Quantstamp Security Assessment Certificate

QUANTSTAMP VERIFIED SECURITY CERTIFICATE

February 13th 2023 — Quantstamp Verified

Davos

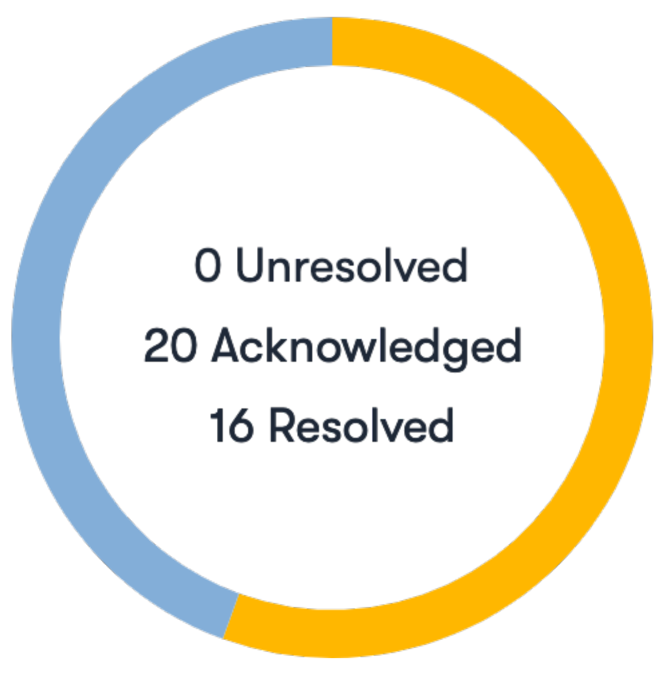
This audit report was prepared by Quantstamp, the leader in blockchain security.

Executive Summary

Type	Cross Chain Lending Protocol
Auditors	Souhail Mssassi, Research Engineer Guillermo Escobero, Security Auditor Jennifer Wu, Auditing Engineer
Timeline	2022-10-10 through 2022-12-02
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	Davos docs
Documentation Quality	Medium
Test Quality	Medium
Source Code	

Repository	Commit
davos-money/new-davos-smart-contracts	46973a1 initial audit

Total Issues	**36**	(16 Resolved)
High Risk Issues	**2**	(2 Resolved)
Medium Risk Issues	**8**	(4 Resolved)
Low Risk Issues	**12**	(5 Resolved)
Informational Risk Issues	**12**	(4 Resolved)
Undetermined Risk Issues	**2**	(1 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).
○ Fixed	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

Initial audit:
Through reviewing the code, we found **35 potential issues** of various levels of severity: 2 high-severity, 7 medium-severity, 13 low-severity, 11 informational-severity and 2 undetermined-severity . The high severity issues represent a real risk for Davos. We recommend addressing all the issues before deploying the code.

Fix review:
The majority of the issues were fixed, and we found two new issues (QSP-9, QSP-10), we recommend the team to fix them and also address the "Adherence to Specification", "Code Documentation" and "Adherence to Best Practices" sections recommendations.

Note: Sikka rebranded to *Davos*.

ID	Description	Severity	Status
QSP-1	Malicious Actor Can Deny User From Receiving Staking Rewards	High	Fixed
QSP-2	Loss of Strategy Funds During Withdrawal	High	Mitigated
QSP-3	Stale Collateral Price Feed Data	Medium	Mitigated
QSP-4	Loss of Stability Fee and Rewards During Payback	Medium	Acknowledged
QSP-5	Missing Pause Implementation	Medium	Fixed
QSP-6	Sikka Total Supply Limitation May Impair Liquidation Process	Medium	Acknowledged
QSP-7	Signature Malleability	Medium	Fixed
QSP-8	Pool Limit Can Be Bypassed	Medium	Fixed
QSP-9	Spot Price May Be Stale	Medium	Acknowledged
QSP-10	Unknown Oracle for Synthetic Assets	Medium	Acknowledged
QSP-11	Privileged Owner Can Prevent User From Collateral Withdrawal	Low	Acknowledged
QSP-12	Privileged Owner Can Withdraw User `aMATICc` From `CeVault`	Low	Acknowledged
QSP-13	Dilute Existing Lp Holders with Ratio Modification	Low	Acknowledged
QSP-14	Missing Uncage Function	Low	Fixed
QSP-15	Old Strategy Still Exist After Migration	Low	Fixed
QSP-16	Silent Failed ERC20 Transfer	Low	Mitigated
QSP-17	Missing Input Validation	Low	Fixed
QSP-18	Revocable Ownership	Low	Acknowledged
QSP-19	Inaccurate Accounting if Deflationary Tokens Are Accepted	Low	Acknowledged
QSP-20	Incorrect Value Emitted During Events	Low	Fixed
QSP-21	Potential Denial of Service when Claiming GIKKA Rewards	Low	Acknowledged
QSP-22	Oracle Decimal Assumption	Low	Acknowledged
QSP-23	Upgradeable Misconfigurations	Informational	Acknowledged
QSP-24	Borrow Sikka Before Claim to Maximize Gikka Rewards	Informational	Acknowledged
QSP-25	Loss of Gikka Rewards	Informational	Mitigated
QSP-26	Missing Events to Signal State Changes	Informational	Fixed
QSP-27	Upgradeable Contract Storage Gaps	Informational	Fixed
QSP-28	Potential Denial of Service Due to Block Gas Limit	Informational	Acknowledged
QSP-29	Potential Denial of Service From Stability Fee Collection	Informational	Acknowledged
QSP-30	Ignored Returned Value From Functions	Informational	Fixed
QSP-31	Risk of Killing Upgrades	Informational	Acknowledged
QSP-32	Usage `transfer()` and `send()` Should Be Avoided	Informational	Acknowledged
QSP-33	Clone-and-Own	Informational	Acknowledged
QSP-34	Unlocked Pragma	Informational	Acknowledged
QSP-35	Auction Maximum Sikka Amount Is Always Greater by 1	? Undetermined	Fixed
QSP-36	Borrowed Amount Is Always Greater by 100	? 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.

DISCLAIMER:
If the final commit hash provided by the client contains features that are not within the scope of the audit or an associated fix review, those features are excluded from consideration in this report.

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.

Findings

QSP-1 Malicious Actor Can Deny User From Receiving Staking Rewards

Severity: *High Risk*

Status: Fixed

File(s) affected: `colander.sol`

Description: `ColanderRewards.replenish()` is an external function that allows users to distribute additional rewards by transferring `_wad` to distribute among the `Sikka` stakers in `Colander`. The `rate` is recalculated if `endTime` has not been reached, using the remaining reward and the `spread` duration. Since `replenish()` is available to the public, a malicious actor can extend reward distribution and reduce the `rate` by invoking `replenish()` function multiple times with zero `_wad` or a minimal `_wad` amount. It is possible to reduce the `rate` to zero using `replenish()` and deny users from receiving rewards.
Furthermore, potential loss of precision can occur when recalculating the `rate` in `replenish()` function. If the `spread` is set to a large value, the reward `rate` can be zero.

Recommendation: Restrict access to `ColanderRewards.replenish()` function using the `auth` modifier, and restrict the size of `spread` when setting the state variable in `ColanderRewards.setSpread()`.

Update: The contract `colander.sol` has been archived and will not be used.

QSP-2 Loss of Strategy Funds During Withdrawal

Severity: *High Risk*

Status: Mitigated

File(s) affected: `MasterVault.sol`, `BaseStrategy.sol`, `IBaseStrategy.sol`

Description: The `MasterVault._withdrawFromStrategy()` internal function allows the withdrawal of funds from strategy through `IBaseStrategy(strategy).withdraw()` function. The withdraw function returns the number of assets withdrawn from the vault as `value`. If the withdrawn `value` exceeds zero, the `MasterVault.totalDebt` and `strategyParams[strategy].debt` are updated based on the `amount` provided externally. This greater-than-zero validation is insufficient. The `MasterVault` needs to confirm the `amount` received before proceeding with debt updates.
The current validation is insufficient for `CerosYieldConverterStrategy.withdraw()` function because it could return fewer than the amount requested. In

`CerosYieldConverterStrategy.withdraw()` the `amount` is set to the `withdrawAmount` if it is lower than the `amount` requested in `CerosYieldConverterStrategy.sol#101`. Furthermore, if more strategies are added in the future, loss of funds is more likely to occur if `withdraw()` can return an amount lower than requested. This problem can compound as more strategies are onboarded and result in protocol insolvency.

Recommendation: Update `MasterVault.totalDebt` and `stategyParams[strategy].debt` based on the `value` returned by `IBaseStategy(strategy).withdraw()`. We also recommend verifying the `MasterVault` final balance to confirm the difference matches the value returned by the `withdraw` function.

Update: The current verification is dependent on the correctness of swapPool.stakeFee(). If the fee is set incorrectly, then the vault risks losing funds during withdrawal. Furthermore, there is no restriction on the pool used by the strategy.

QSP-3 Stale Collateral Price Feed Data

Severity: *Medium Risk*

Status: Mitigated

File(s) affected: `MaticOracle.sol`

Description: The `MaticOracle.peek()` function obtains the underlying collateral price data from ChainLink `latestRoundData()` function. Although the function provides price and timestamp data, only price data is used. The price of the underlying collateral is used without validating the timestamp of the price data. Loss of capital for users or the protocol can occur if stale price data is used.
Furthermore, `MaticOracle.peek()` function converts the price to 18 decimals by assuming the price data to be 8 decimals instead of querying for the number of decimals directly from the oracle.

Recommendation: Add a check to compare the current block timestamp and the timestamp at which the price data is obtained. Assuming that Chainlink has high availability of their service in general with a potential outage for a short period, consider reverting the transaction if the price data obtained is determined to be stale. Before converting the price data, obtain the number of decimals used in price data directly from the oracle. In addition, the price data should be greater than zero. Otherwise, the price data should be considered invalid.

Update: Based on the changes in `MaticOraclev2.sol`, the issue is mostly fixed. However, `MaticOracle.sol` remains in the repository. There are additional issues with the new changes:

1. The code assumes the decimal returned by Chainlink is 8 decimals. This is true for non-ETH pairs. The best practice would be to obtain the decimals directly from the contract.
2. The price returned by the backup Pyth Price Feed should be `x * (10^expo)`. The current usage recalculates the price to 18-decimal. The conversion works if `expo` is negative. However, if `expo` is positive, then this decimal conversion fails because the decimal will be greater than 18. Based on available documentation, unsure if it is possible that `expo` can be greater than zero. However `expo` is type `int64`.
3. For best practice, the price data returned by backup Pyth Price Feeds should check for zero. From Pyth Price Feed best practice, Pyth SDKs guard against price availability failure mode by incorporating a staleness check by default. The developer should confirm the staleness threshold used in Pyth price feed and note that solidity SDK will revert in the event of stale price.

QSP-4 Loss of Stability Fee and Rewards During Payback

Severity: *Medium Risk*

Status: Acknowledged

File(s) affected: `Interaction.sol`

Description: User can pay back `SIKKA` debt with `Interaction.payback()` function. During playback, stability fees and GIKKA rewards are calculated for the user.
When `jug.drip()` is called, stability fees are collected. The stability fee is calculated based on the rate and current debt `Art` between the current time stamp and the last time the drip function was called. When `Interaction.dropRewards()` is called, GIKKA rewards are calculated based on the current debt of the user.
Both `drip()` and `dropRewards()` calculations are dependent on the current `ilk` debt amount. However, `vat.frob()` is called before both functions, with overall `ilk` debt `Art` decreasing before fees and rewards are determined. This result in lower stability fees and rewards when `drip()` and `dropRewards()` are called.

Recommendation: 1. During `Interaction.payback()`, invoke `jug.drip()` and `Interaction.dropRewards()` before proceeding with `vat.frob()`. In general, it is recommended that `drip()` is invoked before `join()` and `exit()` in MakerDAO's documentation.

1. Update the tests in `interaction.test.js` to confirm these types of scenarios.

Update: Marked as "Acknowledged" by the client. The client provided the following explanation: Intended behavior. There is a backend triggering drip() in jug.sol at intervals.

QSP-5 Missing Pause Implementation

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `MasterVault.sol`

Description: The modifier `whenNotPaused` from `PausableUpgradeable` is only enabled when internal variable `_pause` is set to `true` using internal function `PausableUpgradeable._pause()`. The `whenNotPaused` modifier is used in `MasterVault.depositEth()` and `MasterVault.withdrawEth()` functions. However, the `MasterVault` contract is missing `pause()` and `unpause()` functions to enable pause. Therefore, it is not possible to disable withdraw and deposit functionality in the event of emergency.

Recommendation: Add `pause()` and `unpause()` function to enable and disable pause functionalities from `PausableUpgradeable.sol`. Add tests to confirm pause is implemented correctly.

Update: Marked as "Fixed" by the client. Addressed in: `b795dd0`. The client provided the following explanation: Added pause (L460) and (L464) unpause.

QSP-6 Sikka Total Supply Limitation May Impair Liquidation Process

Severity: *Medium Risk*

Status: Acknowledged

File(s) affected: `sikka.sol`, `Interaction.sol`, `AuctionProxy.sol`

Description: When minting `SIKKA` through `sikka.mint()`, the `totalSupply` must be equal to or less than the current `supplyCap`. If the total supply limitation is reached, more `SIKKA` cannot be minted. This supply restriction is unique to SIKKA's fork and does not exist in MakerDao's `DAI` implementation.
When a `SIKKA` borrower's current worth of collateral falls below the liquidation ratio, the liquidation process can be triggered through the `Interaction` contract. The liquidation process involves a dutch auction and allows anyone to start, buy, and reset the dutch auction using `Interaction.startAuction()`, `Interaction.buyFromAuction()`, and

`Interaction.resetAuction()` functions. If `SIKKA` reaches its supply capacity, only existing `SIKKA` borrowers may participate in the liquidation process. This limits participants and can impair the liquidation process and price stability mechanism.

