`.
28. In Orchestrator.sol `setETHOracle` should be set to `external`.
29. In Orchestrator.sol `setCollateralOracle` should be set to `external`.
30. In Orchestrator.sol `setCollateral` should be set to `external`.
31. In Orchestrator.sol `setTCAPOracle` should be set to `external`.
32. In Orchestrator.sol `setTCAP` should be set to `external`.
33. In Orchestrator.sol `setLiquidationPenalty` should be set to `external`.
34. In Orchestrator.sol `enableTCAPCap` should be set to `external`.
35. In Orchestrator.sol `retrieveFees` should be set to `external`.
36. In Orchestrator.sol `retrieveVaultFees` should be set to `external`.
37. In Orchestrator.sol `unpauseVault` should be set to `external`.
38. In Orchestrator.sol `pauseVault` should be set to `external`.
39. In Orchestrator.sol `addTCAPVault` should be set to `external`.
40. In Orchestrator.sol `setTCAPCap` should be set to `external`.
41. In Orchestrator.sol `enableTCAPCap` should be set to `external`.
42. In TCAP.sol `addTokenHandler` should be set to `external`.
43. In TCAP.sol `enableCap` should be set to `external`.
44. In TCAP.sol `setCap` should be set to `external`.
45. In TCAP.sol `burn` should be set to `external`.
46. In TCAP.sol `mint` should be set to `external`.
47. In TCAP.sol `setCap` should be set to `external`.
48. In TCAP.sol `enableCap` should be set to `external`.

Recommendation: Explicitly set correct visibility to all affected variables.

QSP-9 More tokens than `cap`

Severity: *Undetermined*

Status: Acknowledged

File(s) affected: `TCAP.sol`

Description: There is a token supply `cap` at the `TCAP` contract that is not always enabled, which according to L89-90 when `When capEnabled is true, mint is not allowed to issue tokens that would increase the total supply above the specified capacity`. This also means that when the `capEnabled` is being set to true, there is a possibility that the actual total supply will be more than the existing `cap`, thus violating in principle the state boolean.

Recommendation: This might be something that's intended by design, in which case it will serve to have more documentation around this particular scenario. Otherwise, consider validating before allowing `enableCap` to be set.
Update: This issue is addressed by updating the in-code comments to reflect the fact that it's possible to have more tokens than the cap but further minting is not allowed.