Recommendation: The clients should be aware of this issue and monitor the `SIKKA` total supply and supply limitation closely. The `SIKKA` supply limitation should not impair the price stability mechanism. The client should communicate such risks within the user documentation.

Update: Marked as "Acknowledged" by the client. The client provided the following explanation: Yes, we are aware of it and makerDAO also has this limit internally in vat contract at (L54, L70). The supply is actively monitored.

QSP-7 Signature Malleability

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `sikka.sol`

Description: The given implementation of signature verification using `ecrecover` directly is prone to signature malleability.

Recommendation: Consider using a secure wrapper like OpenZeppelin's ECDSA utility library, which performs additional security checks on the signature parameters.

Update: Marked as "Fixed" by the client. Addressed in: `b795dd0`. The client provided the following explanation: Using OpenZeppelin's implementation now in contracts/sikka.sol at L144.

QSP-8 Pool Limit Can Be Bypassed

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `IkkaRewards.sol`

Description: In `IkkaRewards.claim()` there is a violation of the Checks-Effects-Interaction pattern, making the function vulnerable to re-entrancy attacks. An attacker could claim rewards bypassing the pool limit.

Recommendation: Move the external call (`IERC20Upgradeable(ikkaToken).safeTransfer(msg.sender, amount)`) after the update of `poolLimit`.

Update: Marked as "Fixed" by the client. Addressed in: `b795dd0`. The client provided the following explanation: Added pool limit and moved trasnfer to L184 in contracts/IkkaRewards.sol.

QSP-9 Spot Price May Be Stale

Severity: *Medium Risk*

Status: Acknowledged

File(s) affected: `spot.sol`, `Interaction.sol`

Description: In MakerDao, the spot price is updated independently of `vat.frob(...)`. The spot price needs to be poke frequently; otherwise, lack of poke can lead to stale spot prices. Updating the spot price is critical because this is the only way `vat` can obtain price information about ink collateral. This is one of the failure modes noted in Makerdao's spot documentation. The `vat.frob()` function is used in`Interaction.deposit(...)`, `Interaction.borrow(...)`, `Interaction.payback(...)` and `Interaction.withdraw(...)`.

Recommendation: Add documentation regarding how frequent the spot price will be updated especially in event of significant price movement.

Update: Marked as "Acknowledged" by the client. The client provided the following explanation: , we are aware of the fact that the internal spot price must be poked. And we will do every 3-4 hours.

QSP-10 Unknown Oracle for Synthetic Assets

Severity: *Medium Risk*

Status: Acknowledged

File(s) affected: `vat.sol`, `spot.sol`, `Interaction.sol`

Description: In Sikka the MakerDao's vat holds and collateralizes `ceMATIC` which is a synthetic aMATICc. aMATIC is a reward-bearing token and appreciates in value in relation to `MATIC`. Users can deposit `MATIC` in masterVault through `SikkaProvider.provide(...)`. The masterVault will convert `MATIC` into `aMATIC` which gets stored in `ceVault` and the `ceVault` mints ceMATIC and the ceMATIC become collateralized in the vat. Since the asset stored in the `vat` is synthetic, it is unclear which oracle will be used in the spot to provide price information in the `vat`.

Recommendation: Add documentation regarding the oracle used in the `spot` contract, which is used by the `vat` to obtain price information about ink collateral.

Update: Marked as "Acknowledged" by the client. The client provided the following explanation: the price from MaticOracle is being used for ceMatic.

QSP-11 Privileged Owner Can Prevent User From Collateral Withdrawal

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `SikkaProvider.sol`

Description: The `SikkaProvider.daoBurn()` function allows the proxy contract to burn collateral tokens of `account` during the liquidation process. However, the `daoBurn()` function may be abused by the SikkaProvider's `owner` to prevent users from withdrawing collateral. The owner of `SikkaProvider` can change the proxy address to any address through `SikkaProvider.changeProxy()` and invoke `daoBurn()` to burn collateral token for any `account`. When users lose their collateral tokens, the users will be unable to withdraw their `MATIC` collateral in `release()`. Similarly, more `sMATIC` tokens may be created through `SikkaProvider.daoMint()` without locking collateral and allow more `sMATIC` than the `MATIC` collateral stored in the `MasterVault`.

Exploit Scenario: 1. Alice deposits 100 `MATIC` using `SikkaProvider.deposit()` and receives 100 `sMATIC` in return.

1. The owner of `SikkaProvider` updates the proxy address to their address through `SikkaProvider.changeProxy()`.
2. The owner invokes `SikkaProvider.daoBurn()` and burns Alice's collateral token `sMATIC`.
3. Alice tries to withdraw her collateral `MATIC` through `SikkaProvider.release()` and the function fails since she does not have any collateral token.

Recommendation: The client should document the risk to the users of the protocol and consider protecting critical operations behind multi-sig.

Update: Marked as "Acknowledged" by the client. The client provided the following explanation: Yes, we are aware and we're going to use multi-sig.

QSP-12 Privileged Owner Can Withdraw User `aMATICc` From `CeVault`

Severity: Low Risk

Status: Acknowledged

File(s) affected: `CeVault.sol`

Description: The `CeVault.withdrawFor()` function allows the router contract to withdraw `aMATICc` from any `owner` to `recipient` destination. This function is only accessible by the `router` address. However, the owner of `CeVault` can change the router address to any address through `CeVault.changeRouter()` and invoke `withdrawFor()` to withdraw collateral token from any `owner`.

Recommendation: The client should document the risk to the users of the protocol and consider protecting critical operations behind multi-sig.

Update: Marked as "Acknowledged" by the client. The client provided the following explanation: Yes, we are aware and we're going to use multi-sig.

QSP-13 Dilute Existing Lp Holders with Ratio Modification

Severity: Low Risk

Status: Acknowledged

File(s) affected: `SwapPool.sol`

Description: When removing liquidity from the `SwapPool`, users can withdraw native tokens and ceros tokens based on the amount of LP tokens held relative to the total LP supply. The amount of LP tokens minted is based on the number of native tokens and the ratio of native tokens to ceros tokens within the protocol. This ratio is dynamic and can change over time. If the ratio of native tokens to ceros tokens exceeds 1, the newly minted LP tokens start to dilute the existing LP holders and reduce the native funds the liquidity provider can withdraw in the future.
The LP dilution is problematic if the Ceros token ratio drops to a low value or becomes worthless.

Exploit Scenario:

1. Alice adds 30 wMATIC and 30 Ceros tokens (valued at 60 wMATIC) in `SwapPool` using `addLiquidity()` when the Ceros Token ratio is 1 and receives 600000000000 LP tokens (100% of the total LP supply).
2. After some time passes, the Ceros token ratio spikes to 2.
3. Bob adds 15 wMATIC and 30 Ceros tokens (valued at 75 wMATIC) in `SwapPool` using `addLiquidity()` when the Ceros Token ratio is 2 and receives 400000000000 (40% of the total LP supply).
4. After some time passes, the Ceros token ratio drops to 0.5.
5. Bob withdraws his wMATIC and Ceros tokens using `removeLiquidity()` and receives 18 wMATIC and 24 Ceros tokens. Bob obtains more native tokens than he originally deposited.
6. Alice withdraws her wMATIC and Ceros tokens using `removeLiquidity()` and receives 27 wMATIC and 36 Ceros tokens. Alice obtains fewer native tokens than she originally deposited.

Recommendation: Based on available documentation, it is unclear if the Ceros token ratio can be greater than one. If Ceros token ratio can exceed 1, the client should update the documentation to inform the users can receive fewer native tokens than expected when withdrawing liquidity from the swap pool.

Update: Marked as "Acknowledged" by the client. The client provided the following explanation: The ratio never increases but rather it decreases.

QSP-14 Missing Uncage Function

Severity: Low Risk

Status: Fixed

File(s) affected: `dog.sol`, `jar.sol`, `jug.sol`, `join.sol`, `spot.sol`, `vat.sol`, `vow.sol`

Description: The `Dog`, `Jar`, `Jug`, `GemJoin`, `SikkaJoin`, `Spotter`, `Vat`, and `Vow` contracts are missing `uncage` function implementation. Once a contract is locked from `cage`, it is not possible to `uncage` the locked contract.

Recommendation: Add the missing `uncage` implementations in the listed contracts.

Update: Marked as "Fixed" by the client. Addressed in: `b795dd0`. The client provided the following explanation: Added uncage functions in the relevant contracts.

QSP-15 Old Strategy Still Exist After Migration

Severity: Low Risk

Status: Fixed

File(s) affected: `MasterVault.sol`

Description: An existing strategy can be migrated to a new strategy through `MasterVault.migrateStrategy()`. The function withdraws all debt from the old strategy and updates `strategies` with the `newStrategy` address and parameters in `strategiesParams`. However, the old strategy parameters still exist in `strategiesParams` and should be cleared. Furthermore, the funds withdrawn from the old strategy are not deposited into the new strategy. From the documentation available, it is unclear if this is an intentional design.

Recommendation: Update the code documentation to specify the intention of the function and delete old strategy parameters in `strategyParams` after successful migration using `_deactivateStrategy()`.

Update: Marked as "Fixed" by the client. Addressed in: `b795dd0`. The client provided the following explanation: Added deactivation of old strategy in contracts/MasterVault/MasterVault.sol in L418.

QSP-16 Silent Failed ERC20 Transfer

Severity: *Low Risk*

Status: Mitigated

File(s) affected: `CerosYieldConverterStrategy.sol`, `MasterVault.sol`, `SwapPool.sol`

Description: The `CerosYieldConverterStrategy.harvestAndSwap()` and `CerosYieldConverterStrategy._withdraw()` function uses `transfer()` without checking the return value. An ERC20 token can potentially fail on transfer by returning false instead of reverting directly. In that case, the functions will be considered successful despite failed token transfer.
In general, the return value should be verified after transferring ERC20 tokens, the following functions contain transfer operations without return value validation:

1. `MasterVault._depositToStrategy()`
2. `SwapPool._addLiquidity()`
3. `SwapPool._removeLiquidity()`
4. `SwapPool._swap()`
5. `SwapPool._withdrawOwnerFee()`

Recommendation: Use the `safeTransfer(...)` or `safeTransferFrom()` functions from the SafeERC20 contract of the OpenZeppelin (see: doc) or verify the return value to confirm a successful transfer.

Update: Marked as "Fixed" by the client. Addressed in: `b795dd0`. The client provided the following explanation: Using safe now in relevant contracts.

QSP-17 Missing Input Validation

Severity: *Low Risk*

Status: Fixed

File(s) affected: `colander.sol`, `Interaction.sol`, `MasterVault.sol`, `SwapPool.sol`

Description: It is crucial to validate inputs, even if the inputs come from trusted addresses, to avoid human error. A lack of robust input validation can only increase the likelihood and impact in the event of mistakes.
Following is the list of places that can potentially benefit from a stricter input validation:

1. colander.sol#95: the range of `_ray` of the `setProfitRange()` function should be verified before setting the `profitRange`.
2. colander.sol#100: the range of `_wad` of the `setPriceImpact()` function should be verified before setting the `priceImpact`.
3. colander.sol#100: the value of `_flashDelay` of the `setFlashDelay()` function should be greater than zero delay before setting the `flashDelay`.
4. colander.sol#358: the `_wad` of the `replenish()` function should be greater than zero. Otherwise, it is possible to extend the distribution of the current
5. Interaction.sol#126: the `ilk` of the `setCollateralType()` function should not be empty.
6. Interaction.sol#221: the `sikkaAmount` of the `borrow()` function should be greater than zero. The `dart` only confirms if the change in debt is equal to or greater than zero.
7. Interaction.sol#250: the `sikkaAmount` of the `payback()` function should be greater than zero.
8. MasterVault.sol#146: the `amount` of the `withdrawETH()` function should be greater than zero.
9. MasterVault.sol#208: the `strategy` of the `_depositToStrategy()` function should be active.
10. MasterVault.sol#258: the `strategy` of the `_withdrawFromStrategy()` function should be active.
11. SwapPool.sol#273: the `receiver` of the `_swap()` function should not be a zero address.
12. SwapPool.sol#625: the `success` when removing a manager from `managers_` should be verified before deleting `managerRewardDebt`.

Recommendation: Add the validations and checks listed in the description.

Update: Marked as "Fixed" by the client. Addressed in: `b795dd0`. The client provided the following explanation: Input validation being checked now in relevant places.

QSP-18 Revocable Ownership

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `SikkaProvider.sol`, `CerosRouter.sol`, `CeVault.sol`, `CeToken.sol`, `sMATIC.sol`, `MasterVault.sol`, `SwapPool.sol`, `SlidingWindowOracle.sol`, `IkkaOracle.sol`, `PriceOracle.sol`, `PriceOracleTestnet.sol`, `BaseStrategy.sol`

Description: OpenZeppelin's OwnableUpgradeable contract contains `renounceOwnership()` function which allows the owner to renounce ownership. If the owner calls `renounceOwnership()`, the contract would be left without an owner, making some aspects of the contract impossible to use.

Recommendation: Consider if ownership revocation is a necessary feature. If it is not, override the ownership revocation functionality to disable it. If it is, add end-user documentation stating the risk.

Update: Marked as "Acknowledged" by the client. The client provided the following explanation: We are aware of this and in case we revoke the ownership in future, we'll still be able to upgrade the contracts and change the owner.

QSP-19 Inaccurate Accounting if Deflationary Tokens Are Accepted

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `colander.sol`, `join.sol`, `vat.sol`, `SwapPool.sol`

Description: The `colander.join()` function allows the user to stake SIKKA based on the `_wad` deposited. If other deflationary tokens are accepted in the future, internal accounting will be inaccurate for tokens that subtract fees during transfer. The value stored in the protocol may be different from the value tracked using the `_wad` in the `colander.join()` function. Inaccurate accounting of deflationary tokens can lead to insolvency during withdrawal in `colander.exit()`.

Recommendation: If the team wishes to handle tokens with fees on transfer correctly, only account for the difference between the pre-transfer balance and the post-transfer balance of the pool. The difference will capture the number of tokens that have been transferred, regardless of the token transfer mechanism.

Update: Marked as "Acknowledged" by the client. The client provided the following explanation: No plans to introduce deflationary tokens.

QSP-20 Incorrect Value Emitted During Events

Severity: Low Risk

Status: Fixed

File(s) affected: `SikkaProvider.sol`, `MasterVault.sol`

Description: Incorrect values may be emitted in the events listed below:

1. `SikkaProvider.release()`: When a user withdraws MATIC through `SikkaProvider`, the `Withdrawal` event incorrectly emits `amount` instead of `realAmount`.
2. `MasterVault._depositToStrategy()`: `DepositedToStrategy` emits the `amount` instead of the `value` returned by `IBaseStrategy`.

Recommendation:

1. Emit `realAmount` instead of the `amount` in the `Withdrawal` event.
2. Emit `value` instead of the `amount` in the `DepositedToStrategy` event.

Update: Marked as "Fixed" by the client. Addressed in: `b795dd0`. The client provided the following explanation: Added correct event values in relevant contracts.

QSP-21 Potential Denial of Service when Claiming GIKKA Rewards

Severity: Low Risk

Status: Acknowledged

File(s) affected: `IkkaRewards.sol`

Description: When `IkkaRewards.claim()` is called, the `GIKKA` rewards are based on the `SIKKA` borrowed and the current time stamp and the last time the `drop()` function was called. If too much time has elapsed where the `drop()` function was last called, denial of service can occur in `unrealisedRewards()` at `hMath.rpow()` due to the computation of a large value. If denial of service occurs at `unrealisedRewards()`, users will be unable to claim their rewards through `IkkaRewards.claim()`.

Recommendation: The client should be aware of this denial of service risk and ensure that `IkkaRewards.drop()` is invoked regularly.

Update: Marked as "Acknowledged" by the client. The client provided the following explanation: We are aware of this and we'll ensure that ikkaRewards.drop() is triggered regularly.

QSP-22 Oracle Decimal Assumption

Severity: Low Risk

Status: Acknowledged

File(s) affected: `IkkaRewards.sol`

Description: The function `unrealisedRewards` obtains GIKKA price using `IkkaRewards.ikkaPrice()`. The GIKKA price is used to determine the amount of GIKKA tokens for reward distribution. The decimal supplied in the calculation is hardcoded at 18 decimals. This assumes the oracle will always supply IKKA price with 18 decimals. Since it is possible to update the oracle in `IkkaRewards` through `setOracle()` function, the decimals used in the calculation should not be assumed.

Recommendation: Do not assume decimals provided by `ikkaPrice()`. The `ikkaPrice()` should convert the decimals based on the decimals provided by the oracle before returning the result.

Update: Marked as "Acknowledged" by the client. The client provided the following explanation: We are aware of this.

QSP-23 Upgradeable Misconfigurations

Severity: Informational

Status: Acknowledged

File(s) affected: `All implementation contracts`

Description: The implementation contracts are not guaranteed to have their initializer disabled. This makes it possible for a malicious actor to call the `initialize()` function on the implementation contracts.
Additionally, the `initializer` modifier can only be called once even when using inheritance, so parent/base contracts should use the `onlyInitializing` modifier during initialization functions.
The following functions should have their `initializer` modifier changed to `onlyInitializing`:

1. `BaseStrategy.__BaseStrategy_init()`

Recommendation: Add _disableInitializers() to all constructors of upgradeable contracts" is a best practice even if the contract cannot self destruct. It will avoid the initialization of the implementation contract from other users. An initialization with wrong parameters can emit wrong events that could lead to confusion.

Update: Marked as "Mitigated" by the client. Addressed in: `b795dd0`. The client provided the following explanation: onlyInitializing modifier added to contracts/Strategy/BaseStreategy.sol at L34.

QSP-24 Borrow Sikka Before Claim to Maximize Gikka Rewards

Severity: Informational

Status: Acknowledged

File(s) affected: `IkkaRewards.sol`

Description: In the SIKKA protocol, users can earn rewards based on the amount of SIKKA borrowed. To redeem the rewards, the user must initiate `IkkaRewards.claim()` function. The `claim()` function calculates GIKKA rewards based on the current amount of borrowed SIKKA and the time difference between the current time and the time when the reward was last collected

in `IkkaRewards.drop()` function. The reward methodology incorrectly assumes the current borrowed SIKKA amount remains the same between the current time stamp and the time when rewards were calculated in the `drop()` function. A user can borrow a lot of SIKKA from the `vat` before calling the `claim()` function to maximize GIKKA rewards. Based on the issue described, it is possible for a user to take all GIKKA rewards from the contract.
Furthermore, the GIKKA reward system is limited based on the `poolLimit`. Once the limit is exhausted, users are unable to redeem GIKKA rewards through `claim()` until the reward limit is updated in `IkkaRewards.setRewardMaxLimit()`.
The severity of this issue is downgraded to informational since it is not possible to update the SIKKA balance without going through `Interaction` contract.

Exploit Scenario: 1. Alice calls `IkkaRewards.drop()` for collateral `A` and `pile.ts` is updated with the latest timestamp (t=0).

1. Some time passes (t=10), and Alice is ready to collect her GIKKA rewards.
2. In one transaction, Alice deposits collateral `A`, locks her collateral `A` in the vat, borrows SIKKA against her locked collateral, collects her GIKKA rewards, and returns the borrowed SIKKA.

Recommendation: The reward balance is updated whenever the SIKKA balance changes in the vat for an `usr` when borrowing SIKKA through `Interaction.borrow()`; continue to maintain access restriction to the `vat.frob()`. In Makerdao's fork, this function is accessible to anyone. The client should proceed with caution when granting access to `vat.frob()` function.

Update: Marked as "Acknowledged" by the client. The client provided the following explanation: vat.frob() will remain restricted to interaction contract only. Therefore, users will not be able to deposit and borrow directly from vat to maximise their rewards.

QSP-25 Loss of Gikka Rewards

Severity: *Informational*

Status: Mitigated

File(s) affected: `IkkaRewards.sol`

Description: In the SIKKA protocol, users can earn rewards based on the amount of SIKKA borrowed. To redeem the rewards, the user must initiate `IkkaRewards.claim()` function. The `claim()` function updates the rewards from all pools and transfers pending rewards to the user.
The `claim()` function calls `IkkaRewards.drop()` to update all rewards from the pools and the last time stamp in the mapping `pile`. In `drop()` the rewards are summed and tracked in `pile.amount` from `IkkaRewards.unrealisedRewards()` and the timestamp is updated in `pile.ts`. If the last recorded timestamp `pile.ts` is zero, `unrealisedRewards()` returns zero for reward. If the `drop()` function is never called for `usr` the user can lose rewards when the user calls `claim()` after. The severity of this issue is downgraded to informational since it is not possible to update the SIKKA balance without going through `Interaction` contract.

Recommendation: The reward balance is updated whenever the SIKKA balance changes in the vat for an `usr`; continue to maintain access restriction to the `vat.frob()` function. In Makerdao's fork, this function is accessible to anyone. The client should proceed with caution when granting access to `vat.frob()` function.

Update: Marked as "Acknowledged" by the client. The client provided the following explanation: vat.frob() will remain restricted to interaction contract only. Therefore, users will not be able to deposit and borrow directly from vat to maximise their rewards.

QSP-26 Missing Events to Signal State Changes

Severity: *Informational*

Status: Fixed

File(s) affected: `spot.sol`, `jug.sol`, `Interaction.sol`, `SwapPool.sol`

Description: Events are important for signalling state changes in a contract. This helps debug the contract in the event of any attacks or critical bugs and helps indicate to users any configuration changes in the contract.
A non-exhaustive list of functions where events can be useful includes:

1. `spot.file(bytes32 ilk, bytes32 what, address pip_)`
2. `spot.file(bytes32 what, uint data)`
3. `spot.file(bytes32 ilk, bytes32 what, uint data)`
4. `jug.file(bytes32 ilk, bytes32 what, uint data)`
5. `jug.file(bytes32 what, uint data)`
6. `jug.file(bytes32 what, address data)`
7. `vat.file(bytes32 what, uint data)`
8. `vat.file(bytes32 ilk, bytes32 what, uint data)`
9. `vow.file(bytes32 what, uint data)`
10. `vow.file(bytes32 what, address data)`
11. `Interaction.setWhitelistOperator(address usr)`
12. `Interaction.setRewards(address rewards)`
13. `Interaction.setCores(address vat_, address spot_, address sikkaJoin_, address jug_)`
14. `Interaction.setSikkaProvider(address token, address sikkaProvider)`
15. `Interaction.setReward(address rewards)`
16. `SwapPool.setThreshold(uint24 newThreshold)`
17. `SwapPool.setMaticPool(address newMaticPool)`

Recommendation: Add events to signal the contract's state changes.

Update: Marked as "Fixed" by the client. Addressed in: `b795dd0`. The client provided the following explanation: Added events in relevant contracts mentioned.

QSP-27 Upgradeable Contract Storage Gaps

Severity: *Informational*

Status: Fixed

File(s) affected: `BaseStrategy.sol`

Description: For future update considerations, add storage gaps in upgradeable base contracts to avoid storage collisions when introducing new variables. This pattern is used in OpenZeppelin's upgradeable contracts.

Recommendation: Add a storage gap at the end of the upgradeable abstract contract.

Update: Marked as "Fixed" by the client. Addressed in: `b795dd0`. The client provided the following explanation: Added gaps in relevant contracts.

QSP-28 Potential Denial of Service Due to Block Gas Limit

Severity: Informational

Status: Acknowledged

File(s) affected: `IkkaRewards.sol`

Description: * Iterating over a list in storage requires a `SLOAD` (`2100` gas when it is accessed for the first time, and every subsequent access `100` gas) for each element in the list. By default, dynamic lists have no maximum size and so the maximum cost of iterating through them is unbounded. Therefore, the cost of said iteration may exceed Ethereum's block gas limit, which would result in a transaction reversion. This would render all code executed after said iteration unexecutable.
Storage array iteration is performed in the following places:

1. In IkkaRewards.sol#183 for `poolsList` of `claim()` function.
2. In WaitingPool.sol#53 for `people` of `tryRemove()` function.

Recommendation:

1. Iteration 1 can be addressed by ensuring a cap on the size of the arrays in `poolsList`. When adding an address in the `poolsList` in `IkkaRewards.initPool()`, the function should include a length validation.
2. Iteration 2 can be addressed by ensuring a cap on the size of the arrays in `people`. When adding a person in the `people` in `WaitingPool.addToQueue()`, the function should include a length validation.

Update: Marked as "Acknowledged" by the client. The client provided the following explanation: We have a cap limit in tryRemove() of waiting pool to avoid exceeding block gas limit.

QSP-29 Potential Denial of Service From Stability Fee Collection

Severity: Informational

Status: Acknowledged

File(s) affected: `jug.sol`, `Interaction.sol`

Description: When `jug.drip()` is called, the `SIKKA` balance of vow is increased to account for the stability fees collected from opened CDP positions. The stability fee is calculated from base and duty between the current time stamp and the last time the drip function was called. If too much time has elapsed where the drip function was last called, denial of service can occur in `jug.drip()` at `sMath.rpow()` due to the computation of a large value. If denial of service occurs at `jug.drip()`, the following functions will be impacted and become unusable:

1. `Interaction.deposit()`
2. `Interaction.borrow()`
3. `Interaction.payback()`
4. `Interaction.setCollateralDuty()`
5. `Interaction.drip()`

Recommendation: The client should be aware of this denial of service risk and ensure that `jug.drip()` is invoked regularly.

Update: Marked as "Acknowledged" by the client. The client provided the following explanation: We are aware of this. Also, jug.drip() is triggered in deposit() borrow() repay().

QSP-30 Ignored Returned Value From Functions

Severity: Informational

Status: Fixed

File(s) affected: `SikkaProvider.sol`

Description: For best practice, the returned value should be used and verified.
A non-exhaustive list of functions ignores returned values from function calls:

1. Users can deposit `MATIC` in `masterVault` through `SikkaProvider.provide()`. The `masterVault` mints `ceMATIC` to the `SikkaProvider` and the `ceMATIC` become collateralized in the `vat`. After collateralization, `sMATIC`'s are minted to the users based on the amount collateralized in the vat. For best practice, the minted amount should be obtained from the return value of the `_dao.deposit()` function.
2. Users can withdraw `MATIC` from `masterVault` through `SikkaProvider.release()`. `sMATIC`'s are burnt based on the amount removed from the vat. For best practice, the burnt amount should be obtained from the return value of the `_dao.withdraw()` function.
3. MasterVault strategy with minimal or no debt can be activated through the `_deactivateStrategy()` function. When retiring a strategy, the return value of `_deactivateStrategy()` is not used in `retireStrat()`. For best practice, the operation status should be checked.