Automated Analyses

Slither

There were 271 issues found by Slither with 46 detectors being used across the 31 contracts(inherited library contracts included). We have looked through all of the issues and find most of them to be false positives, and have added to the report the valid issues.

```
ETHVaultHandler.safeTransferETH(address,uint256) (contracts/ETHVaultHandler.sol#68-71) sends eth to arbitrary user
    Dangerous calls:
    - (success) = to.call{value: value}(new bytes(0)) (contracts/ETHVaultHandler.sol#69)
IVaultHandler.retrieveFees() (contracts/IVaultHandler.sol#447-451) sends eth to arbitrary user
    Dangerous calls:
    - address(owner()).transfer(amount) (contracts/IVaultHandler.sol#449)
Orchestrator.retrieveFees() (contracts/Orchestrator.sol#465-468) sends eth to arbitrary user
    Dangerous calls:
    - address(owner()).transfer(amount) (contracts/Orchestrator.sol#467)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations ✓[0m
INFO:Detectors: ✓[93m
IVaultHandler.liquidationReward(uint256) (contracts/IVaultHandler.sol#545-559) performs a multiplication on the result of a division:
    -reward = (req.mul(liquidationPenalty.add(100))).div(100) (contracts/IVaultHandler.sol#557)
    -rewardCollateral = (reward.mul(tcapPrice)).div(collateralPrice) (contracts/IVaultHandler.sol#558)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply ✓[0m
INFO:Detectors: ✓[93m
Reentrancy in Orchestrator.enableTCAPCap(TCAP,bool) (contracts/Orchestrator.sol#477-489):
    External calls:
    - _tcap.enableCap(_enable) (contracts/Orchestrator.sol#487)
    - validTCAP(_tcap) (contracts/Orchestrator.sol#480)
        - require(bool,string)(ERC165Checker.supportsInterface(address(_tcap),_INTERFACE_ID_TCAP),Not a valid TCAP ERC20)
(contracts/Orchestrator.sol#88-91)
        - (success,result) = account.staticcall{gas: 30000}(encodedParams)
(node_modules/@openzeppelin/contracts/introspection/ERC165Checker.sol#102)
    State variables written after the call(s):
    - _lockFunction(address(_tcap),Functions.ENABLECAP) (contracts/Orchestrator.sol#488)
        - timelock[address(_contract)][_fn] = 0 (contracts/Orchestrator.sol#195)
    - _lockFunction(address(_tcap),Functions.ENABLECAP) (contracts/Orchestrator.sol#488)
        - timelockValue[address(_contract)][_fn] = 0 (contracts/Orchestrator.sol#196)
Reentrancy in Orchestrator.initializeVault(IVaultHandler,uint256,uint256,uint256,uint256,address,TCAP,address,address,address)
(contracts/Orchestrator.sol#140-168):
    External calls:
    - _validChainlinkOracle(_tcapOracle) (contracts/Orchestrator.sol#153)
        - require(bool,string)(ERC165Checker.supportsInterface(address(_oracle),_INTERFACE_ID_CHAINLINK_ORACLE),Not a valid Chainlink Oracle)
(contracts/Orchestrator.sol#116-122)
        - (success,result) = account.staticcall{gas: 30000}(encodedParams)
(node_modules/@openzeppelin/contracts/introspection/ERC165Checker.sol#102)
    - _validChainlinkOracle(_collateralOracle) (contracts/Orchestrator.sol#154)
        - require(bool,string)(ERC165Checker.supportsInterface(address(_oracle),_INTERFACE_ID_CHAINLINK_ORACLE),Not a valid Chainlink Oracle)
(contracts/Orchestrator.sol#116-122)
        - (success,result) = account.staticcall{gas: 30000}(encodedParams)
(node_modules/@openzeppelin/contracts/introspection/ERC165Checker.sol#102)
    - _validChainlinkOracle(_ethOracle) (contracts/Orchestrator.sol#155)
        - require(bool,string)(ERC165Checker.supportsInterface(address(_oracle),_INTERFACE_ID_CHAINLINK_ORACLE),Not a valid Chainlink Oracle)
(contracts/Orchestrator.sol#116-122)
        - (success,result) = account.staticcall{gas: 30000}(encodedParams)
(node_modules/@openzeppelin/contracts/introspection/ERC165Checker.sol#102)
    - _vault.initialize(_divisor,_ratio,_burnFee,_liquidationPenalty,_tcapOracle,_tcapAddress,_collateralAddress,_collateralOracle,_ethOracle)
(contracts/Orchestrator.sol#156-166)
    - validVault(_vault) (contracts/Orchestrator.sol#151)
        - require(bool,string)(ERC165Checker.supportsInterface(address(_vault),_INTERFACE_ID_IVAULT),Not a valid vault)
(contracts/Orchestrator.sol#76-79)
        - (success,result) = account.staticcall{gas: 30000}(encodedParams)
(node_modules/@openzeppelin/contracts/introspection/ERC165Checker.sol#102)
    - validTCAP(_tcapAddress) (contracts/Orchestrator.sol#151)
        - require(bool,string)(ERC165Checker.supportsInterface(address(_tcap),_INTERFACE_ID_TCAP),Not a valid TCAP ERC20)
(contracts/Orchestrator.sol#88-91)
        - (success,result) = account.staticcall{gas: 30000}(encodedParams)
(node_modules/@openzeppelin/contracts/introspection/ERC165Checker.sol#102)
    State variables written after the call(s):
    - initialized[_vault] = true (contracts/Orchestrator.sol#167)
Reentrancy in IVaultHandler.liquidateVault(uint256,uint256) (contracts/IVaultHandler.sol#407-428):
    External calls:
    - _burn(vault.Id,requiredCollateral) (contracts/IVaultHandler.sol#424)
        - TCAPToken.burn(msg.sender,_amount) (contracts/IVaultHandler.sol#650)
    State variables written after the call(s):
    - vault.Collateral = vault.Collateral.sub(reward) (contracts/IVaultHandler.sol#425)
Reentrancy in Orchestrator.setBurnFee(IVaultHandler,uint256) (contracts/Orchestrator.sol#265-277):
    External calls:
    - _vault.setBurnFee(_burnFee) (contracts/Orchestrator.sol#275)
    - validVault(_vault) (contracts/Orchestrator.sol#268)
        - require(bool,string)(ERC165Checker.supportsInterface(address(_vault),_INTERFACE_ID_IVAULT),Not a valid vault)
(contracts/Orchestrator.sol#76-79)
        - (success,result) = account.staticcall{gas: 30000}(encodedParams)
(node_modules/@openzeppelin/contracts/introspection/ERC165Checker.sol#102)
    State variables written after the call(s):
    - _lockFunction(address(_vault),Functions.BURNFEE) (contracts/Orchestrator.sol#276)
        - timelock[address(_contract)][_fn] = 0 (contracts/Orchestrator.sol#195)
    - _lockFunction(address(_vault),Functions.BURNFEE) (contracts/Orchestrator.sol#276)
        - timelockValue[address(_contract)][_fn] = 0 (contracts/Orchestrator.sol#196)
Reentrancy in Orchestrator.setCollateral(IVaultHandler,IERC20) (contracts/Orchestrator.sol#358-370):
    External calls:
    - _vault.setCollateralContract(_collateral) (contracts/Orchestrator.sol#368)
    - validVault(_vault) (contracts/Orchestrator.sol#361)
        - require(bool,string)(ERC165Checker.supportsInterface(address(_vault),_INTERFACE_ID_IVAULT),Not a valid vault)
(contracts/Orchestrator.sol#76-79)
        - (success,result) = account.staticcall{gas: 30000}(encodedParams)
(node_modules/@openzeppelin/contracts/introspection/ERC165Checker.sol#102)
    State variables written after the call(s):
    - _lockFunction(address(_vault),Functions.COLLATERAL) (contracts/Orchestrator.sol#369)
        - timelock[address(_contract)][_fn] = 0 (contracts/Orchestrator.sol#195)
    - _lockFunction(address(_vault),Functions.COLLATERAL) (contracts/Orchestrator.sol#369)
        - timelockValue[address(_contract)][_fn] = 0 (contracts/Orchestrator.sol#196)
Reentrancy in Orchestrator.setCollateralOracle(IVaultHandler,address) (contracts/Orchestrator.sol#380-393):
    External calls:
    - _vault.setCollateralPriceOracle(ChainlinkOracle(_collateralOracle)) (contracts/Orchestrator.sol#391)
    - validVault(_vault) (contracts/Orchestrator.sol#383)
        - require(bool,string)(ERC165Checker.supportsInterface(address(_vault),_INTERFACE_ID_IVAULT),Not a valid vault)
(contracts/Orchestrator.sol#76-79)
        - (success,result) = account.staticcall{gas: 30000}(encodedParams)
(node_modules/@openzeppelin/contracts/introspection/ERC165Checker.sol#102)
    - validChainlinkOracle(_collateralOracle) (contracts/Orchestrator.sol#389)
        - require(bool,string)(ERC165Checker.supportsInterface(address(_oracle),_INTERFACE_ID_CHAINLINK_ORACLE),Not a valid Chainlink Oracle)
(contracts/Orchestrator.sol#100-106)
        - (success,result) = account.staticcall{gas: 30000}(encodedParams)
(node_modules/@openzeppelin/contracts/introspection/ERC165Checker.sol#102)
    State variables written after the call(s):
    - _lockFunction(address(_vault),Functions.COLLATERALORACLE) (contracts/Orchestrator.sol#392)
        - timelock[address(_contract)][_fn] = 0 (contracts/Orchestrator.sol#195)
    - _lockFunction(address(_vault),Functions.COLLATERALORACLE) (contracts/Orchestrator.sol#392)
        - timelockValue[address(_contract)][_fn] = 0 (contracts/Orchestrator.sol#196)
Reentrancy in Orchestrator.setDivisor(IVaultHandler,uint256) (contracts/Orchestrator.sol#221-233):
    External calls:
    - _vault.setDivisor(_divisor) (contracts/Orchestrator.sol#231)
    - validVault(_vault) (contracts/Orchestrator.sol#224)
        - require(bool,string)(ERC165Checker.supportsInterface(address(_vault),_INTERFACE_ID_IVAULT),Not a valid vault)
(contracts/Orchestrator.sol#76-79)
        - (success,result) = account.staticcall{gas: 30000}(encodedParams)
(node_modules/@openzeppelin/contracts/introspection/ERC165Checker.sol#102)
    State variables written after the call(s):
    - _lockFunction(address(_vault),Functions.DIVISOR) (contracts/Orchestrator.sol#232)
        - timelock[address(_contract)][_fn] = 0 (contracts/Orchestrator.sol#195)
    - _lockFunction(address(_vault),Functions.DIVISOR) (contracts/Orchestrator.sol#232)
        - timelockValue[address(_contract)][_fn] = 0 (contracts/Orchestrator.sol#196)
Reentrancy in Orchestrator.setETHOracle(IVaultHandler,address) (contracts/Orchestrator.sol#403-416):
    External calls:
    - _vault.setETHPriceOracle(ChainlinkOracle(_ethOracles)) (contracts/Orchestrator.sol#414)
    - validVault(_vault) (contracts/Orchestrator.sol#406)
        - require(bool,string)(ERC165Checker.supportsInterface(address(_vault),_INTERFACE_ID_IVAULT),Not a valid vault)
(contracts/Orchestrator.sol#76-79)
        - (success,result) = account.staticcall{gas: 30000}(encodedParams)
(node_modules/@openzeppelin/contracts/introspection/ERC165Checker.sol#102)
    - validChainlinkOracle(_ethOracles) (contracts/Orchestrator.sol#412)
        - require(bool,string)(ERC165Checker.supportsInterface(address(_oracle),_INTERFACE_ID_CHAINLINK_ORACLE),Not a valid Chainlink Oracle)
```

```
(contracts/Orchestrator.sol#100-106)
        - (success,result) = account.staticcall{gas: 30000}(encodedParams)
(node_modules/@openzeppelin/contracts/introspection/ERC165Checker.sol#102)
    State variables written after the call(s):
    - _lockFunction(address(_vault),Functions.ETHORACLE) (contracts/Orchestrator.sol#415)
        - timelock[address(_contract)][_fn] = 0 (contracts/Orchestrator.sol#195)
    - _lockFunction(address(_vault),Functions.ETHORACLE) (contracts/Orchestrator.sol#415)
        - timelockValue[address(_contract)][_fn] = 0 (contracts/Orchestrator.sol#196)
Reentrancy in Orchestrator.setLiquidationPenalty(IVaultHandler,uint256) (contracts/Orchestrator.sol#287-302):
    External calls:
    - _vault.setLiquidationPenalty(_liquidationPenalty) (contracts/Orchestrator.sol#300)
    - validVault(_vault) (contracts/Orchestrator.sol#293)
        - require(bool,string)(ERC165Checker.supportsInterface(address(_vault),_INTERFACE_ID_IVAULT),Not a valid vault)
(contracts/Orchestrator.sol#76-79)