Recommendation:

1. The amount of minted `sMATIC` should be from the returned value from `_dao.deposit()` function.
2. Confirm the strategy is successfully disabled after withdrawal by checking the returned value of `MasterVault._deactivateStrategy()`.

Update: Marked as "Fixed" by the client. Addressed in: `b795dd0`. The client provided the following explanation: Return values are considered.

QSP-31 Risk of Killing Upgrades

Severity: Informational

Status: Acknowledged

Description: The UUPS pattern was chosen for the upgradeable contracts. One of the drawbacks of using such a pattern is that if a future implementation does not implement the function, then upgrades for the contracts have effectively been killed.

Recommendation: Understand the drawbacks of using a UUPS pattern and document the potential risks for users.

Update: Marked as "Acknowledged" by the client. The client provided the following explanation: UUPS is being used in unused contracts only, we'll remove these contracts later.

QSP-32 Usage `transfer()` and `send()` Should Be Avoided

Severity: Informational

Status: Acknowledged

File(s) affected: `MasterVault.sol`, `WaitingPool.sol`

Related Issue(s): SWC-134

Description: The functions `address.transfer()` and `address.send()` assume a fixed amount of gas to execute the call. The use of these functions protects against re-entrancy attacks. The default amount of gas may change in the future. In the worst case, it could lead to failed transfers.

Recommendation: Immediate changes are not required, but the client should be aware of the potential consequences. If more flexibility regarding the forwarded gas is needed, `address.call{value: amount}("")` may be used to perform transfers.

Update: Marked as "Acknowledged" by the client. The client provided the following explanation: We are avoiding passing extra gas to the receiver's receive(). However, thanks for pointing it out, we'll discuss this internally and address later if required.

QSP-33 Clone-and-Own

Severity: Informational

Status: Acknowledged

File(s) affected: `uniswapv3PriceGetter/libraries/*`, `oracle/libraries/*`

Description: The clone-and-own approach involves copying and adjusting open-source code at one's own discretion. From the development perspective, it is initially beneficial as it reduces the amount of effort. However, from the security perspective, it involves some risks as the code may not follow the best practices, may contain a security vulnerability or may include intentionally or unintentionally modified upstream libraries.

Recommendation: Rather than the clone-and-own approach, good industry practice is to use a package manager (e.g., npm) for handling library dependencies. This eliminates the clone-and-own risks yet allows for following best practices, such as using libraries. If the file is cloned anyway, a comment including the repository, the commit hash of the version cloned, and the summary of modifications (if any) should be added. This helps to improve the traceability of the file.

1. The contracts within `uniswapv3PriceGetter/libraries/\*` can be replaced by importing the libraries from `@uniswap/v3-core`.
2. The contracts within `oracle/libraries/\*` can be replaced by importing the libraries from `@uniswap/lib/contracts/libraries`, `@uniswap/v3-periphery`, and `@uniswap/v2-periphery`.
3. Remove the library `hMath` since the library is not used.

Update: Marked as "Acknowledged" by the client. The client provided the following explanation: We are aware of this. The contracts we are using are adapted from Uniswap's official git but not a part of their contracts library.

QSP-34 Unlocked Pragma

Severity: Informational

Status: Acknowledged

File(s) affected: `All files`

Related Issue(s): SWC-103

Description: Every Solidity file specifies in the header a version number of the format `pragma solidity (^)0.8.*`. 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 behaviour in the future, we recommend removing the caret to lock the file onto a specific Solidity version.

Update: Marked as "Acknowledged" by the client. The client provided the following explanation: We are aware of this.

QSP-35 Auction Maximum Sikka Amount Is Always Greater by 1

Severity: Undetermined

Status: Fixed

File(s) affected: `AuctionProxy.sol`

Description: The `AuctionProxy.buyFromAuction()` function calculates the maximum `SIKKA` amount based on the maximum price accepted and the collateral amount to be purchased. Based on available documentation and test, it is unclear why the maximum `SIKKA` amount is always greater by 1. This will be confusing for a user attempting to buy from the auction.

Recommendation: Update the code documentation to explain why the `AuctionProxy.buyFromAuction()` function increases the transferred `SIKKA` by 1. It is unclear why the `SIKKA` amount needs to be increased by 1.

Update: Marked as "Fixed" by the client. Addressed in: `b795dd0`. The client provided the following explanation: Removed +1 from buyFromAuction() in AuctionProxy.sol.

QSP-36 Borrowed Amount Is Always Greater by 100

Severity: *Undetermined*

Status: Acknowledged

File(s) affected: `Interaction.sol`

Description: The `Interaction.borrowed()` function returns a user's total borrowed `SIKKA` for a specific collateral `token`. It is unclear why the borrowed amount is always greater than the actual borrowed `SIKKA`. Based on the available documentation and tests, it is unknown why the borrowed amount is increased by 100.

Recommendation: Update the code documentation to explain why the `Interaction.borrowed()` function returns an inaccurate borrowed `SIKKA` amount. It is unclear why the `SIKKA`amount needs to be increased by 100.

Update: Marked as "Acknowledged" by the client. The client provided the following explanation: Intended - This is just to make sure that users can close their CDPs as frontend needs to get the approval of (borrowed amount + stability fee) as we noticed sometimes approval amount(fetched from borrowed()) was lesser than the actual borrowed amount that is required to close the CDP.

Adherence to Specification

1. The token name is outdated and should be updated to `GIKKA` to match the documentation. Please update the token name and variable names in the codebase. There are also references to `IKKA` in the documentation, which should be updated too.
2. The usage of `ETH` should be renamed to `MATIC` in the `MasterVault`.
3. There are some typographical errors in the codebase, e.g. "certToekn", "yeild". We recommend using a spell checker to fix them.
4. Comment for `WaitingPool.initialize()` looks wrong.

Code Documentation

1. Update the Docstrings following the Ethereum Natural Specification Format (NatSpec). There are many instances where the codebase is not adhering to the NatSpec.

Adherence to Best Practices

1. The visibility of `initialize` functions can be changed from `public` to `external`.
2. The pause features of `PausableUpgrade` are unused; consider removing `PausableUpgradeable` from the following contracts' inheritance:
   1. `CeVault.sol`
   2. `CerosRouter.sol`
   3. `BaseStrategy.sol`
   4. `SwapPool.sol`

3. Consider removing unused imports in the following locations:
   1. `ICerosRouter.sol` in `MasterVault.sol`
   2. `ISikkaProvider.sol` in `MasterVault.sol`
   3. `ICertToken.sol` in `MasterVault.sol`
   4. `IDao.sol` in `MasterVault.sol`
   5. `IPriceGetter.sol` in `MasterVault.sol`
   6. `IVault.sol` in `SikkaProvider.sol`
   7. `IDex.sol` in `SikkaProvider.sol`
   8. `ICerosRouter.sol` in `SikkaProvider.sol`

4. The following functions shadow the `OwnableUpgradeable.owner` variable with its local variable `owner`; for best practice, the local variable should be renamed, otherwise `onlyOwner` implementation would be correct if applied to the functions:
   1. `CeVault.claimYieldsFor()`
   2. `CeVault._claimYields()`
   3. `CeVault._withdrawFor()`
   4. `CeVault._withdraw()`

5. In `MasterVault.allocate()` function, the local variable `totalAssets` shadows `ERC4626Upgradeable.totalAssets`. Rename the local variable to avoid shadowing state variables.
6. In `CerosYieldConverterStrategy.initialize()` function, the local variable `destination` and `feeRecipient` shadow `BaseStrategy.destination` and `BaseStrategy.feeRecipient`. Rename the local variable to avoid shadowing state variables.
7. The following functions can be renamed to accurately convey their functionality:
   1. `MasterVault.withdrawETH()` can be renamed to `MasterVault.withdrawMATIC()`.
   2. `MasterVault.depositETH()` can be renamed to `MasterVault.depositMATIC()`.
   3. `MasterVault.setStrategy()` can be renamed to `MasterVault.addStrategy()`.
   4. `MasterVault._deactivateStrategy()` can be renamed to `MasterVault._deactivateDustStrategy()`.
   5. `SwapPool.addLiquidityEth()` can be renamed to `SwapPool.addLiquidityMATIC()`.
   6. `SwapPool.removeLiquidityEth()` can be renamed to `SwapPool.removeLiquidityMATIC()`.

    7. `SwapPool.removeLiquidityPercentEth()` can be renamed to `SwapPool.removeLiquidityPercentMATIC()`.
    8. `SwapPool.swapEth()` can be renamed to `SwapPool.swapMATIC()`.

8. The following modifiers can be renamed to more accurately convey their functionality:
    1. `MasterVault.onlyProvider` can be renamed to `onlyOwnerOrProvider()`.
    2. `SikkaProvider.onlyProxy()` can be renamed to `onlyOwnerOrProxy()`.
    3. `CeToken.onlyMinter()` can be renamed to `onlyVault()`.

9. Consider replacing the usage of magic numbers with constants variable.
10. Remove dead code from the codebase in the following locations:
    1. `MasterVault.sol#346-#350`
    2. `MasterVaut.sol#344`
    3. `colander.sol#176`
    4. `colander.sol#181`
    5. `colander.sol#187-188`
    6. `colander.sol#194-195`
    7. `colander.sol#202-203`
    8. `colander.sol#232`
    9. `colander.sol#397`
    10. `Interaction.sol#205-219`
    11. `SikkaProvider.sol#24`
    12. `SikkaProvider.sol#50`
    13. `SikkaProvider.sol#54`
    14. `CeVault.sol#150-153`
    15. `CeVault.sol#163-164`

11. For code consistency, use `uint256` instead of `uint` in the codebase.
12. Each Solidity file should only have a single `contract` or `interface` implementation to increase readability. The following Solidity files contain both `interface` and/or multiple `contract` implementations:
    1. `colander.sol` implements both `Colander` and `ColanderRewards`.
    2. `join.sol` implements both `GemJoin` and `SikkaJoin`.
    3. `dog.sol` contains interfaces for `VatLike` and `ClipperLike`.
    4. `clip.sol` contains multiple interfaces.

13. `onlyOwner` modifier can be used to verify ownership in `IkkaOracle.changePriceToken()`.
14. `LP` can inherit from `ERC20BurnableUpgradeable` instead of `ERC20Upgradeable`.
15. Indexed keywords are not used in some events; consider adding indexed keywords to access filterable events. The following events can benefit from indexed variables:
    1. `BaseStrategy.UpdatedStrategist`
    2. `BaseStrategy.UpdatedFeeRecipient`
    3. `CerosYieldConverterStrategy.SwapPoolChanged`
    4. `CerosYieldConverterStrategy.CeRouterChanged`

16. There are inconsistent YEAR representations within the codebase, the client should document the reason for the differences or update `YEAR` to be consistent.
    1. Interaction.sol#26: `YEAR` is represented as `31556952`
    2. IkkaRewards.sol#24: `YEAR` is represented as `31553600`

17. `Interaction.stringToBytes32()` is not used. Consider removing it.
18. Private and internal functions names should start with an underscore according to the [Solidity Style Guide](#).
19. Remove the commented (old) code from the production codebase

Test Results

Test Suite Results

All of 419 tests were passing, however 10 tests were pending.

```
·--------------------------------|-------------|----------------·
|  Contract Name                 ·  Size (KB)  ·  Change (KB)   |
·································|·············|·················
|  Address                       ·      0.086  ·                |
·································|·············|·················
|  AddressUpgradeable            ·      0.086  ·                |
·································|·············|·················
|  aMATICb                       ·      9.401  ·                |
·································|·············|·················
|  aMATICc                       ·      5.074  ·                |
·································|·············|·················
|  AuctionProxy                  ·      7.926  ·                |
·································|·············|·················
|  Babylonian                    ·      0.086  ·                |
·································|·············|·················
|  BitMath                       ·      0.086  ·                |
·································|·············|·················
|  BitMath                       ·      0.086  ·                |
·································|·············|················
|  CerosRouter                   ·      8.623  ·               |
·································|·············|················
|  CerosToken                    ·      3.037  ·               |
·································|·············|················
|  CerosYieldConverterStrategy   ·      7.951  ·               |
·································|·············|················
|  CeToken                       ·      4.696  ·               |
·································|·············|················
|  CeVault                       ·      6.256  ·               |
·································|·············|················
|  Clipper                       ·      9.228  ·               |
·································|·············|················
|  Dog                           ·      4.972  ·               |
·································|·············|················
|  ECDSAUpgradeable              ·      0.086  ·               |
·································|·············|················
|  EnumerableSet                 ·      0.086  ·               |
·································|·············|················
|  EnumerableSetUpgradeable      ·      0.086  ·               |
·································|·············|················
|  ERC20                         ·      2.234  ·               |
·································|·············|················
|  ERC20ModUpgradeable           ·      2.103  ·               |
·································|·············|················
|  ERC20Upgradeable              ·      2.182  ·               |
·································|·············|················
|  ExponentialDecrease           ·      1.654  ·               |
·································|·············|················
|  FixedPoint                    ·      0.173  ·               |
·································|·············|················
|  FixedPoint128                 ·      0.086  ·               |
·································|·············|················
|  FixedPoint96                  ·      0.086  ·               |
·································|·············|················
|  FullMath                      ·      0.086  ·               |
·································|·············|················
|  FullMath                      ·      0.086  ·               |
·································|·············|················
|  GemJoin                       ·      2.743  ·               |
·································|·············|················
|  IkkaOracle                    ·      1.254  ·               |
·································|·············|················
|  IkkaRewards                   ·      7.304  ·               |
·································|·············|················
|  IkkaToken                     ·      5.206  ·               |
·································|·············|················
|  Interaction                   ·     22.404  ·               |
·································|·············|················
|  Jar                           ·      7.811  ·               |
·································|·············|················
|  Jug                           ·      2.924  ·               |
·································|·············|················
|  LinearDecrease                ·      1.516  ·               |
·································|·············|················
|  LiquidityMath                 ·      0.086  ·               |
·································|·············|················
|  LowGasSafeMath                ·      0.086  ·               |
·································|·············|················
|  LP                            ·      4.222  ·               |
·································|·············|················
|  MasterVault                   ·     19.460  ·               |
·································|·············|················
|  MathUpgradeable               ·      0.086  ·               |
·································|·············|················
|  MaticOracle                   ·      0.959  ·               |
·································|·············|················
|  MaticOracleV2                 ·      2.506  ·               |
·································|·············|················
|  MaticPool                     ·      3.590  ·               |
·································|·············|················
|  NonTransferableERC20          ·      0.993  ·               |
·································|·············|················
|  Oracle                        ·      0.847  ·               |
·································|·············|················
|  PriceGetter                   ·      9.845  ·               |
·································|·············|················
|  PriceOracle                   ·      5.155  ·               |
·································|·············|················
|  PriceOracleTestnet            ·      5.155  ·               |
·································|·············|················
|  ProxyLike                     ·      0.906  ·               |
·································|·············|················
|  PythStructs                   ·      0.063  ·               |
·································|·············|················
|  SafeCast                      ·      0.086  ·               |
·································|·············|················
|  SafeERC20Upgradeable          ·      0.086  ·               |
·································|·············|················
|  SafeMath                      ·      0.086  ·               |
·································|·············|················
|  Sikka                         ·      7.075  ·               |
·································|·············|················
|  SikkaJoin                     ·      2.290  ·               |
·································|·············|················
|  SikkaProvider                 ·      5.539  ·               |
·································|·············|················
|  SlidingWindowOracle           ·      7.244  ·               |
·································|·············|················
|  sMath                         ·      0.086  ·               |
·································|·············|················
|  sMATIC                        ·      3.322  ·               |
·································|·············|················
|  Spotter                       ·      2.673  ·               |
·································|·············|················
|  SqrtPriceMath                 ·      0.086  ·               |
·································|·············|················
|  StairstepExponentialDecrease  ·      1.779  ·               |
·································|·············|················
|  StorageSlotUpgradeable        ·      0.086  ·               |
·································|·············|················
|  StringsUpgradeable            ·      0.086  ·               |
·································|·············|················
|  SwapMath                      ·      0.086  ·               |
·································|·············|················
|  SwapPool                      ·     19.641  ·               |
·································|·············|················
|  TickMath                      ·      0.086  ·               |
·································|·············|················
|  Token                         ·      2.953  ·               |
·································|·············|················
|  UniswapV2Library              ·      0.086  ·               |
·································|·············|················
|  UniswapV2OracleLibrary        ·      0.086  ·               |
·································|·············|················
|  UnsafeMath                    ·      0.086  ·               |
·································|·············|················
|  Vat                           ·      7.045  ·               |
·································|·············|················
|  Vow                           ·      4.415  ·               |
·································|·············|················
|  WaitingPool                   ·      2.618  ·               |
·································|·············|················
|  wMATIC                        ·      4.466  ·               |
·································|·············|················
|  WNative                       ·      3.239  ·               |
·--------------------------------|-------------|---------------·


 Auction
   - example
   - auction started as expected
   - auction works as expected
```

```
    - auction works as expected

  Ceros Router (Mumbai Fork)
    Basic functionality
      - staker_1 deposits wMatic and claims profit(swap executed on uniswap-v3)

  Ceros Router
    Basic functionality
------- initial balances and supplies -------
MATIC balance(staker_1): 8999999999880236027473056
balance of staker_1 in aMATICc: 1000000000000000000000000
balance of CeVault in aMATICc: 0
balance of staker_1 in ceToken: 0
supply ceToken: 0
yield for staker_1: 0
current ratio: 1000000000000000000
------- balances and supplies after deposit 1 aMATICc-------
MATIC balance(staker_1): 89999999999168466465369050
balance of staker_1 in aMATICc: 1000000000000000000000000
balance of CeVault in aMATICc: 10000000020000000000
balance of staker_1 in ceToken: 10000000020000000000
supply ceToken: 10000000020000000000
yield for staker_1: 0
current ratio: 1000000000000000000
       ✓ staker_1 deposits wMatic (326ms)
------- balances after ratio has been changed -------
MATIC balance(staker_1): 89999999999168466465369050
balance of staker_1 in aMATICc: 1000000000000000000000000
balance of CeVault in aMATICc: 10000000020000000000
balance of staker_1 in ceToken: 10000000020000000000
supply ceToken: 10000000020000000000
yield for staker_1: 9000000018000000000
current ratio: 100000000000000000
------- balance after yields have been claimed -------
MATIC balance(staker_1): 89999999999016323284036158
balance of staker_1 in aMATICc: 1000000000000000000000000
balance of CeVault in aMATICc: 1000000002000000000
balance of staker_1 in ceToken: 10000000020000000000
supply ceToken: 10000000020000000000
yield for staker_1: 0
current ratio: 100000000000000000
       ✓ claim yields for staker_1 (157ms)
BigNumber.toString does not accept any parameters; base-10 is assumed
------- balance after staker_1 deposited 1 MATIC(Staking Pool) -------
MATIC balance(staker_1): 89999989998819665381935542
balance of staker_1 in aMATICc: 1000000000000000000000000
balance of CeVault in aMATICc: 2000000004000000000
balance of staker_1 in ceToken: 20000000040000000000
supply ceToken: 20000000040000000000
yield for staker_1: 0
current ratio: 100000000000000000
       ✓ staker_1 deposits 1 MATIC(via Staking Pool) (78ms)
------- balance after staker_1 withdrawn MATIC(5000000010000000000) -------
MATIC balance(staker_1): 89999979998445927135179202
balance of staker_1 in aMATICc: 1000000000000000000000000
balance of CeVault in aMATICc: 2500000005000000000
balance of staker_1 in ceToken: 25000000050000000000
supply ceToken: 25000000050000000000
yield for staker_1: 0
current ratio: 100000000000000000
       ✓ staker_1 withdraws MATIC (109ms)
    Updating functionality
       ✓ change Provider
       ✓ change Pool and verify allowances (44ms)
       ✓ change Dex and verify allowances
       ✓ change vault and verify allowancesken
       ✓ change price getter contract address
       ✓ change price getter contract address

  Contract: CeVault
    Basic functionality
       ✓ get vault name
-------balance after deposit 1 MATIC from staker_1-------
current ratio: 1000000000000000000
principal(staker_1): 1000000000000000000
yieldFor(staker_1): 0
principal + yield(staker_1): 1000000000000000000
total in vault(amaticc): 1000000000000000000
max to withdrawal: 1000000000000000000
claimed of staker_1 in vault: 0
deposit of staker_1 in vault: 1000000000000000000
ceTokenBalanceOf(staker_1): 1000000000000000000
balance in ceToken: 1000000000000000000
deposited in aMATICc: 1000000000000000000
       ✓ update ratio -> deposit (40ms)
-------balance after ratio was changed-------
current ratio: 800000000000000000
principal(staker_1): 800000000000000000
yieldFor(staker_1): 200000000000000000
principal + yield(staker_1): 1000000000000000000
total in vault(amaticc): 1000000000000000000
max to withdrawal: 800000000000000000
claimed of staker_1 in vault: 0
deposit of staker_1 in vault: 1000000000000000000
ceTokenBalanceOf(staker_1): 1000000000000000000
balance in ceToken: 1000000000000000000
deposited in aMATICc: 1000000000000000000
       ✓ update ratio
-------balances after yields were claimed-------
current ratio: 800000000000000000
principal(staker_1): 800000000000000000
yieldFor(staker_1): 0
principal + yield(staker_1): 800000000000000000
total in vault(amaticc): 800000000000000000
max to withdrawal: 800000000000000000
claimed of staker_1 in vault: 200000000000000000
deposit of staker_1 in vault: 1000000000000000000
ceTokenBalanceOf(staker_1): 1000000000000000000
balance in ceToken: 1000000000000000000
deposited in aMATICc: 1000000000000000000
       ✓ claim yields (65ms)
-------balance after ratio was changed-------
current ratio: 700000000000000000
principal(staker_1): 700000000000000000
yieldFor(staker_1): 100000000000000000
principal + yield(staker_1): 800000000000000000
total in vault(amaticc): 800000000000000000
max to withdrawal: 700000000000000000
claimed of staker_1 in vault: 200000000000000000
deposit of staker_1 in vault: 1000000000000000000
ceTokenBalanceOf(staker_1): 1000000000000000000
balance in ceToken: 1000000000000000000
deposited in aMATICc: 1000000000000000000
       ✓ update ratio -> print balances
       ✓ try to withdrawal more than have and owner of cetoken (48ms)
to withdrawal: 700000000000000000
withdrawn: 700000000000000000
-------balance after withdrawal full-------
current ratio: 700000000000000000
principal(staker_1): 0
yieldFor(staker_1): 100000000000000000
principal + yield(staker_1): 100000000000000000
total in vault(amaticc): 100000000000000000
max to withdrawal: 0
claimed of staker_1 in vault: 200000000000000000
deposit of staker_1 in vault: 300000000000000000
ceTokenBalanceOf(staker_1): 0
balance in ceToken: 0
deposited in aMATICc: 1000000000000000000
       ✓ try to withdrawal more than have and owner of cetoken (39ms)
-------balances after yields were claimed-------
current ratio: 700000000000000000
principal(staker_1): 0
```

```
yieldFor(staker_1): 0
principal + yield(staker_1): 0
total in vault(amaticc): 0
max to withdrawal: 0
claimed of staker_1 in vault: 300000000000000000
deposit of staker_1 in vault: 300000000000000000
ceTokenBalanceOf(staker_1): 0
balance in ceToken: 0
deposited in aMATICc: 1000000000000000000
       ✓ claim left yields
-------balance after ratio was changed-------
current ratio: 600000000000000000
principal(staker_1): 0
yieldFor(staker_1): 0
principal + yield(staker_1): 0
total in vault(amaticc): 0
max to withdrawal: 0
claimed of staker_1 in vault: 300000000000000000
deposit of staker_1 in vault: 300000000000000000
ceTokenBalanceOf(staker_1): 0
balance in ceToken: 0
deposited in aMATICc: 1000000000000000000
       ✓ update ratio(ratio_4)
-------balance after deposit 1 MATIC from staker_1-------
current ratio: 600000000000000000
principal(staker_1): 999999999999999999
yieldFor(staker_1): 1
principal + yield(staker_1): 1000000000000000000
total in vault(amaticc): 1000000000000000000
max to withdrawal: 999999999999999999
claimed of staker_1 in vault: 300000000000000000
deposit of staker_1 in vault: 1300000000000000000
ceTokenBalanceOf(staker_1): 1666666666666666666
balance in ceToken: 1666666666666666666
deposited in aMATICc: 2000000000000000000
       ✓ deposit more(1MATIC)
-------balance after ratio was changed-------
current ratio: 300000000000000000
principal(staker_1): 499999999999999999
yieldFor(staker_1): 500000000000000001
principal + yield(staker_1): 1000000000000000000
total in vault(amaticc): 1000000000000000000
max to withdrawal: 499999999999999999
claimed of staker_1 in vault: 300000000000000000
deposit of staker_1 in vault: 1300000000000000000
ceTokenBalanceOf(staker_1): 1666666666666666666
balance in ceToken: 1666666666666666666
deposited in aMATICc: 2000000000000000000
       ✓ update ratio(ratio_5)
withdrawn: 499999999999999999
-------balance after withdrawal full-------
current ratio: 300000000000000000
principal(staker_1): 0
yieldFor(staker_1): 0
principal + yield(staker_1): 0
total in vault(amaticc): 0
max to withdrawal: 0
claimed of staker_1 in vault: 800000000000000001
deposit of staker_1 in vault: 800000000000000001
ceTokenBalanceOf(staker_1): 0
balance in ceToken: 0
deposited in aMATICc: 2000000000000000000
       ✓ claim yields and then withdrawal (50ms)

  Sikka Provider
    Basic functionality
------- balance after staker_1 provided(20000000020000000000 MATIC) -------
MATIC balance(staker_1): 89999959997388436037872858
balance of staker_1 in aMATICc: 0
balance of ce_vault in aMATICc: 0
balance of staker_1 in sMATIC: 20000000020000000000
supply sMATIC: 20000000020000000000
balance of staker_1 in ceToken: 0
supply ceToken: 0
yield for staker_1: 0
yield for sikka: 0
current ratio: 600000000000000000
       ✓ staker_1 provides MATIC (147ms)
------- balance after staker_1 provided(20000000020000000000 MATIC) -------
MATIC balance(staker_1): 89999969997044646035122378
balance of staker_1 in aMATICc: 0
balance of ce_vault in aMATICc: 0
balance of staker_1 in sMATIC: 10000000000000000000
supply sMATIC: 10000000000000000000
balance of staker_1 in ceToken: 0
supply ceToken: 0
yield for staker_1: 0
yield for sikka: 0
current ratio: 600000000000000000
       ✓ staker_1 releases MATIC (104ms)
    Dao functionality
       ✓ daoBurn()
       ✓ daoMint()
    Updating functionality
       ✓ change Dao and verify allowances
       ✓ change ceToken and verify allowances (53ms)
       ✓ change collateral token
       ✓ dao token(non-transferable)

  ===GemJoin===
    --- initialize()
       ✓ initialize
    --- rely()
       ✓ reverts: GemJoin/not-authorized
       ✓ relies on address
    --- deny()
       ✓ reverts: GemJoin/not-authorized
       ✓ denies an address
    --- cage()
       ✓ cages
    --- join()
       ✓ reverts: GemJoin/not-live
       ✓ reverts: GemJoin/overflow
       ✓ joins sikka erc20 (57ms)
    --- exit()
       ✓ reverts: GemJoin/overflow
       ✓ exits sikka erc20 (69ms)

  ===SikkaJoin===
    --- initialize()
       ✓ initialize
    --- rely()
       ✓ reverts: SikkaJoin/not-authorized
       ✓ relies on address
    --- deny()
       ✓ reverts: SikkaJoin/not-authorized
       ✓ denies an address
    --- cage()
       ✓ cages
    --- join()
       ✓ joins sikka erc20 (125ms)
    --- exit()
       ✓ reverts: SikkaJoin/not-live
       ✓ exits sikka erc20 (103ms)

  ===Jug===
    --- initialize()
       ✓ initialize
    --- init()
       ✓ reverts: Jug/ilk-already-init
       ✓ inits an ilk
    --- rely()
       ✓ reverts: Jug/not-authorized
       ✓ relies on address
     --- deny()
       ✓ reverts: Jug/not-authorized
       ✓ denies an address
     --- file(3)
       ✓ reverts: Jug/rho-not-updated
       ✓ reverts: Jug/file-unrecognized-param