        - (success,result) = account.staticcall{gas: 30000}(encodedParams)
(node_modules/@openzeppelin/contracts/introspection/ERC165Checker.sol#102)
    State variables written after the call(s):
    - _lockFunction(address(_vault),Functions.LIQUIDATION) (contracts/Orchestrator.sol#301)
        - timelock[address(_contract)][_fn] = 0 (contracts/Orchestrator.sol#195)
    - _lockFunction(address(_vault),Functions.LIQUIDATION) (contracts/Orchestrator.sol#301)
        - timelockValue[address(_contract)][_fn] = 0 (contracts/Orchestrator.sol#196)
Reentrancy in Orchestrator.setRatio(IVaultHandler,uint256) (contracts/Orchestrator.sol#243-255):
    External calls:
    - _vault.setRatio(_ratio) (contracts/Orchestrator.sol#253)
    - validVault(_vault) (contracts/Orchestrator.sol#246)
        - require(bool,string)(ERC165Checker.supportsInterface(address(_vault),_INTERFACE_ID_IVAULT),Not a valid vault)
(contracts/Orchestrator.sol#76-79)
        - (success,result) = account.staticcall{gas: 30000}(encodedParams)
(node_modules/@openzeppelin/contracts/introspection/ERC165Checker.sol#102)
    State variables written after the call(s):
    - _lockFunction(address(_vault),Functions.RATIO) (contracts/Orchestrator.sol#254)
        - timelock[address(_contract)][_fn] = 0 (contracts/Orchestrator.sol#195)
    - _lockFunction(address(_vault),Functions.RATIO) (contracts/Orchestrator.sol#254)
        - timelockValue[address(_contract)][_fn] = 0 (contracts/Orchestrator.sol#196)
Reentrancy in Orchestrator.setTCAP(IVaultHandler,TCAP) (contracts/Orchestrator.sol#312-325):
    External calls:
    - _vault.setTCAPContract(_tcap) (contracts/Orchestrator.sol#323)
    - validVault(_vault) (contracts/Orchestrator.sol#315)
        - require(bool,string)(ERC165Checker.supportsInterface(address(_vault),_INTERFACE_ID_IVAULT),Not a valid vault)
(contracts/Orchestrator.sol#76-79)
        - (success,result) = account.staticcall{gas: 30000}(encodedParams)
(node_modules/@openzeppelin/contracts/introspection/ERC165Checker.sol#102)
    - validTCAP(_tcap) (contracts/Orchestrator.sol#321)
        - require(bool,string)(ERC165Checker.supportsInterface(address(_tcap),_INTERFACE_ID_TCAP),Not a valid TCAP ERC20)
(contracts/Orchestrator.sol#88-91)
        - (success,result) = account.staticcall{gas: 30000}(encodedParams)
(node_modules/@openzeppelin/contracts/introspection/ERC165Checker.sol#102)
    State variables written after the call(s):
    - _lockFunction(address(_vault),Functions.TCAP) (contracts/Orchestrator.sol#324)
        - timelock[address(_contract)][_fn] = 0 (contracts/Orchestrator.sol#195)
    - _lockFunction(address(_vault),Functions.TCAP) (contracts/Orchestrator.sol#324)
        - timelockValue[address(_contract)][_fn] = 0 (contracts/Orchestrator.sol#196)
Reentrancy in Orchestrator.setTCAPCap(TCAP,uint256) (contracts/Orchestrator.sol#498-510):
    External calls:
    - _tcap.setCap(_cap) (contracts/Orchestrator.sol#508)
    - validTCAP(_tcap) (contracts/Orchestrator.sol#501)
        - require(bool,string)(ERC165Checker.supportsInterface(address(_tcap),_INTERFACE_ID_TCAP),Not a valid TCAP ERC20)
(contracts/Orchestrator.sol#88-91)
        - (success,result) = account.staticcall{gas: 30000}(encodedParams)
(node_modules/@openzeppelin/contracts/introspection/ERC165Checker.sol#102)
    State variables written after the call(s):
    - _lockFunction(address(_tcap),Functions.SETCAP) (contracts/Orchestrator.sol#509)
        - timelock[address(_contract)][_fn] = 0 (contracts/Orchestrator.sol#195)
    - _lockFunction(address(_tcap),Functions.SETCAP) (contracts/Orchestrator.sol#509)
        - timelockValue[address(_contract)][_fn] = 0 (contracts/Orchestrator.sol#196)
Reentrancy in Orchestrator.setTCAPOracle(IVaultHandler,address) (contracts/Orchestrator.sol#335-348):
    External calls:
    - _vault.setTCAPOracle(ChainlinkOracle(_tcapOracle)) (contracts/Orchestrator.sol#346)
    - validVault(_vault) (contracts/Orchestrator.sol#338)
        - require(bool,string)(ERC165Checker.supportsInterface(address(_vault),_INTERFACE_ID_IVAULT),Not a valid vault)
(contracts/Orchestrator.sol#76-79)
        - (success,result) = account.staticcall{gas: 30000}(encodedParams)
(node_modules/@openzeppelin/contracts/introspection/ERC165Checker.sol#102)
    - validChainlinkOracle(_tcapOracle) (contracts/Orchestrator.sol#344)
        - require(bool,string)(ERC165Checker.supportsInterface(address(_oracle),_INTERFACE_ID_CHAINLINK_ORACLE),Not a valid Chainlink Oracle)
(contracts/Orchestrator.sol#100-106)
        - (success,result) = account.staticcall{gas: 30000}(encodedParams)
(node_modules/@openzeppelin/contracts/introspection/ERC165Checker.sol#102)
    State variables written after the call(s):
    - _lockFunction(address(_vault),Functions.TCAPORACLE) (contracts/Orchestrator.sol#347)
        - timelock[address(_contract)][_fn] = 0 (contracts/Orchestrator.sol#195)
    - _lockFunction(address(_vault),Functions.TCAPORACLE) (contracts/Orchestrator.sol#347)
        - timelockValue[address(_contract)][_fn] = 0 (contracts/Orchestrator.sol#196)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1 ✓[0m
INFO:Detectors: ✓[93m
IVaultHandler.addCollateral(uint256) (contracts/IVaultHandler.sol#314-326) ignores return value by
collateralContract.transferFrom(msg.sender,address(this),_amount) (contracts/IVaultHandler.sol#322)
IVaultHandler.removeCollateral(uint256) (contracts/IVaultHandler.sol#333-356) ignores return value by
collateralContract.transfer(msg.sender,_amount) (contracts/IVaultHandler.sol#354)
IVaultHandler.liquidateVault(uint256,uint256) (contracts/IVaultHandler.sol#407-428) ignores return value by
collateralContract.transfer(msg.sender,reward) (contracts/IVaultHandler.sol#426)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return ✓[0m
INFO:Detectors: ✓[92m
ERC20.constructor(string,string).name (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#57) shadows:
    - ERC20.name() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#66-68) (function)
ERC20.constructor(string,string).symbol (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#57) shadows:
    - ERC20.symbol() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#74-76) (function)
IVaultHandler.mint(uint256).requiredCollateral (contracts/IVaultHandler.sol#372) shadows:
    - IVaultHandler.requiredCollateral(uint256) (contracts/IVaultHandler.sol#482-496) (function)
IVaultHandler.liquidateVault(uint256,uint256).requiredCollateral (contracts/IVaultHandler.sol#407) shadows:
    - IVaultHandler.requiredCollateral(uint256) (contracts/IVaultHandler.sol#482-496) (function)
TCAP.constructor(string,string,uint256,Orchestrator)._name (contracts/TCAP.sol#41) shadows:
    - ERC20._name (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#44) (state variable)
TCAP.constructor(string,string,uint256,Orchestrator)._symbol (contracts/TCAP.sol#42) shadows:
    - ERC20._symbol (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#45) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing ✓[0m
INFO:Detectors: ✓[92m
Reentrancy in IVaultHandler.addCollateral(uint256) (contracts/IVaultHandler.sol#314-326):
    External calls:
    - collateralContract.transferFrom(msg.sender,address(this),_amount) (contracts/IVaultHandler.sol#322)
    State variables written after the call(s):
    - vault.Collateral = vault.Collateral.add(_amount) (contracts/IVaultHandler.sol#324)
Reentrancy in ETHVaultHandler.addCollateralETH() (contracts/ETHVaultHandler.sol#20-32):
    External calls:
    - IWETH(address(collateralContract)).deposit{value: msg.value}() (contracts/ETHVaultHandler.sol#28)
    State variables written after the call(s):
    - vault.Collateral = vault.Collateral.add(msg.value) (contracts/ETHVaultHandler.sol#30)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2 ✓[0m
INFO:Detectors: ✓[92m
Reentrancy in IVaultHandler.addCollateral(uint256) (contracts/IVaultHandler.sol#314-326):
    External calls:
    - collateralContract.transferFrom(msg.sender,address(this),_amount) (contracts/IVaultHandler.sol#322)
    Event emitted after the call(s):
```

```
    - LogAddCollateral(msg.sender,vault.Id,_amount) (contracts/IVaultHandler.sol#325)
Reentrancy in ETHVaultHandler.addCollateralETH() (contracts/ETHVaultHandler.sol#20-32):
    External calls:
    - IWETH(address(collateralContract)).deposit{value: msg.value}() (contracts/ETHVaultHandler.sol#28)
    Event emitted after the call(s):
    - LogAddCollateral(msg.sender,vault.Id,msg.value) (contracts/ETHVaultHandler.sol#31)
Reentrancy in IVaultHandler.burn(uint256) (contracts/IVaultHandler.sol#388-401):
    External calls:
    - _burn(vault.Id,_amount) (contracts/IVaultHandler.sol#399)
        - TCAPToken.burn(msg.sender,_amount) (contracts/IVaultHandler.sol#650)
    Event emitted after the call(s):
    - LogBurn(msg.sender,vault.Id,_amount) (contracts/IVaultHandler.sol#400)
Reentrancy in IVaultHandler.liquidateVault(uint256,uint256) (contracts/IVaultHandler.sol#407-428):
    External calls:
    - _burn(vault.Id,requiredCollateral) (contracts/IVaultHandler.sol#424)
        - TCAPToken.burn(msg.sender,_amount) (contracts/IVaultHandler.sol#650)
    - collateralContract.transfer(msg.sender,reward) (contracts/IVaultHandler.sol#426)
    Event emitted after the call(s):
    - LogLiquidateVault(vault.Id,msg.sender,req,reward) (contracts/IVaultHandler.sol#427)
Reentrancy in IVaultHandler.mint(uint256) (contracts/IVaultHandler.sol#363-381):
    External calls:
    - TCAPToken.mint(msg.sender,_amount) (contracts/IVaultHandler.sol#379)
    Event emitted after the call(s):
    - LogMint(msg.sender,vault.Id,_amount) (contracts/IVaultHandler.sol#380)
Reentrancy in IVaultHandler.removeCollateral(uint256) (contracts/IVaultHandler.sol#333-356):
    External calls:
    - collateralContract.transfer(msg.sender,_amount) (contracts/IVaultHandler.sol#354)
    Event emitted after the call(s):
    - LogRemoveCollateral(msg.sender,vault.Id,_amount) (contracts/IVaultHandler.sol#355)
Reentrancy in ETHVaultHandler.removeCollateralETH(uint256) (contracts/ETHVaultHandler.sol#39-63):
    External calls:
    - IWETH(address(collateralContract)).withdraw(_amount) (contracts/ETHVaultHandler.sol#60)
    - safeTransferETH(msg.sender,_amount) (contracts/ETHVaultHandler.sol#61)
        - (success) = to.call{value: value}(new bytes(0)) (contracts/ETHVaultHandler.sol#69)
    External calls sending eth:
    - safeTransferETH(msg.sender,_amount) (contracts/ETHVaultHandler.sol#61)
        - (success) = to.call{value: value}(new bytes(0)) (contracts/ETHVaultHandler.sol#69)
    Event emitted after the call(s):
    - LogRemoveCollateral(msg.sender,vault.Id,_amount) (contracts/ETHVaultHandler.sol#62)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3 ✓[0m
INFO:Detectors: ✓[92m
Address.isContract(address) (node_modules/@openzeppelin/contracts/utils/Address.sol#26-35) uses assembly
    - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#33)
Address._functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#119-140) uses assembly
    - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#132-135)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage ✓[0m
INFO:Detectors: ✓[92m
Different versions of Solidity is used in :
    - Version used: ['>=0.6.0', '>=0.6.8', '^0.6.0', '^0.6.2', '^0.6.8']