       ✓ sets duty rate
     --- file(2a)
       ✓ reverts: Jug/file-unrecognized-param
       ✓ sets base rate
     --- file(2b)
       ✓ reverts: Jug/file-unrecognized-param
       ✓ sets vow (38ms)
     --- drip()
       ✓ drips a rate (51ms)

   ===Spot===
     --- initialize()
       ✓ initialize
     --- rely()
       ✓ reverts: Sikka/not-authorized
       ✓ relies on address
     --- deny()
       ✓ reverts: Sikka/not-authorized
       ✓ denies an address
     --- mint()
       ✓ reverts: Sikka/mint-to-zero-address
       ✓ reverts: Sikka/cap-reached
       ✓ mints sikka to an address
     --- burn()
       ✓ reverts: Sikka/burn-from-zero-address
       ✓ reverts: Sikka/insufficient-balance
       ✓ reverts: Sikka/insufficient-allowance
       ✓ burns with allowance
       ✓ burns from address
     --- transferFrom()
       ✓ reverts: Sikka/transfer-from-zero-address
       ✓ reverts: Sikka/transfer-to-zero-address
       ✓ reverts: Sikka/insufficient-balance
       ✓ reverts: Sikka/insufficient-allowance
       ✓ transferFrom with allowance
       ✓ transferFrom an address
     --- transfer()
       ✓ transfers to an address
     --- push()
       ✓ pushes to an address
     --- pull()
       ✓ pulls from an address
     --- move()
       ✓ move between addresses
     --- increaseAllowance()
       ✓ increases allowance
     --- decreaseAllowance()
       ✓ reverts: Sikka/decreased-allowance-below-zero
       ✓ decreases allowance
     --- setSupplyCap()
       ✓ reverts: Sikka/more-supply-than-cap
       ✓ sets the cap
     --- updateDomainSeparator()
       ✓ sets domain separator

   ===Spot===
     --- initialize()
       ✓ initialize
     --- rely()
       ✓ reverts: Spot/not-authorized
       ✓ relies on address
     --- deny()
       ✓ reverts: Spot/not-authorized
       ✓ denies an address
     --- file(3a)
       ✓ reverts: Spotter/not-live
       ✓ reverts: Spotter/file-unrecognized-param
       ✓ sets pip
     --- file(2)
       ✓ reverts: Spotter/not-live
       ✓ reverts: Spotter/file-unrecognized-param
       ✓ sets par
     --- file(3b)
       ✓ reverts: Spotter/not-live
       ✓ reverts: Spotter/file-unrecognized-param
       ✓ sets mat
     --- poke()
       ✓ pokes new price of an ilk (48ms)

   ===Vat===
     --- initialize()
       ✓ initialize
     --- rely()
       ✓ reverts: Vat/not-authorized
       ✓ reverts: Vat/not-live
       ✓ relies on address
     --- deny()
       ✓ reverts: Vat/not-authorized
       ✓ reverts: Vat/not-live
       ✓ denies an address
     --- behalf()
       ✓ reverts: Vat/not-authorized
       ✓ behalfs an address
     --- regard()
       ✓ reverts: Vat/not-authorized
       ✓ regards an address
     --- hope()
       ✓ hopes on address
     --- nope()
       ✓ nopes on address
     --- wish()
       ✓ bit == usr
       ✓ can[bit][usr] == 1
     --- init()
       ✓ reverts: Vat/ilk-already-init
       ✓ initialize a new ilk
     --- file(2)
       ✓ reverts: Vat/not-live
       ✓ reverts: Vat/file-unrecognized-param
       ✓ sets Line
     --- file(3)
       ✓ reverts: Vat/not-live
       ✓ reverts: Vat/file-unrecognized-param
       ✓ sets spot
       ✓ sets line
       ✓ sets dust
     --- slip()
       ✓ slips an amount
     --- flux()
       ✓ reverts: Vat/not-allowed
       ✓ flux an amount
     --- move()
       ✓ reverts: Vat/not-allowed
       ✓ flux an amount (67ms)
     --- frob()
       ✓ reverts: Vat/not-allowed
       ✓ reverts: Vat/ilk-not-init
       ✓ reverts: Vat/ceiling-exceeded
       ✓ reverts: Vat/not-safe (55ms)
       ✓ reverts: Vat/not-allowed-u (53ms)
       ✓ reverts: Vat/not-allowed-v (58ms)
       ✓ reverts: Vat/not-allowed-w (68ms)
       ✓ reverts: Vat/dust (61ms)
       ✓ frobs collateral and frobs stablecoin (70ms)
     --- fork()
       ✓ reverts: Vat/not-allowed (108ms)
```

```
      ✓ reverts: Vat/not-safe-src Vat/not-safe-dst (119ms)
      ✓ reverts: Vat/dust-src Vat/dust-dst (119ms)
      ✓ forks between two addresses (114ms)
    --- grab()
      ✓ grabs ink and art of an address (83ms)
    --- suck()
      ✓ sucks more sikka for an address for sin on another
    --- heal()
      ✓ heals sin of a caller
    --- fold()
      ✓ reverts: Vat/not-live
      ✓ reverts: Vat/not-live

  ===Vow===
    --- initialize()
      ✓ initialize
    --- rely()
      ✓ reverts: Vow/not-authorized
      ✓ reverts: Vow/not-live (41ms)
      ✓ relies on address
    --- deny()
      ✓ reverts: Vow/not-authorized
      ✓ denies an address (41ms)
    --- file(2a)
      ✓ reverts: Vow/not-authorized
      ✓ reverts: Vow/file-unrecognized-param
      ✓ sets hump
    --- file(2b)
      ✓ reverts: Vow/not-authorized
      ✓ reverts: Vow/file-unrecognized-param
      ✓ sets multisig
      ✓ sets sikkaJoin (38ms)
      ✓ sets sikka
      ✓ sets vat
    --- heal()
      ✓ reverts: Vow/insufficient-surplus
      ✓ reverts: Vow/insufficient-debt (88ms)
      ✓ reverts: Vow/not-authorized (120ms)
    --- feed()
      ✓ feeds surplus sikka to vow (147ms)
    --- flap()
      ✓ reverts: Vow/insufficient-surplus (125ms)
      ✓ flaps sikka to multisig (212ms)
    --- cage()
      ✓ reverts: Vow/not-live (41ms)

  ===IkkaRewards===
    --- initialize()
      ✓ initialize
    --- rely()
      ✓ reverts: Rewards/not-authorized
      ✓ reverts: Rewards/not-live
      ✓ relies on address
    --- deny()
      ✓ reverts: Rewards/not-authorized
      ✓ reverts: Rewards/not-live
      ✓ denies an address
    --- cage()
      ✓ disables the live flag
    --- uncage()
      ✓ enables the live flag
    --- initPool()
      ✓ reverts: Reward/not-enough-reward-token
      ✓ reverts: Reward/pool-existed
      ✓ reverts: Reward/invalid-token
      ✓ inits a pool
    --- setIkkaToken()
      ✓ reverts: Reward/invalid-token
      ✓ sets ikka token address
    --- setRewardsMaxLimit()
      ✓ reverts: Reward/not-enough-reward-token
      ✓ sets rewards max limit
    --- setOracle()
      ✓ reverts: Reward/invalid-oracle
      ✓ sets oracle
    --- setRate()
      ✓ reverts: Reward/pool-existed
      ✓ reverts: Reward/invalid-token
      ✓ reverts: Reward/negative-rate
      ✓ reverts: Reward/high-rate
      ✓ sets rate
    --- ikkaPrice()
      ✓ returns ikka price
    --- rewardsRate()
      ✓ returns token  rate
    --- drop()
      ✓ returns if rho is 0
      ✓ drops rewards (359ms)
    --- distributionApy()
      ✓ returns token APY

  ===IkkaToken===
    --- initialize()
      ✓ initialize
    --- rely()
      ✓ reverts: IkkaToken/not-authorized
      ✓ reverts: IkkaToken/invalid-address
      ✓ relies on address
    --- deny()
      ✓ reverts: IkkaToken/not-authorized
      ✓ reverts: IkkaToken/invalid-address
      ✓ denies an address
    --- mint()
      ✓ reverts: IkkaToken/rewards-oversupply
      ✓ mints sikka to an address
    --- burn()
      ✓ burns from address
    --- pause()
      ✓ pauses transfers
    --- unpause()
      ✓ unpauses transfers

  Interaction
    Basic functionality
      ✓ revert:: whitelist: only authorized account can enable whitelist
      ✓ whitelist: should let authorized account enable whitelist
      ✓ revert:: deposit(): cannot deposit collateral for inactive collateral type
      ✓ deposit(): should let user deposit collateral (76ms)
      ✓ revert:: deposit(): only whitelisted account can deposit (57ms)
      ✓ revert:: deposit(): should not let user deposit collateral directly to interaction (58ms)
      ✓ withdraw(): should let user withdraw (104ms)
      ✓ revert:: withdraw(): Caller must be the same address as participant(!sikkaProvider) (88ms)
      ✓ revert:: withdraw(): Caller must be the same address as participant(!sikkaProvider) (88ms)
      ✓ borrow(): should let user borrow (158ms)
      ✓ borrow(): should let user borrow and compare getter function values (236ms)
      ✓ revert:: borrow(): should not let borrow more than available (123ms)
      ✓ revert:: withdraw(): should not let withdraw when user has outstanding debt (157ms)
      ✓ payback(): should let user payback outstanding debt(borrowed sikka) (197ms)
      ✓ revert:: payback(): should revert if user leave dust (178ms)
      ✓ auction started as expected (187ms)
      ✓ revert:: cannot reset auction if status is false(redo not required) (229ms)
      ✓ auction works as expected (622ms)
      ✓ stringToBytes32(): should convert string to bytes32
      ✓ stringToBytes32(): should return 0x00 for empty string
      ✓ removeCollateralType(): should remove an active collateral type (50ms)
      ✓ revert:: removeCollateralType(): cannot remove an inactive collateral type
    setters
      ✓ revert:: whitelist: only authorized account can enable whitelist
      ✓ whitelist: should let authorized account enable whitelist
      ✓ revert:: whitelist: only authorized account can disable whitelist
      ✓ whitelist: should let authorized account enable whitelist
      ✓ revert:: whitelist: only authorized account can set whitelist operator
      ✓ whitelist: should let authorized account set whitelist operator
```

```
       ✓ revert:: whitelist: only authorized account can add an account to whitelist
       ✓ whitelist: should whitelist operator add account to whitelist
       ✓ revert:: whitelist: only authorized account can remove an account from whitelist
       ✓ whitelist: should whitelist operator remove account to whitelist
       ✓ setCores(): only authorized account can set core contracts
       ✓ setCores(): should let authorized account set core contracts
       ✓ setSikkaApprove(): only authorized account can set core contracts
       ✓ setSikkaApprove(): should let authorized account set core contracts
       ✓ setCollateralType(): only authorized account can set core contracts
       ✓ setCollateralType(): collateral type can be set for once only (44ms)
       ✓ setRewards(): only authorized account can set core contracts
       ✓ setRewards(): should let authorized account set core contracts

  ===Jar===
    --- Setters
       ✓ checks setters
    --- Extract Dust and Cage
       ✓ checks extracting dust (54ms)
    --- cage()
       ✓ denies an address
    --- rely()
       ✓ reverts: Jar/not-authorized
       ✓ relies on address
    --- deny()
       ✓ reverts: Jar/not-authorized
       ✓ denies an address
    ---Full Flow
       ✓ checks join/exit flow (280ms)

  MasterVault
    Basic functionality
       ✓ reverts:: Deposit 0 amount
       ✓ Deposit: valid amount (81ms)
       ✓ Deposit: valid amount(when swapFee is set in swapPool) (75ms)
       ✓ Deposit: wMatic balance of master vault should increase by deposit amount (118ms)
       ✓ Deposit: wMatic balance of master vault should increase by deposit amount(deposit fee: 0) (92ms)
       ✓ Deposit: totalsupply of master vault should increase by amount(deposit fee: 0) (99ms)
       ✓ Deposit: totalsupply of master vault should increase by amount(deposit fee: 0.1%) (115ms)
       ✓ Allocate: wMatic balance should match allocation ratios (100ms)
       ✓ Allocate: wMatic balance should match allocation ratios (97ms)
       ✓ Allocate: wMatic balance should match allocation ratios(deposit fee: 0.1%) (111ms)
       ✓ revert:: withdraw: should revert if withdrawal amount is more than vault-token balance(depositAmount)
       ✓ withdraw: should let user withdraw (withdrawal fee: 0) (96ms)
       ✓ withdrawFromStrategy(): should let owner withdraw from strategy (209ms)
       ✓ revert:: withdrawFromStrategy(): only owner can withdraw from strategy (107ms)
       ✓ revert:: withdrawFromStrategy(): only owner can withdraw from strategy (100ms)
       ✓ revert:: withdrawFromStrategy(): only owner can withdraw from strategy (115ms)
       ✓ withdrawAllFromStrategy(): should let owner withdraw all from strategy (146ms)
       ✓ withdraw: should let user withdraw (withdrawal fee: 0.1%) (128ms)
       ✓ withdraw: should let user withdraw when funds are allocated to strategy (withdrawal fee: 0.1%) (203ms)
       ✓ withdrawFee: should let user withdraw when funds are allocated to strategy (withdrawal fee: 0.1%) (210ms)
       ✓ withdraw: should let user withdraw (withdrawal fee: 0.1%) (187ms)
       ✓ withdraw: withdrawal request should go to the waiting pool(withdrawal fee: 0) (177ms)
       ✓ payDebt: should pay the pending withdrawal (withdrawal fee: 0) (213ms)
       ✓ payDebt: should pay the pending withdrawal (withdrawal fee: 0) (243ms)
       ✓ revert:: waitingPool: withdrawUnsettled(): cannot withdraw already settled debt (208ms)
       ✓ retireStrat(): should withdraw all the assets from given strategy (103ms)
       ✓ retireStrat(): should mark strategy inactive if debt is less than 10 (112ms)
       ✓ migrateStrategy(): should withdraw all the assets from given strategy (281ms)
       ✓ depositAllToStrategy(): should deposit all the assets to given strategy (104ms)
       ✓ depositToStrategy(): should deposit given amount of assets to given strategy (102ms)
       ✓ depositAllToStrategy(): should deposit all the assets to given strategy (103ms)
       ✓ revert:: deposit(): should revert
       ✓ revert:: mint(): should revert
       ✓ revert:: withdraw(): should revert
       ✓ revert:: redeem(): should revert
      setters
         ✓ revert:: setDepositFee(): cannot set more than max
         ✓ setDepositFee(): should let owner set new fee
         ✓ revert:: setWithdrawalFee(): cannot set more than max
         ✓ setWithdrawalFee(): should let owner set new fee
         ✓ revert:: setWaitingPool(): cannot set zero address
         ✓ revert:: setWaitingPool(): onlyOwner can call
         ✓ setWaitingPool(): should let set new waiting pool
         ✓ revert:: addManager(): onlyOwner can call
         ✓ revert:: addManager(): cannot set zero address
         ✓ addManager(): should let add new manager
         ✓ revert:: removeManager(): cannot set zero address
         ✓ removeManager(): should let owner remove manager
         ✓ revert:: changeProvider(): cannot set zero address
         ✓ changeProvider(): should let owner change provider address
         ✓ revert:: changeStrategyAllocation(): cannot change allocation of zero address
         ✓ changeStrategyAllocation(): should let owner change allocation
         ✓ revert:: changeStrategyAllocation(): cannot change allocation to more than 100%
         ✓ revert:: setStrategy(): cannot set already existing strategy
         ✓ revert:: setStrategy(): cannot set already existing strategy
         ✓ revert:: setWaitingPoolCap(): onlyOwner can call
         ✓ revert:: setWaitingPoolCap(): should let owner set waiting pool cap limit
         ✓ revert:: setCapLimit(): onlyMasterVault can call
         ✓ revert:: setCapLimit(): cannot be zero
         ✓ revert:: setCapLimit(): onlyMasterVault can call
         ✓ revert:: changeSwapPool(): onlyOwner can call
         ✓ revert:: changeSwapPool(): cannot set zero address
         ✓ changeSwapPool(): should let owner change swapPool
         ✓ revert:: changeFeeReceiver(): onlyOwner can call
         ✓ revert:: changeFeeReceiver(): cannot set zero address
         ✓ revert:: initialize(): cannot set maxDeposit more than 1e6
         ✓ rrevert:: initialize(): cannot set maxDeposit more than 1e6
         ✓ changeFeeReceiver(): should let owner change swapPool
      CerosYieldConverterStrategy: setters
         ✓ revert:: changeSwapPool(): onlyOwner can call
         ✓ revert:: changeCeRouter(): onlyOwner can call
         ✓ revert:: changeCeRouter(): cannot set zero address
         ✓ changeCeRouter(): should let owner change ceRouter
         ✓ changeSwapPool(): should let owner change swapPool
         ✓ changeSwapPool(): cannot set to zero address
         ✓ revert:: setStrategist(): onlyOwner can call
         ✓ revert:: setStrategist(): cannot set zero address
         ✓ setStrategist(): should let owner change Strategist
         ✓ revert:: setStrategist(): onlyOwner can call
         ✓ revert:: setFeeRecipient(): cannot set zero address
         ✓ setFeeRecipient(): should let owner change feeRecipient
         ✓ revert:: withdraw(): only masterVault can withdraw funds
         ✓ revert:: pause(): onlyStrategist can call
         ✓ pause(): should let Strategist pause deposits
         ✓ pause(): cannot deposit when paused (41ms)
         ✓ unpause(): should let Strategist unpause deposits
         ✓ harvest(): should let strategiest harvest(claim yeild) (143ms)
         ✓ harvestAndSwap(): should let strategiest harvest(claim yeild in wMatic) (144ms)

  SwapPool
    # add/remove(userType)
      reverts:
         ✓ if want to add zero address
         ✓ if want to add manager from non-owner account
         ✓ if want to add liquidity provider or integrator from non-manager account
         ✓ if want to remove zero address
         ✓ if want to remove manager from non-owner account
         ✓ if want to add liquidity provider or integrator from non-manager account
      works:
         ✓ add manager (52ms)
         ✓ add liquidity provider (66ms)
         ✓ add integrator (62ms)
         ✓ remove manager (60ms)
         ✓ remove liquidity provider (77ms)
         ✓ remove integrator (70ms)
    # setFee
      reverts:
         ✓ only owner or manager can call this function
         ✓ only owner can add Owner fee
         ✓ fee should be less than FEE_MAX
         ✓ fee sum is more than 100% (66ms)
      works:
        ✓ fee changing works (96ms)
    # enableIntegratorLock/enableProviderLock/excludeFromFee
      reverts:
        ✓ when the caller is not owner or manager
      works:
        ✓ works as expected (72ms)
    # addLiquidity
      reverts:
        ✓ when add first liquidity less than 1 token
        ✓ if providerLockEnabled and the caller is not integrator
      works:
        ✓ add the first liquidity with ratio 1
        ✓ addLiquidityEth works ok too
        ✓ add liquidity when providerLockEnabled and the caller is is provider (49ms)
        ✓ add the first different amounts with ratio 1
        ✓ add first liquidity with ratio 0.98
        ✓ add first liquidity with ratio 0.98 with less ratio token amount
        ✓ add liquidity second time (71ms)
    # swap
      reverts:
        ✓ when integratorLock is enabled and caller is not integrator
        ✓ when not enough liquidity
      works:
        ✓ swap: swaps native to ceros and gets the expected amount (74ms)
        ✓ swapEth: swaps native to ceros and gets the expected amount (58ms)
        ✓ swap: swaps ceros to native and gets the expected amount (69ms)
        ✓ swapEth: swaps ceros to native and gets the expected amount (63ms)
        ✓ swap to receiver address works(native to ceros) (69ms)
        ✓ swap to receiver address works(ceros to native) (69ms)
        ✓ swaps and transfers integrator fee if integratorLock is enabled (native to ceros) (94ms)
        ✓ swaps and transfers integrator fee if integratorLock is enabled (ceros to native) (89ms)
        ✓ will not charge a fee if msg.sender is excluded from fee(native to ceros) (72ms)
        ✓ will not charge a fee if msg.sender is excluded from fee(ceros to native) (69ms)
    # removeLiquidity/removeLiquidityPercent
      reverts:
        ✓ removeLiquidity: if lpAmount is 0
        ✓ removeLiquidity: if userBalance is less than lpAmount
        ✓ removeLiquidityPercent: if percent is more than 10**18 or it is 0
      works:
        ✓ removeLiquidity: remove liquidity with ratio 1
        ✓ removeLiquidityEth: remove liquidity with ratio 1
        ✓ removeLiquidityPercent: remove liquidity with ratio 1
        ✓ removeLiquidityPercentEth: remove liquidity with ratio 1
        ✓ removeLiquidityPercent: should remove all liquidity (39ms)
        ✓ removeLiquidity: should remove all liquidity (39ms)
        ✓ removeLiquidity: remove after swapping (65ms)
        ✓ removeLiquidityPercent: remove after swapping (59ms)
    # withdrawOwnerFee
      reverts:
        ✓ when the caller is not the owner
        ✓ when owner want to remove more than have(except MaxUint256)
      works
        ✓ withdrawOwnerFee full amount works
        ✓ withdrawOwnerFeeEth full amount works
        ✓ withdrawOwnerFee partially works
        ✓ withdrawOwnerFee will NOT fail if amounts are 0
    # withdrawManagerFee
      reverts:
        ✓ when the caller is not the manager
      works:
        ✓ withdrawing fee works as expected for 1 manager (38ms)
        ✓ withdrawing fee Eth works as expected for 1 manager
        ✓ withdrawing fee works as expected for 1 manager(swap after adding managers) (105ms)
        ✓ withdrawing fee works as expected for many managers (312ms)
        ✓ withdrawing fee works as expected for many managers(adding and removing) (470ms)

===INTERACTION-Multicollateral===
  - defaults
  - put collateral and borrow
  - payback and withdraw
  - drip
  - rewards

sikka-protocol
  provide, borrow, payback & release flow
      ✓ signer1 provides MATIC (623ms)
  DAO functionality
      ✓ auction flow (531ms)


419 passing (50s)
10 pending
```