    - >=0.6.0 (node_modules/@chainlink/contracts/src/v0.6/interfaces/AggregatorV3Interface.sol#1)
    - ^0.6.0 (node_modules/@openzeppelin/contracts/GSN/Context.sol#3)
    - ^0.6.0 (node_modules/@openzeppelin/contracts/access/AccessControl.sol#3)
    - ^0.6.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3)
    - ^0.6.2 (node_modules/@openzeppelin/contracts/introspection/ERC165Checker.sol#3)
    - ^0.6.0 (node_modules/@openzeppelin/contracts/introspection/IERC165.sol#3)
    - ^0.6.0 (node_modules/@openzeppelin/contracts/math/SafeMath.sol#3)
    - ^0.6.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#3)
    - ^0.6.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3)
    - ^0.6.2 (node_modules/@openzeppelin/contracts/utils/Address.sol#3)
    - ^0.6.0 (node_modules/@openzeppelin/contracts/utils/Counters.sol#3)
    - ^0.6.0 (node_modules/@openzeppelin/contracts/utils/EnumerableSet.sol#3)
    - ^0.6.0 (node_modules/@openzeppelin/contracts/utils/Pausable.sol#3)
    - ^0.6.0 (node_modules/@openzeppelin/contracts/utils/ReentrancyGuard.sol#3)
    - ^0.6.0 (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#3)
    - ^0.6.8 (contracts/ERC20VaultHandler.sol#2)
    - ^0.6.8 (contracts/ETHVaultHandler.sol#2)
    - ^0.6.8 (contracts/IVaultHandler.sol#2)
    - >=0.6.8 (contracts/IWETH.sol#2)
    - ^0.6.8 (contracts/Orchestrator.sol#2)
    - ^0.6.8 (contracts/TCAP.sol#2)
    - >=0.6.0 (contracts/mocks/AggregatorInterface.sol#2)
    - >=0.6.0 (contracts/mocks/AggregatorInterfaceStable.sol#2)
    - >=0.6.0 (contracts/mocks/AggregatorInterfaceTCAP.sol#2)
    - ^0.6.8 (contracts/mocks/DAI.sol#4)
    - ^0.6.8 (contracts/mocks/USDC.sol#4)
    - ^0.6.8 (contracts/mocks/USDT.sol#4)
    - ^0.6.8 (contracts/mocks/WBTC.sol#4)
    - ^0.6.8 (contracts/mocks/WETH.sol#17)
    - ^0.6.8 (contracts/oracles/ChainlinkOracle.sol#2)
    - ^0.6.8 (contracts/oracles/TcapOracle.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used ✓[0m
INFO:Detectors: ✓[92m
Pragma version>=0.6.0 (node_modules/@chainlink/contracts/src/v0.6/interfaces/AggregatorV3Interface.sol#1) allows old versions
Pragma version^0.6.0 (node_modules/@openzeppelin/contracts/GSN/Context.sol#3) allows old versions
Pragma version^0.6.0 (node_modules/@openzeppelin/contracts/access/AccessControl.sol#3) allows old versions
Pragma version^0.6.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3) allows old versions
Pragma version^0.6.2 (node_modules/@openzeppelin/contracts/introspection/ERC165Checker.sol#3) allows old versions
Pragma version^0.6.0 (node_modules/@openzeppelin/contracts/introspection/IERC165.sol#3) allows old versions
Pragma version^0.6.0 (node_modules/@openzeppelin/contracts/math/SafeMath.sol#3) allows old versions
Pragma version^0.6.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#3) allows old versions
Pragma version^0.6.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3) allows old versions
Pragma version^0.6.2 (node_modules/@openzeppelin/contracts/utils/Address.sol#3) allows old versions
Pragma version^0.6.0 (node_modules/@openzeppelin/contracts/utils/Counters.sol#3) allows old versions
Pragma version^0.6.0 (node_modules/@openzeppelin/contracts/utils/EnumerableSet.sol#3) allows old versions
Pragma version^0.6.0 (node_modules/@openzeppelin/contracts/utils/Pausable.sol#3) allows old versions
Pragma version^0.6.0 (node_modules/@openzeppelin/contracts/utils/ReentrancyGuard.sol#3) allows old versions
Pragma version^0.6.0 (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#3) allows old versions
Pragma version>=0.6.0 (contracts/mocks/AggregatorInterface.sol#2) allows old versions
Pragma version>=0.6.0 (contracts/mocks/AggregatorInterfaceStable.sol#2) allows old versions
Pragma version>=0.6.0 (contracts/mocks/AggregatorInterfaceTCAP.sol#2) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity ✓[0m
INFO:Detectors: ✓[92m
Low level call in ERC165Checker._callERC165SupportsInterface(address,bytes4)
(node_modules/@openzeppelin/contracts/introspection/ERC165Checker.sol#96-105):
    - (success,result) = account.staticcall{gas: 30000}(encodedParams)
(node_modules/@openzeppelin/contracts/introspection/ERC165Checker.sol#102)
Low level call in Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#53-59):
    - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts/utils/Address.sol#57)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#119-
140):
    - (success,returndata) = target.call{value: weiValue}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#123)
Low level call in ETHVaultHandler.safeTransferETH(address,uint256) (contracts/ETHVaultHandler.sol#68-71):
    - (success) = to.call{value: value}(new bytes(0)) (contracts/ETHVaultHandler.sol#69)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls ✓[0m
INFO:Detectors: ✓[92m
Parameter ETHVaultHandler.removeCollateralETH(uint256)._amount (contracts/ETHVaultHandler.sol#39) is not in mixedCase
Parameter IVaultHandler.initialize(uint256,uint256,uint256,uint256,address,TCAP,address,address,address)._divisor
(contracts/IVaultHandler.sol#173) is not in mixedCase
Parameter IVaultHandler.initialize(uint256,uint256,uint256,uint256,address,TCAP,address,address,address)._ratio
(contracts/IVaultHandler.sol#174) is not in mixedCase
Parameter IVaultHandler.initialize(uint256,uint256,uint256,uint256,address,TCAP,address,address,address)._burnFee
(contracts/IVaultHandler.sol#175) is not in mixedCase
```

```
Parameter IVaultHandler.initialize(uint256,uint256,uint256,uint256,address,TCAP,address,address,address)._liquidationPenalty
(contracts/IVaultHandler.sol#176) is not in mixedCase
Parameter IVaultHandler.initialize(uint256,uint256,uint256,uint256,address,TCAP,address,address,address)._tcapOracle
(contracts/IVaultHandler.sol#177) is not in mixedCase
Parameter IVaultHandler.initialize(uint256,uint256,uint256,uint256,address,TCAP,address,address,address)._tcapAddress
(contracts/IVaultHandler.sol#178) is not in mixedCase
Parameter IVaultHandler.initialize(uint256,uint256,uint256,uint256,address,TCAP,address,address,address)._collateralAddress
(contracts/IVaultHandler.sol#179) is not in mixedCase
Parameter IVaultHandler.initialize(uint256,uint256,uint256,uint256,address,TCAP,address,address,address)._collateralOracle
(contracts/IVaultHandler.sol#180) is not in mixedCase
Parameter IVaultHandler.initialize(uint256,uint256,uint256,uint256,address,TCAP,address,address,address)._ethOracle
(contracts/IVaultHandler.sol#181) is not in mixedCase
Parameter IVaultHandler.setTCAPContract(TCAP)._TCAPToken (contracts/IVaultHandler.sol#199) is not in mixedCase
Parameter IVaultHandler.setTCAPOracle(ChainlinkOracle)._oracle (contracts/IVaultHandler.sol#209) is not in mixedCase
Parameter IVaultHandler.setCollateralContract(IERC20)._collateralContract (contracts/IVaultHandler.sol#219) is not in mixedCase
Parameter IVaultHandler.setCollateralPriceOracle(ChainlinkOracle)._collateral (contracts/IVaultHandler.sol#233) is not in mixedCase
Parameter IVaultHandler.setETHPriceOracle(ChainlinkOracle)._ethPriceOracle (contracts/IVaultHandler.sol#246) is not in mixedCase
Parameter IVaultHandler.setDivisor(uint256)._divisor (contracts/IVaultHandler.sol#256) is not in mixedCase
Parameter IVaultHandler.setRatio(uint256)._ratio (contracts/IVaultHandler.sol#266) is not in mixedCase
Parameter IVaultHandler.setBurnFee(uint256)._burnFee (contracts/IVaultHandler.sol#276) is not in mixedCase
Parameter IVaultHandler.setLiquidationPenalty(uint256)._liquidationPenalty (contracts/IVaultHandler.sol#286) is not in mixedCase
Parameter IVaultHandler.addCollateral(uint256)._amount (contracts/IVaultHandler.sol#314) is not in mixedCase
Parameter IVaultHandler.removeCollateral(uint256)._amount (contracts/IVaultHandler.sol#333) is not in mixedCase
Parameter IVaultHandler.mint(uint256)._amount (contracts/IVaultHandler.sol#363) is not in mixedCase
Parameter IVaultHandler.burn(uint256)._amount (contracts/IVaultHandler.sol#388) is not in mixedCase
Parameter IVaultHandler.liquidateVault(uint256,uint256)._vaultId (contracts/IVaultHandler.sol#407) is not in mixedCase
Function IVaultHandler.TCAPPrice() (contracts/IVaultHandler.sol#463-468) is not in mixedCase
Parameter IVaultHandler.requiredCollateral(uint256)._amount (contracts/IVaultHandler.sol#482) is not in mixedCase
Parameter IVaultHandler.requiredLiquidationCollateral(uint256)._vaultId (contracts/IVaultHandler.sol#512) is not in mixedCase
Parameter IVaultHandler.liquidationReward(uint256)._vaultId (contracts/IVaultHandler.sol#545) is not in mixedCase
Parameter IVaultHandler.getVault(uint256)._id (contracts/IVaultHandler.sol#566) is not in mixedCase
Parameter IVaultHandler.getVaultRatio(uint256)._vaultId (contracts/IVaultHandler.sol#592) is not in mixedCase
Parameter IVaultHandler.getFee(uint256)._amount (contracts/IVaultHandler.sol#625) is not in mixedCase
Variable IVaultHandler.TCAPToken (contracts/IVaultHandler.sol#91) is not in mixedCase
Variable IVaultHandler.ETHPriceOracle (contracts/IVaultHandler.sol#99) is not in mixedCase
Parameter Orchestrator.initializeVault(IVaultHandler,uint256,uint256,uint256,uint256,address,TCAP,address,address,address)._vault
(contracts/Orchestrator.sol#141) is not in mixedCase
Parameter Orchestrator.initializeVault(IVaultHandler,uint256,uint256,uint256,uint256,address,TCAP,address,address,address)._divisor
(contracts/Orchestrator.sol#142) is not in mixedCase
Parameter Orchestrator.initializeVault(IVaultHandler,uint256,uint256,uint256,uint256,address,TCAP,address,address,address)._ratio
(contracts/Orchestrator.sol#143) is not in mixedCase
Parameter Orchestrator.initializeVault(IVaultHandler,uint256,uint256,uint256,uint256,address,TCAP,address,address,address)._burnFee
(contracts/Orchestrator.sol#144) is not in mixedCase
Parameter Orchestrator.initializeVault(IVaultHandler,uint256,uint256,uint256,uint256,address,TCAP,address,address,address)._liquidationPenalty
(contracts/Orchestrator.sol#145) is not in mixedCase
Parameter Orchestrator.initializeVault(IVaultHandler,uint256,uint256,uint256,uint256,address,TCAP,address,address,address)._tcapOracle
(contracts/Orchestrator.sol#146) is not in mixedCase
Parameter Orchestrator.initializeVault(IVaultHandler,uint256,uint256,uint256,uint256,address,TCAP,address,address,address)._tcapAddress
(contracts/Orchestrator.sol#147) is not in mixedCase
Parameter Orchestrator.initializeVault(IVaultHandler,uint256,uint256,uint256,uint256,address,TCAP,address,address,address)._collateralAddress
(contracts/Orchestrator.sol#148) is not in mixedCase
Parameter Orchestrator.initializeVault(IVaultHandler,uint256,uint256,uint256,uint256,address,TCAP,address,address,address)._collateralOracle
(contracts/Orchestrator.sol#149) is not in mixedCase
Parameter Orchestrator.initializeVault(IVaultHandler,uint256,uint256,uint256,uint256,address,TCAP,address,address,address)._ethOracle
(contracts/Orchestrator.sol#150) is not in mixedCase
Parameter Orchestrator.unlockFunction(address,Orchestrator.Functions,bytes32)._contract (contracts/Orchestrator.sol#179) is not in mixedCase
Parameter Orchestrator.unlockFunction(address,Orchestrator.Functions,bytes32)._fn (contracts/Orchestrator.sol#180) is not in mixedCase
Parameter Orchestrator.unlockFunction(address,Orchestrator.Functions,bytes32)._hash (contracts/Orchestrator.sol#181) is not in mixedCase
Parameter Orchestrator.lockVaultFunction(IVaultHandler,Orchestrator.Functions)._vault (contracts/Orchestrator.sol#206) is not in mixedCase
Parameter Orchestrator.lockVaultFunction(IVaultHandler,Orchestrator.Functions)._fn (contracts/Orchestrator.sol#206) is not in mixedCase
Parameter Orchestrator.setDivisor(IVaultHandler,uint256)._vault (contracts/Orchestrator.sol#221) is not in mixedCase
Parameter Orchestrator.setDivisor(IVaultHandler,uint256)._divisor (contracts/Orchestrator.sol#221) is not in mixedCase
Parameter Orchestrator.setRatio(IVaultHandler,uint256)._vault (contracts/Orchestrator.sol#243) is not in mixedCase
Parameter Orchestrator.setRatio(IVaultHandler,uint256)._ratio (contracts/Orchestrator.sol#243) is not in mixedCase
Parameter Orchestrator.setBurnFee(IVaultHandler,uint256)._vault (contracts/Orchestrator.sol#265) is not in mixedCase
Parameter Orchestrator.setBurnFee(IVaultHandler,uint256)._burnFee (contracts/Orchestrator.sol#265) is not in mixedCase
Parameter Orchestrator.setLiquidationPenalty(IVaultHandler,uint256)._vault (contracts/Orchestrator.sol#288) is not in mixedCase
Parameter Orchestrator.setLiquidationPenalty(IVaultHandler,uint256)._liquidationPenalty (contracts/Orchestrator.sol#289) is not in mixedCase
Parameter Orchestrator.setTCAP(IVaultHandler,TCAP)._vault (contracts/Orchestrator.sol#312) is not in mixedCase
Parameter Orchestrator.setTCAP(IVaultHandler,TCAP)._tcap (contracts/Orchestrator.sol#312) is not in mixedCase
Parameter Orchestrator.setTCAPOracle(IVaultHandler,address)._vault (contracts/Orchestrator.sol#335) is not in mixedCase
Parameter Orchestrator.setTCAPOracle(IVaultHandler,address)._tcapOracle (contracts/Orchestrator.sol#335) is not in mixedCase
Parameter Orchestrator.setCollateral(IVaultHandler,IERC20)._vault (contracts/Orchestrator.sol#358) is not in mixedCase
Parameter Orchestrator.setCollateral(IVaultHandler,IERC20)._collateral (contracts/Orchestrator.sol#358) is not in mixedCase
Parameter Orchestrator.setCollateralOracle(IVaultHandler,address)._vault (contracts/Orchestrator.sol#380) is not in mixedCase
Parameter Orchestrator.setCollateralOracle(IVaultHandler,address)._collateralOracle (contracts/Orchestrator.sol#380) is not in mixedCase
Parameter Orchestrator.setETHOracle(IVaultHandler,address)._vault (contracts/Orchestrator.sol#403) is not in mixedCase
Parameter Orchestrator.setETHOracle(IVaultHandler,address)._ethOracles (contracts/Orchestrator.sol#403) is not in mixedCase
Parameter Orchestrator.pauseVault(IVaultHandler)._vault (contracts/Orchestrator.sol#424) is not in mixedCase
Parameter Orchestrator.unpauseVault(IVaultHandler)._vault (contracts/Orchestrator.sol#438) is not in mixedCase
Parameter Orchestrator.retrieveVaultFees(IVaultHandler)._vault (contracts/Orchestrator.sol#452) is not in mixedCase
Parameter Orchestrator.enableTCAPCap(TCAP,bool)._tcap (contracts/Orchestrator.sol#477) is not in mixedCase
Parameter Orchestrator.enableTCAPCap(TCAP,bool)._enable (contracts/Orchestrator.sol#477) is not in mixedCase
Parameter Orchestrator.setTCAPCap(TCAP,uint256)._tcap (contracts/Orchestrator.sol#498) is not in mixedCase
Parameter Orchestrator.setTCAPCap(TCAP,uint256)._cap (contracts/Orchestrator.sol#498) is not in mixedCase
Parameter Orchestrator.addTCAPVault(TCAP,IVaultHandler)._tcap (contracts/Orchestrator.sol#520) is not in mixedCase
Parameter Orchestrator.addTCAPVault(TCAP,IVaultHandler)._vault (contracts/Orchestrator.sol#520) is not in mixedCase
Parameter TCAP.addTokenHandler(address)._handler (contracts/TCAP.sol#61) is not in mixedCase
Parameter TCAP.mint(address,uint256)._account (contracts/TCAP.sol#72) is not in mixedCase
Parameter TCAP.mint(address,uint256)._amount (contracts/TCAP.sol#72) is not in mixedCase
Parameter TCAP.burn(address,uint256)._account (contracts/TCAP.sol#82) is not in mixedCase
Parameter TCAP.burn(address,uint256)._amount (contracts/TCAP.sol#82) is not in mixedCase
Parameter TCAP.setCap(uint256)._cap (contracts/TCAP.sol#93) is not in mixedCase
Parameter TCAP.enableCap(bool)._enable (contracts/TCAP.sol#104) is not in mixedCase
Parameter AggregatorInterfaceTCAP.setLatestAnswer(int256)._tcap (contracts/mocks/AggregatorInterfaceTCAP.sol#26) is not in mixedCase
Parameter DAI.mint(address,uint256)._account (contracts/mocks/DAI.sol#11) is not in mixedCase
Parameter DAI.mint(address,uint256)._amount (contracts/mocks/DAI.sol#11) is not in mixedCase
Parameter DAI.burn(address,uint256)._account (contracts/mocks/DAI.sol#15) is not in mixedCase
Parameter DAI.burn(address,uint256)._amount (contracts/mocks/DAI.sol#15) is not in mixedCase
Parameter USDC.mint(address,uint256)._account (contracts/mocks/USDC.sol#11) is not in mixedCase
Parameter USDC.mint(address,uint256)._amount (contracts/mocks/USDC.sol#11) is not in mixedCase
Parameter USDC.burn(address,uint256)._account (contracts/mocks/USDC.sol#15) is not in mixedCase
Parameter USDC.burn(address,uint256)._amount (contracts/mocks/USDC.sol#15) is not in mixedCase
Parameter USDT.mint(address,uint256)._account (contracts/mocks/USDT.sol#11) is not in mixedCase
Parameter USDT.mint(address,uint256)._amount (contracts/mocks/USDT.sol#11) is not in mixedCase
Parameter USDT.burn(address,uint256)._account (contracts/mocks/USDT.sol#15) is not in mixedCase
Parameter USDT.burn(address,uint256)._amount (contracts/mocks/USDT.sol#15) is not in mixedCase
Parameter WBTC.mint(address,uint256)._account (contracts/mocks/WBTC.sol#11) is not in mixedCase
Parameter WBTC.mint(address,uint256)._amount (contracts/mocks/WBTC.sol#11) is not in mixedCase
Parameter WBTC.burn(address,uint256)._account (contracts/mocks/WBTC.sol#15) is not in mixedCase
Parameter WBTC.burn(address,uint256)._amount (contracts/mocks/WBTC.sol#15) is not in mixedCase
Parameter ChainlinkOracle.setReferenceContract(address)._aggregator (contracts/oracles/ChainlinkOracle.sol#42) is not in mixedCase
Parameter ChainlinkOracle.getRound(uint80)._id (contracts/oracles/ChainlinkOracle.sol#83) is not in mixedCase
Parameter ChainlinkOracle.getPreviousAnswer(uint80)._id (contracts/oracles/ChainlinkOracle.sol#118) is not in mixedCase
Parameter ChainlinkOracle.getPreviousTimestamp(uint80)._id (contracts/oracles/ChainlinkOracle.sol#129) is not in mixedCase
Parameter TcapOracle.setLatestAnswer(uint256)._tcap (contracts/oracles/TcapOracle.sol#9) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformity-to-solidity-naming-conventions ✓[0m
INFO:Detectors: ✓[92m
Reentrancy in IVaultHandler.retrieveFees() (contracts/IVaultHandler.sol#447-451):
    External calls:
    - address(owner()).transfer(amount) (contracts/IVaultHandler.sol#449)
    Event emitted after the call(s):
    - LogRetrieveFees(msg.sender,amount) (contracts/IVaultHandler.sol#450)
Reentrancy in WETH.withdraw(uint256) (contracts/mocks/WETH.sol#40-45):
    External calls:
```

```
    - msg.sender.transfer(wad) (contracts/mocks/WETH.sol#43)
    Event emitted after the call(s):
    - Withdrawal(msg.sender,wad) (contracts/mocks/WETH.sol#44)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4 ✓[0m
INFO:Detectors: ✓[92m
IVaultHandler.TCAPPrice() (contracts/IVaultHandler.sol#463-468) uses literals with too many digits:
    - totalMarketPrice = tcapOracle.getLatestAnswer().toUint256().mul(10000000000) (contracts/IVaultHandler.sol#464-466)
IVaultHandler.requiredCollateral(uint256) (contracts/IVaultHandler.sol#482-496) uses literals with too many digits:
    - collateralPrice = collateralPriceOracle.getLatestAnswer().toUint256().mul(10000000000) (contracts/IVaultHandler.sol#489-492)
IVaultHandler.requiredLiquidationCollateral(uint256) (contracts/IVaultHandler.sol#512-533) uses literals with too many digits:
    - collateralPrice = collateralPriceOracle.getLatestAnswer().toUint256().mul(10000000000) (contracts/IVaultHandler.sol#520-523)
IVaultHandler.liquidationReward(uint256) (contracts/IVaultHandler.sol#545-559) uses literals with too many digits:
    - collateralPrice = collateralPriceOracle.getLatestAnswer().toUint256().mul(10000000000) (contracts/IVaultHandler.sol#553-556)
IVaultHandler.getVaultRatio(uint256) (contracts/IVaultHandler.sol#592-612) uses literals with too many digits:
    - collateralPrice = collateralPriceOracle.getLatestAnswer().toUint256().mul(10000000000) (contracts/IVaultHandler.sol#602-605)
IVaultHandler.getFee(uint256) (contracts/IVaultHandler.sol#625-630) uses literals with too many digits:
    - ethPrice = ETHPriceOracle.getLatestAnswer().toUint256().mul(10000000000) (contracts/IVaultHandler.sol#626-628)
AggregatorInterfaceStable.slitherConstructorVariables() (contracts/mocks/AggregatorInterfaceStable.sol#4-36) uses literals with too many
digits:
    - value = 100000000 (contracts/mocks/AggregatorInterfaceStable.sol#5)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits ✓[0m
INFO:Detectors: ✓[92m
AggregatorInterface.value (contracts/mocks/AggregatorInterface.sol#5) should be constant
AggregatorInterfaceStable.value (contracts/mocks/AggregatorInterfaceStable.sol#5) should be constant
WETH.decimals (contracts/mocks/WETH.sol#22) should be constant
WETH.name (contracts/mocks/WETH.sol#20) should be constant
WETH.symbol (contracts/mocks/WETH.sol#21) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant ✓[0m
INFO:Detectors: ✓[92m
getRoleMemberCount(bytes32) should be declared external:
    - AccessControl.getRoleMemberCount(bytes32) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#95-97)
getRoleMember(bytes32,uint256) should be declared external:
    - AccessControl.getRoleMember(bytes32,uint256) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#111-113)
getRoleAdmin(bytes32) should be declared external:
    - AccessControl.getRoleAdmin(bytes32) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#121-123)
grantRole(bytes32,address) should be declared external:
    - AccessControl.grantRole(bytes32,address) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#135-139)
revokeRole(bytes32,address) should be declared external:
    - AccessControl.revokeRole(bytes32,address) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#150-154)
renounceRole(bytes32,address) should be declared external:
    - AccessControl.renounceRole(bytes32,address) (node_modules/@openzeppelin/contracts/access/AccessControl.sol#170-174)
renounceOwnership() should be declared external:
    - Ownable.renounceOwnership() (node_modules/@openzeppelin/contracts/access/Ownable.sol#54-57)
name() should be declared external:
    - ERC20.name() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#66-68)
symbol() should be declared external:
    - ERC20.symbol() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#74-76)
decimals() should be declared external:
    - ERC20.decimals() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#91-93)
balanceOf(address) should be declared external:
    - ERC20.balanceOf(address) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#105-107)
transfer(address,uint256) should be declared external:
    - ERC20.transfer(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#117-120)
allowance(address,address) should be declared external:
    - ERC20.allowance(address,address) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#125-127)