Code Coverage

The average of the coverage 66.5%, however we advise the team to aim for a minimum of a 90% coverage.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	86.86	72.62	85.66	88.6	
IkkaRewards.sol	75	77.27	80.95	71.62	… 182,184,185
IkkaToken.sol	100	100	100	100	
Interaction.sol	95.04	69.44	93.88	95.87	… 464,489,508
abaci.sol	20	15	24	34.48	… 230,238,273
clip.sol	77.78	56.76	88.89	78.42	… 461,462,463
dog.sol	79.69	46.88	82.35	87.93	… 228,232,233
jar.sol	97.59	75	95.45	97.65	184,185
join.sol	91.11	87.5	88.24	91.67	95,96,154,155
jug.sol	100	92.86	100	100	
sMath.sol	100	100	100	100	
sikka.sol	86.89	75	94.44	86.89	… 147,148,149
spot.sol	96.43	100	91.67	95.65	114
vat.sol	**98.95**	**100**	**96.77**	**98.86**	**144**
vow.sol	95.24	91.67	91.67	93.55	114,115

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/MasterVault/	96.55	79.41	95.74	96.65	
MasterVault.sol	99	81.15	95	99.02	461,465
WaitingPool.sol	81.25	64.29	100	82.86	… 102,103,104
contracts/MasterVault/interfaces/	100	100	100	100	
IMasterVault.sol	100	100	100	100	
IWETH.sol	100	100	100	100	
IWaitingPool.sol	100	100	100	100	
contracts/Strategy/	89.74	71.05	86.36	88.89	
BaseStrategy.sol	84.21	100	70	85	56,60,64
CerosYieldConverterStrategy.sol	91.53	65.63	100	90.16	… 108,139,140
IBaseStrategy.sol	100	100	100	100	
contracts/ceros/	90.87	71.62	84.09	91.06	
CeToken.sol	88.89	75	83.33	90	54
CeVault.sol	94.83	92.86	86.36	94.92	203,206,209
CerosRouter.sol	89.25	50	91.67	89.25	… 183,193,242
NonTransferableERC20.sol	68	100	46.67	69.23	… 113,131,154
SikkaProvider.sol	100	77.78	100	100	
sMATIC.sol	100	75	100	100	
contracts/ceros/interfaces/	100	100	100	100	
IBondToken.sol	100	100	100	100	
ICerosRouter.sol	100	100	100	100	
ICertToken.sol	100	100	100	100	
IDao.sol	100	100	100	100	
IDex.sol	100	100	100	100	
IMaticPool.sol	100	100	100	100	
IPriceGetter.sol	100	100	100	100	
ISikkaProvider.sol	100	100	100	100	
ISwapPool.sol	100	100	100	100	
ISwapRouter.sol	100	100	100	100	
IVault.sol	100	100	100	100	
contracts/interfaces/	100	100	100	100	
ClipperLike.sol	100	100	100	100	
DexV3Like.sol	100	100	100	100	
DogLike.sol	100	100	100	100	
GemJoinLike.sol	100	100	100	100	
GemLike.sol	100	100	100	100	
IAuctionProxy.sol	100	100	100	100	
IColanderRewards.sol	100	100	100	100	
IMovingWindowOracle.sol	100	100	100	100	
IRewards.sol	100	100	100	100	
ISikka.sol	100	100	100	100	
IStabilityPool.sol	100	100	100	100	
JugLike.sol	100	100	100	100	
PipLike.sol	100	100	100	100	
SikkaJoinLike.sol	**100**	**100**	**100**	**100**	
SpotLike.sol	100	100	100	100	