approve(address,uint256) should be declared external:
    - ERC20.approve(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#136-139)
transferFrom(address,address,uint256) should be declared external:
    - ERC20.transferFrom(address,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#153-157)
increaseAllowance(address,uint256) should be declared external:
    - ERC20.increaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#171-174)
decreaseAllowance(address,uint256) should be declared external:
    - ERC20.decreaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#190-193)
paused() should be declared external:
    - Pausable.paused() (node_modules/@openzeppelin/contracts/utils/Pausable.sol#39-41)
addCollateralETH() should be declared external:
    - ETHVaultHandler.addCollateralETH() (contracts/ETHVaultHandler.sol#20-32)
removeCollateralETH(uint256) should be declared external:
    - ETHVaultHandler.removeCollateralETH(uint256) (contracts/ETHVaultHandler.sol#39-63)
initialize(uint256,uint256,uint256,uint256,address,TCAP,address,address,address) should be declared external:
    - IVaultHandler.initialize(uint256,uint256,uint256,uint256,address,TCAP,address,address,address) (contracts/IVaultHandler.sol#172-192)
setTCAPContract(TCAP) should be declared external:
    - IVaultHandler.setTCAPContract(TCAP) (contracts/IVaultHandler.sol#199-202)
setTCAPOracle(ChainlinkOracle) should be declared external:
    - IVaultHandler.setTCAPOracle(ChainlinkOracle) (contracts/IVaultHandler.sol#209-212)
setCollateralContract(IERC20) should be declared external:
    - IVaultHandler.setCollateralContract(IERC20) (contracts/IVaultHandler.sol#219-226)
setCollateralPriceOracle(ChainlinkOracle) should be declared external:
    - IVaultHandler.setCollateralPriceOracle(ChainlinkOracle) (contracts/IVaultHandler.sol#233-239)
setETHPriceOracle(ChainlinkOracle) should be declared external:
    - IVaultHandler.setETHPriceOracle(ChainlinkOracle) (contracts/IVaultHandler.sol#246-249)
setDivisor(uint256) should be declared external:
    - IVaultHandler.setDivisor(uint256) (contracts/IVaultHandler.sol#256-259)
setRatio(uint256) should be declared external:
    - IVaultHandler.setRatio(uint256) (contracts/IVaultHandler.sol#266-269)
setBurnFee(uint256) should be declared external:
    - IVaultHandler.setBurnFee(uint256) (contracts/IVaultHandler.sol#276-279)
setLiquidationPenalty(uint256) should be declared external:
    - IVaultHandler.setLiquidationPenalty(uint256) (contracts/IVaultHandler.sol#286-293)
createVault() should be declared external:
    - IVaultHandler.createVault() (contracts/IVaultHandler.sol#299-307)
addCollateral(uint256) should be declared external:
    - IVaultHandler.addCollateral(uint256) (contracts/IVaultHandler.sol#314-326)
removeCollateral(uint256) should be declared external:
    - IVaultHandler.removeCollateral(uint256) (contracts/IVaultHandler.sol#333-356)
mint(uint256) should be declared external:
    - IVaultHandler.mint(uint256) (contracts/IVaultHandler.sol#363-381)
burn(uint256) should be declared external:
    - IVaultHandler.burn(uint256) (contracts/IVaultHandler.sol#388-401)
liquidateVault(uint256,uint256) should be declared external:
    - IVaultHandler.liquidateVault(uint256,uint256) (contracts/IVaultHandler.sol#407-428)
pause() should be declared external:
    - IVaultHandler.pause() (contracts/IVaultHandler.sol#433-435)
unpause() should be declared external:
    - IVaultHandler.unpause() (contracts/IVaultHandler.sol#440-442)
retrieveFees() should be declared external:
    - IVaultHandler.retrieveFees() (contracts/IVaultHandler.sol#447-451)
getVault(uint256) should be declared external:
    - IVaultHandler.getVault(uint256) (contracts/IVaultHandler.sol#566-579)
initializeVault(IVaultHandler,uint256,uint256,uint256,uint256,address,TCAP,address,address,address) should be declared external:
    - Orchestrator.initializeVault(IVaultHandler,uint256,uint256,uint256,uint256,address,TCAP,address,address,address)
(contracts/Orchestrator.sol#140-168)
unlockFunction(address,Orchestrator.Functions,bytes32) should be declared external:
    - Orchestrator.unlockFunction(address,Orchestrator.Functions,bytes32) (contracts/Orchestrator.sol#178-186)
lockVaultFunction(IVaultHandler,Orchestrator.Functions) should be declared external:
    - Orchestrator.lockVaultFunction(IVaultHandler,Orchestrator.Functions) (contracts/Orchestrator.sol#206-211)
setDivisor(IVaultHandler,uint256) should be declared external:
    - Orchestrator.setDivisor(IVaultHandler,uint256) (contracts/Orchestrator.sol#221-233)
setRatio(IVaultHandler,uint256) should be declared external:
    - Orchestrator.setRatio(IVaultHandler,uint256) (contracts/Orchestrator.sol#243-255)
setBurnFee(IVaultHandler,uint256) should be declared external:
    - Orchestrator.setBurnFee(IVaultHandler,uint256) (contracts/Orchestrator.sol#265-277)
setLiquidationPenalty(IVaultHandler,uint256) should be declared external:
    - Orchestrator.setLiquidationPenalty(IVaultHandler,uint256) (contracts/Orchestrator.sol#287-302)
```

```
setTCAP(IVaultHandler,TCAP) should be declared external:
    - Orchestrator.setTCAP(IVaultHandler,TCAP) (contracts/Orchestrator.sol#312-325)
setTCAPOracle(IVaultHandler,address) should be declared external:
    - Orchestrator.setTCAPOracle(IVaultHandler,address) (contracts/Orchestrator.sol#335-348)
setCollateral(IVaultHandler,IERC20) should be declared external:
    - Orchestrator.setCollateral(IVaultHandler,IERC20) (contracts/Orchestrator.sol#358-370)
setCollateralOracle(IVaultHandler,address) should be declared external:
    - Orchestrator.setCollateralOracle(IVaultHandler,address) (contracts/Orchestrator.sol#380-393)
setETHOracle(IVaultHandler,address) should be declared external:
    - Orchestrator.setETHOracle(IVaultHandler,address) (contracts/Orchestrator.sol#403-416)
pauseVault(IVaultHandler) should be declared external:
    - Orchestrator.pauseVault(IVaultHandler) (contracts/Orchestrator.sol#424-430)
unpauseVault(IVaultHandler) should be declared external:
    - Orchestrator.unpauseVault(IVaultHandler) (contracts/Orchestrator.sol#438-444)
retrieveVaultFees(IVaultHandler) should be declared external:
    - Orchestrator.retrieveVaultFees(IVaultHandler) (contracts/Orchestrator.sol#452-458)
retrieveFees() should be declared external:
    - Orchestrator.retrieveFees() (contracts/Orchestrator.sol#465-468)
enableTCAPCap(TCAP,bool) should be declared external:
    - Orchestrator.enableTCAPCap(TCAP,bool) (contracts/Orchestrator.sol#477-489)
setTCAPCap(TCAP,uint256) should be declared external:
    - Orchestrator.setTCAPCap(TCAP,uint256) (contracts/Orchestrator.sol#498-510)
addTCAPVault(TCAP,IVaultHandler) should be declared external:
    - Orchestrator.addTCAPVault(TCAP,IVaultHandler) (contracts/Orchestrator.sol#520-527)
addTokenHandler(address) should be declared external:
    - TCAP.addTokenHandler(address) (contracts/TCAP.sol#61-64)
mint(address,uint256) should be declared external:
    - TCAP.mint(address,uint256) (contracts/TCAP.sol#72-74)
burn(address,uint256) should be declared external:
    - TCAP.burn(address,uint256) (contracts/TCAP.sol#82-84)
setCap(uint256) should be declared external:
    - TCAP.setCap(uint256) (contracts/TCAP.sol#93-96)
enableCap(bool) should be declared external:
    - TCAP.enableCap(bool) (contracts/TCAP.sol#104-107)
latestAnswer() should be declared external:
    - AggregatorInterface.latestAnswer() (contracts/mocks/AggregatorInterface.sol#7-9)
latestRoundData() should be declared external:
    - AggregatorInterface.latestRoundData() (contracts/mocks/AggregatorInterface.sol#11-24)
latestAnswer() should be declared external:
    - AggregatorInterfaceStable.latestAnswer() (contracts/mocks/AggregatorInterfaceStable.sol#7-9)
latestRoundData() should be declared external:
    - AggregatorInterfaceStable.latestRoundData() (contracts/mocks/AggregatorInterfaceStable.sol#11-24)
latestAnswer() should be declared external:
    - AggregatorInterfaceTCAP.latestAnswer() (contracts/mocks/AggregatorInterfaceTCAP.sol#7-9)
latestRoundData() should be declared external:
    - AggregatorInterfaceTCAP.latestRoundData() (contracts/mocks/AggregatorInterfaceTCAP.sol#11-24)
setLatestAnswer(int256) should be declared external:
    - AggregatorInterfaceTCAP.setLatestAnswer(int256) (contracts/mocks/AggregatorInterfaceTCAP.sol#26-28)
mint(address,uint256) should be declared external:
    - DAI.mint(address,uint256) (contracts/mocks/DAI.sol#11-13)
burn(address,uint256) should be declared external:
    - DAI.burn(address,uint256) (contracts/mocks/DAI.sol#15-17)
mint(address,uint256) should be declared external:
    - USDC.mint(address,uint256) (contracts/mocks/USDC.sol#11-13)
burn(address,uint256) should be declared external:
    - USDC.burn(address,uint256) (contracts/mocks/USDC.sol#15-17)
mint(address,uint256) should be declared external:
    - USDT.mint(address,uint256) (contracts/mocks/USDT.sol#11-13)
burn(address,uint256) should be declared external:
    - USDT.burn(address,uint256) (contracts/mocks/USDT.sol#15-17)
mint(address,uint256) should be declared external:
    - WBTC.mint(address,uint256) (contracts/mocks/WBTC.sol#11-13)
burn(address,uint256) should be declared external:
    - WBTC.burn(address,uint256) (contracts/mocks/WBTC.sol#15-17)
deposit() should be declared external:
    - WETH.deposit() (contracts/mocks/WETH.sol#35-38)
withdraw(uint256) should be declared external:
    - WETH.withdraw(uint256) (contracts/mocks/WETH.sol#40-45)
totalSupply() should be declared external:
    - WETH.totalSupply() (contracts/mocks/WETH.sol#47-49)
approve(address,uint256) should be declared external:
    - WETH.approve(address,uint256) (contracts/mocks/WETH.sol#51-55)
transfer(address,uint256) should be declared external:
    - WETH.transfer(address,uint256) (contracts/mocks/WETH.sol#57-59)
setReferenceContract(address) should be declared external:
    - ChainlinkOracle.setReferenceContract(address) (contracts/oracles/ChainlinkOracle.sol#42-44)
getLatestAnswer() should be declared external:
    - ChainlinkOracle.getLatestAnswer() (contracts/oracles/ChainlinkOracle.sol#50-52)
getLatestRound() should be declared external:
    - ChainlinkOracle.getLatestRound() (contracts/oracles/ChainlinkOracle.sol#57-77)
getRound(uint80) should be declared external:
    - ChainlinkOracle.getRound(uint80) (contracts/oracles/ChainlinkOracle.sol#83-103)
getLatestTimestamp() should be declared external:
    - ChainlinkOracle.getLatestTimestamp() (contracts/oracles/ChainlinkOracle.sol#108-111)
getPreviousAnswer(uint80) should be declared external:
    - ChainlinkOracle.getPreviousAnswer(uint80) (contracts/oracles/ChainlinkOracle.sol#118-122)
getPreviousTimestamp(uint80) should be declared external:
    - ChainlinkOracle.getPreviousTimestamp(uint80) (contracts/oracles/ChainlinkOracle.sol#129-133)
setLatestAnswer(uint256) should be declared external:
    - TcapOracle.setLatestAnswer(uint256) (contracts/oracles/TcapOracle.sol#9-11)
getLatestAnswer() should be declared external:
    - TcapOracle.getLatestAnswer() (contracts/oracles/TcapOracle.sol#13-15)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external ✓[0m
INFO:Slither:. analyzed (31 contracts with 46 detectors), 271 result(s) found
```

Code Documentation

1. Typo in IVaultHandler.sol::L582 `Collateral Ratio fo` -> `Collateral Ratio of`
2. Typo in IVaultHandler.sol::L585 `D - P` -> `D * P`
3. Under subsection `Liquidation Event` in page 6 of the current whitepaper you can see that the same mistake regarding `D - P` -> `D * P`
4. In IVaultHandler.sol, for the function `requiredCollateral`, the variable `r` is not defined in the spec and the comment.
5. In ChainlinkOracle.sol::L14 consider using a more descriptive name.
6. In ChainlinkOracle.sol::L9 wrong title.
7. In ChainlinkOracle.sol::L31, in TCAP.sol::L112, inOrchestrator.sol::L52 `it's` -> `is`.
8. In ChainlinkOracle.sol::L47,L55,L80 `referece` -> `reference`.
9. In ChainlinkOracle.sol::L80 `the latest round` -> `a given round`.
10. In ChainlinkOracle.sol::L114,L125 `the previous answer` -> `a previous answer`.
11. In TCAP.sol::L50,L52,L57 “the handler” implies there is only one handler.
12. In TCAP.sol, L58 and L78 are probably erroneous and do not make sense in context.
13. In IVaultHandler.sol, L101 to L104 are probably erroneous and do not make sense in context.

14. In IVaultHandler.sol, `vaultToUser` should be renamed as `userToVault`.
15. In IVaultHandler.sol::L131, remove "all".
16. In IVaultHandler.sol::L272, it seems like a mistake was made copy-pasting.
17. In IVaultHandler.sol::L329, `from collateral` -> `from vault`.
18. In IVaultHandler.sol, the usage of the term "liquidation collateral" to refer to TCAP tokens is confusing.
19. In IVaultHandler.sol::L582, `fo` -> `for`.
20. In IVaultHandler.sol::L618, `on ETH` -> `in wei`.
21. In Orchestrator.sol::L50, `constant` -> `function`.
22. In Orchestrator.sol::L191,L202, `unlocked` -> `locked`.
23. In Orchestrator.sol::L395-L416, `_ethOracles` -> `ethOracle`.
24. In Orchestrator.sol::L492, copy-paste mistake.
25. In Orchestrator.sol::L400 `if _vault and oracle` should be `if _vault and _ethOracles` instead.
26. In Orchestrator.sol::L377 `if _vault and oracle` should be `if _vault and _collateralOracle` instead.
27. In ETHVaultHandler.sol::L35, `from collateral` -> `from vault`.

Adherence to Best Practices

1. In Orchestrator.sol::L40 `_TIMELOCK` is set to private. Some projects choose to make pertinent information like this publicly accessible, consider setting it to public or having a view function for it.
2. In IVaultHandler.sol, naming of the function `requiredLiquidationCollateral()` can be misleading. This function (as documented in spec/comments) actually returns the amount of TCAP token needed to liquidate a vault instead of the amount of collateral asset needed. The same issue applies to the argument `requiredCollateral` in the function `liquidateVault()` and calculations in the function `liquidationReward()`.

Test Results

Test Suite Results

Tests were easy to run with the instructions and appeared comprehensive.

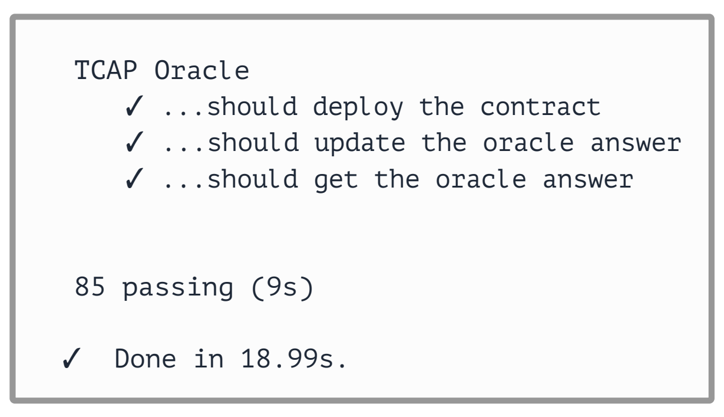
```
Chainlink Oracle
   ✓ ...should deploy the contract (217ms)
   ✓ ...should get the oracle answer

ERC20 Vault
   ✓ ...should deploy the contract (783ms)
   ✓ ...should return the token price
   ✓ ...should allow users to create a vault (84ms)
   ✓ ...should get vault by id
   ✓ ...should allow user to stake collateral (278ms)
   ✓ ...should allow user to retrieve unused collateral (181ms)
   ✓ ...should return the correct minimal collateral required (77ms)
   ✓ ...should allow user to mint tokens (273ms)
   ✓ ...should allow token transfers (59ms)
   ✓ ...shouldn't allow user to send tokens to tcap contract
   ✓ ...should allow users to get collateral ratio
   ✓ ...shouln't allow users to retrieve stake unless debt is paid (44ms)
   ✓ ...should calculate the burn fee (68ms)
   ✓ ...should allow users to burn tokens (204ms)
   ✓ ...should update the collateral ratio
   ✓ ...should allow users to retrieve stake when debt is paid (40ms)
   ✓ ...should allow owner to retrieve fees in the contract (64ms)
   ✓ ...should test liquidation requirements (154ms)
   ✓ ...should get the required collateral for liquidation (56ms)
   ✓ ...should get the liquidation reward (71ms)
   ✓ ...should allow liquidators to return profits (62ms)
   ✓ ...should allow users to liquidate users on vault ratio less than ratio (389ms)
   ✓ ...should allow owner to pause contract
   ✓ ...shouldn't allow contract calls if contract is paused
   ✓ ...should allow owner to unpause contract

ETH Vault
   ✓ ...should deploy the contract (436ms)
   ✓ ...should return the token price
   ✓ ...should allow users to create a vault (45ms)
   ✓ ...should get vault by id
   ✓ ...should allow user to stake weth collateral (181ms)
   ✓ ...should allow user to stake eth collateral (106ms)
   ✓ ...should allow user to retrieve unused collateral on eth (122ms)
   ✓ ...should allow user to retrieve unused collateral on weth (123ms)
   ✓ ...should return the correct minimal collateral required
   ✓ ...should allow user to mint tokens (169ms)
   ✓ ...should allow users to get collateral ratio
   ✓ ...shouln't allow users to retrieve stake unless debt is paid (38ms)
   ✓ ...should calculate the burn fee (52ms)
   ✓ ...should allow users to burn tokens (150ms)
   ✓ ...should update the collateral ratio
   ✓ ...should allow users to retrieve stake when debt is paid (40ms)
   ✓ ...should allow owner to retrieve fees in the contract (55ms)
   ✓ ...should test liquidation requirements (134ms)
   ✓ ...should get the required collateral for liquidation (50ms)
   ✓ ...should get the liquidation reward (78ms)
   ✓ ...should allow liquidators to return profits (83ms)
   ✓ ...should allow users to liquidate users on vault ratio less than ratio (386ms)
   ✓ ...should allow owner to pause contract
   ✓ ...shouldn't allow contract calls if contract is paused
   ✓ ...should allow owner to unpause contract

Orchestrator Contract
   ✓ ...should deploy the contract (477ms)
   ✓ ...should set the owner
   ✓ ...should validate the data on initialize (98ms)
   ✓ ...should initialize vault contracts (112ms)
   ✓ ...shouldn't allow a vault to initialize more than once
   ✓ ...should allow to unlock timelock for a function (101ms)
   ✓ ...should allow to lock again a function
   ✓ ...should set vault divisor (79ms)
   ✓ ...should set vault ratio (85ms)
   ✓ ...should set vault burn fee (78ms)
   ✓ ...should set vault liquidation penalty (76ms)
   ✓ ...should set vault TCAP Contract (107ms)
   ✓ ...should set vault TCAP Oracle Contract (110ms)
   ✓ ...should set vault Collateral Contract (123ms)
   ✓ ...should set vault Collateral Oracle Contract (108ms)
   ✓ ...should set vault ETH Oracle Contract (104ms)
   ✓ ...should pause the Vault
   ✓ ...should unpause the vault
   ✓ ...should be able to retrieve funds from vault
   ✓ ...should be able to send funds to owner of orchestrator
   ✓ ...should enable the TCAP cap (86ms)
   ✓ ...should set the TCAP cap (81ms)
   ✓ ...should add vault to TCAP token (48ms)

TCAP Token
   ✓ ...should deploy the contract (286ms)
   ✓ ...should set the correct initial values
   ✓ ...should have the ERC20 standard functions
   ✓ ...should allow to approve tokens
   ✓ ...shouldn't allow users to mint
   ✓ ...shouldn't allow users to burn
   ✓ ...should allow owner to add Handlers (71ms)
```



```
TCAP Oracle
   ✓ ...should deploy the contract
   ✓ ...should update the oracle answer
   ✓ ...should get the oracle answer


85 passing (9s)

✓  Done in 18.99s.
```

Appendix

File Signatures

The following are the SHA-256 hashes of the reviewed files. A file with a different SHA-256 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different SHA-256 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review.

Contracts

```
384ab2d04c383bd9a841ef9c68c6589a0f4d314fd602bd6954fb9bc389712691 ./contracts/ETHVaultHandler.sol
23e055aacfd91140278b312d07eac3552459e70544ab08a23534664cd4640c80 ./contracts/IVaultHandler.sol
ec34b55df835adedd0833f9371ea05a4716b463be2340d3ad69062f628b24f34 ./contracts/Orchestrator.sol
1076dd41cddd632925accba2b7aadf06d3c4a31be1802b28f4e27906214549be ./contracts/IWETH.sol
7df1ac613d9d6769831b267749a29463e23a384907112b4901726fb3d532262c ./contracts/ERC20VaultHandler.sol
ffe8c77d0892a3fe6807e100e62f7387acdce75c0d2106c6dd4338d65335111a ./contracts/TCAP.sol
0c05f91c5c27a348b184e64a0201b8caa9ee8b3aab1046fffeabaaa7237546ae ./contracts/oracles/TcapOracle.sol
a33a85b731458fdb062e36252767c5bcdf16297c39afc544c479d361c84b893f ./contracts/oracles/ChainlinkOracle.sol
cf4badabfc70c858b73d03b8855678f892ecd6f0b59e87ae46e34700accf8048 ./contracts/mocks/DAI.sol
471ea5016d9f03a1bb67261bd0cfde59a8643caeb7135b24d1ad48bcc218af5d ./contracts/mocks/WBTC.sol
33c0cf8fed142f08f4b310f6576cbf3f0619334e97889ef9f22d673e80cfd17b ./contracts/mocks/AggregatorInterfaceStable.sol
74a77be705b4a269e9527c823029a68fae0d5fcb0c8ebecb19803c82e1da2e6b ./contracts/mocks/USDC.sol
0f9420f9c8e5e3ac1c8423ccfe886cec6e488f30b6731fd389b0725f98d2be20 ./contracts/mocks/USDT.sol
011e2d8660d8e1e7e661b7b862da9a86c90b9af9fa5a2fcf5224688c970e2471 ./contracts/mocks/AggregatorInterface.sol
5e63aa1b3d176906cfbb4fed3e5e65d59a38399ed600b707dca03b8c918446af ./contracts/mocks/WETH.sol
017dcf5330e221fa01e122f9b935a2f6031d86b51d5bd0610f3143ebfc8f5aa6 ./contracts/mocks/AggregatorInterfaceTCAP.sol
```

Tests

```
52080258e92fc6fc418681fbd83854aaf50800c1497febc5eba60ba714665c8e ./test/Orchestrator.test.js
45deedc01288cafeb3e23e1581deed79e80db55f2610d4b71618ff9fb335bba8 ./test/ChainlinkOracle.test.js
341a1e0dd48c8f945a30d41813320eec5b9ba7a826a32853e0d905bfd8123e22 ./test/ETHVaultHandler.test.js
cf360b2ac21790cc9a81c4fa92a9e2f5c1f76d5fcc1d42b97aa132315e64b340 ./test/TCAP.test.js
1a59aa2f8769949465b57b2a0afc76e71ae88c75824771e6755a4f14575b4dfb ./test/TCapOracle.test.js
814705fea063feeda4bb8e6e0db5b926e378e22e8c492938308cdfb3d45e8bef ./test/ERCVaultHandler.test.js
```

Changelog

- 2020-10-31 - Initial report
- 2020-11-26 - Final report

About Quantstamp

Quantstamp is a Y Combinator-backed company that helps to secure blockchain platforms at scale using computer-aided reasoning tools, with a mission to help boost the adoption of this exponentially growing technology.

With over 1000 Google scholar citations and numerous published papers, Quantstamp's team has decades of combined experience in formal verification, static analysis, and software verification. Quantstamp has also developed a protocol to help smart contract developers and projects worldwide to perform cost-effective smart contract security scans.

To date, Quantstamp has protected $5B in digital asset risk from hackers and assisted dozens of blockchain projects globally through its white glove security assessment services. As an evangelist of the blockchain ecosystem, Quantstamp assists core infrastructure projects and leading community initiatives such as the Ethereum Community Fund to expedite the adoption of blockchain technology.

Quantstamp's collaborations with leading academic institutions such as the National University of Singapore and MIT (Massachusetts Institute of Technology) reflect our commitment to research, development, and enabling world-class blockchain security.