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
VatLike.sol	100	100	100	100	
contracts/libraries/	93.18	100	100	93.18	
AuctionProxy.sol	93.18	100	100	93.18	60,61,62
ERC20ModUpgradeable.sol	6	0	10	5.56	… 328,329,333
MaticPool.sol	0	0	0	0	… 129,133,137
aMATICb.sol	8.13	0	11.11	7.81	… 388,404,420
aMATICc.sol	47.37	0	30	45	… 77,85,86,90
multicalls.sol	100	100	100	100	
oracle.sol	80	50	100	80	23
proxyLike.sol	100	100	50	66.67	31,37
wMATIC.sol	0	0	0	0	… 26,27,28,29
contracts/swapPool/	86.75	85.06	87.23	86.67	
LP.sol	83.33	50	80	85.71	21
SwapPool.sol	86.82	85.88	88.1	86.69	… 721,722,723
contracts/swapPool/interfaces/	100	100	100	100	
ICerosToken.sol	100	100	100	100	
ILP.sol	100	100	100	100	
IMaticPool.sol	100	100	100	100	
INativeERC20.sol	100	100	100	100	
contracts/swapPool/mocks/	50	50	50	46.67	
CerosToken.sol	60	100	66.67	60	22,26
Token.sol	0	100	0	0	10,14,18,22,26
WNative.sol	100	50	100	80	12
PriceGetter.sol	0.97	0	10	1.05	… 350,359,360
IPool.sol	100	100	100	100	
IUniswapV3Factory.sol	100	100	100	100	
BitMath.sol	0	0	0	0	… 87,88,90,92
FixedPoint128.sol	100	100	100	100	
FixedPoint96.sol	100	100	100	100	
FullMath.sol	0	0	0	0	… 119,120,121
LiquidityMath.sol	0	0	0	0	11,12,14
LowGasSafeMath.sol	0	0	0	0	12,20,28,36,44
SafeCast.sol	0	0	0	0	11,18,25,26
SqrtPriceMath.sol	0	0	0	0	… 190,206,222
SwapMath.sol	0	0	0	0	… 92,95,97,99
TickMath.sol	0	0	0	0	… 200,201,203
UnsafeMath.sol	100	100	0	0	13
All files	**66.65**	**56.07**	**65.44**	**66.37**	

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

```
e75c8db111960689ed1214775c7b28bcfe3fd8639ee6579f26d2ef9cca64021c ./contracts/abaci.sol
ff7fea861434e59e4b383c10187a8331cc4fb04c4fff37c91bae2b13bb8f010c ./contracts/clip.sol
4fabdc0df33f0f877bbc077c0ddcef353b2fcfd4b786fd15c73b5dee1265b736 ./contracts/colander.sol
```

```
420a8e2862644590338614107a7203ef8f51d01e5ed3133b084374f2ea958cb1 ./contracts/dog.sol
ff99fc87e0855c3c66e3649a25ac4fc2ae36cf89336f7fa7b87c2ae0b3f00ecf ./contracts/IkkaRewards.sol
4a2822d4efb6907a7c579265b1d4c73806c36920c5123034a10145a62798b480 ./contracts/IkkaToken.sol
0d41e61ff664ab2974b3e7e83388a2e189980a0df7d737c193caa81705884f30 ./contracts/Interaction.sol
74648f00d5a71ae834fd4307f90c40b605fe138db78cecc21002c6465508a008 ./contracts/jar.sol
06d8d2122c8501ba3caef86992a76d28da675483d8ad54295559bba748ac58a5 ./contracts/join.sol
35bf6d329325fda173fec8a18186dc56a73f3da3e09f86165ed6b0af8a70d7cc ./contracts/jug.sol
8996c21eef834fedd3ae087ac05161f52b60614f1ba4213ba8731510025a1cf2 ./contracts/sikka.sol
7626e6f4067a3ed43f15d43fffd58e3a932f7931da2b7b19333c52ccfdd9dddc ./contracts/sMath.sol
a073ace0618f8896a3155a5432febffcabd60f86e39fe0816f3c820b1ec0d24e ./contracts/spot.sol
9eb57a2b38f06903d4f608d7dc0a177c04abbcf4ace2ff5622f61ed512a0dab5 ./contracts/vat.sol
0710a4be2fd39e5ffd4a5164208e5485c3c9ae87f4db8a5f5374f00224247534 ./contracts/vow.sol
9512c0660c5270ebb0679924ef8e5357b1092963297e3fff5cc9d63bb701b62e ./contracts/swapPool/LP.sol
0a8b2f3ea4331795899b2e2f8101254b9de27fb74b8db66f26e0524ba4bf4051 ./contracts/swapPool/SwapPool.sol
241bd2813ab0e722b05b64e1c666778059a5ddf80b8f91590d998b3c1416c1a1 ./contracts/swapPool/mocks/CerosToken.sol
140c31c960e9d8e4cd1c8a7d2f049eb1d4285581cbbf25b068dff6cdc5e588be ./contracts/swapPool/mocks/Token.sol
dbd17fc2e51edabb0bb34b5e6bf4c5e5600f09fdcc0f44de96598f59d47b6418 ./contracts/swapPool/mocks/WNative.sol
7dc6c02e45919086197996b7986ff7be9819356793fb0f75a2ea41abe27c79f9 ./contracts/swapPool/interfaces/ICerosToken.sol
33d92640a5da2b6b92351aae9e6b0f652f0d1518e7dde2df4fb10354e3236c5f ./contracts/swapPool/interfaces/ILP.sol
cf9af8cdbea53b78ba6df3ea6a4a3833b9f7e83be6547f942e02c05e125049b2 ./contracts/swapPool/interfaces/IMaticPool.sol
aee2ef4ee2871d89c13b84fc6eb05d3140cf1b38eb46a7dd004c730e79e893ed ./contracts/swapPool/interfaces/INativeERC20.sol
a953e9d8c4d4817408078ba1b7b41748f84c227512ad4fbf176e81c4e54e2d01 ./contracts/Strategy/BaseStrategy.sol
34fa3bba710dc0da307077b8aebb1c207f9aef12e3cb5702792250b23305417c ./contracts/Strategy/CerosYieldConverterStrategy.sol
8b8ddd0e319221b3b24f9038663a9039f0bccfad5ea87b28285320c40f91ceeb ./contracts/Strategy/IBaseStrategy.sol
c38978cd308303dec25bbccedbcadf3d6de3f971488acf2b6237c04f351f6186 ./contracts/oracle/IkkaOracle.sol
4db53c10d3d7e99abe5de28d45daa30e6f588f3245cfb4dfc40f322548dcad7c ./contracts/oracle/MaticOracle.sol
df68ccb316ec617fee8f1b9162d768bb4afccaecb1a1d2e7d68a96b9cb02001e ./contracts/oracle/PriceOracle.sol
5c00a3e4aa2c06fb2fb88ea851b9b082b42ad66c933c764f4c2dc3b4741f44e2 ./contracts/oracle/PriceOracleTestnet.sol
6380a00456180b5acf73b246e5602e4f7ba677f143d40f516ba0dd45443907c0 ./contracts/oracle/SlidingWindowOracle.sol
b2f5a4d2ceea0a1cc09da449a33390966ab951a5b34fd34891d7b3dba7bbfd0d ./contracts/oracle/libraries/Babylonian.sol
4e7ac1e4aff8a491bbe56447065911e293bdb2d841155869b72c8018ad69c92b ./contracts/oracle/libraries/BitMath.sol
ff9f86d5e694722b66b872732cf26aa6f014724626f8765d4517968c1d62b752 ./contracts/oracle/libraries/FixedPoint.sol
f71432885d5e6f369793dbdfe95da311e990c9e5423755eba548ccf5fa37e891 ./contracts/oracle/libraries/FullMath.sol
1f13a7a2ecf5bf1b81cb5f2d3e4111d2777da9c1da8efc5b7261d6dd15ba0ea8 ./contracts/oracle/libraries/hMath.sol
8e082e9afd4a6451209c382fbc225ad25dbc2da6439260492c0c5af7131fa9ba ./contracts/oracle/libraries/UniswapV2Library.sol
2dc81ac61d4e8612c43ddaff3b5ce6ec42b4efca89d4659165f3b4562966f5b1 ./contracts/oracle/libraries/UniswapV2OracleLibrary.sol
21a9c3fe0a2e7a78db7bba1b4220ca2b56180cd6241910fe29ffd4b99eb7654d ./contracts/oracle/interfaces/IUniswapV2Factory.sol
267e62716c5d1cee4519bfa84962fa4e34310e418f3c9ae8b2600c1c30d3903f ./contracts/oracle/interfaces/IUniswapV2Pair.sol
6a739c499397ecbe02c51809beecf7611c32649a33518bcdc009e81e6e87d055 ./contracts/mock/aMATICb.sol
fa8529f1b945bf11dca3c30203a7c72f90aee529a3f88d7866834ee080a709b9 ./contracts/mock/aMATICc.sol
c6af0ac672b6df651b5190f00841382811249058d4bb11e33b45289ac633e55f ./contracts/mock/ERC20ModUpgradeable.sol
2d02d49f10dee39fcccd428315119ca79c353e677b957be32f387429c052a753 ./contracts/mock/MaticPool.sol
cd5e9a26ff5568d81a58b95642c3dcad32bbb43de73481d92d1bccca4f495c46 ./contracts/mock/multicalls.sol
e4144feff8b5f118640934da0ac9c958c9039fe0a1619e79afa4235cd1fdfbc2 ./contracts/mock/oracle.sol
b65e6fb78947d6f8a7b854d0a2e4b283da3e320586f08951eaaef79e50b54531 ./contracts/mock/proxyLike.sol
c921bae9a755786cb9d868dd766f7f2119a59632cec3ec07df2a1d1db4ba3118 ./contracts/mock/wMATIC.sol
f1857cb52beddab05bc677d2b44542a68a44df3d6914f0aa1e945aedc86310f9 ./contracts/mock/dex/PancakeERC20.sol
0562dceb35519937322171a0c50964b8d85273b19056527488ff1ed3b1ed7fd7 ./contracts/mock/dex/PancakeFactory.sol
26de412551611f76cfbe5250ddf37379362ad66f2a97b8cb7a5c17340bfcef77 ./contracts/mock/dex/PancakeMigrator.sol
bafff68c84618204fe1864bfc7f4ff84508a39777aba8bdfab315f081730ad0e ./contracts/mock/dex/PancakePair.sol
07b43035b2ba7378389a3e35798d6a17f09d998299d55aff19897bcd6816f021 ./contracts/mock/dex/PancakeRouter.sol
393f22982b8759bda8f88dd724c1479eb674cdf13194897963767fa63f6e0402 ./contracts/mock/dex/libraries/Babylonian.sol
79c69f52e93e78620f1805d85c381394558a0720ef01eab61d82547b1176d67a ./contracts/mock/dex/libraries/BitMath.sol
094158d0ad33e4cdc12636665ac5c27159ae62e864f24d94c5d36d4e2d92e068 ./contracts/mock/dex/libraries/FixedPoint.sol
2872eafc2d91c1d204042e2135fa294fed24c96bbc67188fce0e1aca37ceffb9 ./contracts/mock/dex/libraries/InitHash.sol
7091407638fc74a5b5147c39ecfbeee35df13916de96e1a39abc0608eeb215fc ./contracts/mock/dex/libraries/Math.sol
5e824fe5da29f21bcc8d08f7a5ece01d68823acf2801f93e83311e5b7ed7ab73 ./contracts/mock/dex/libraries/PancakeLibrary.sol
d4d4d30e6290c30a0bfe0967d4244e33120ab98c9806fec9b5dc78f0afc7c11a ./contracts/mock/dex/libraries/PancakeOracleLibrary.sol
8158851701c8358132fa27375e4573e19c1efad34887f4c0de30ecbe0a110aa2 ./contracts/mock/dex/libraries/SafeMath.sol
737c028b907f7ec27d9608f821883ece7be5c9cbeaac996b22964180219d98d2 ./contracts/mock/dex/libraries/TransferHelper.sol
8bfc230b3269715d2f120a562fd9100a22344504acaf9562fcf03041db6ababe ./contracts/mock/dex/libraries/UQ112x112.sol
```

```
9de1d36d169b3de60036c44395b51d92d75455b3661c8719eea64c97e38d677f ./contracts/mock/dex/interfaces/IPancakeCallee.sol
aa10721c2f9ec38d850d0e0906529c2c686dd3e825883c06825a6372f24c325e ./contracts/mock/dex/interfaces/IPancakeERC20.sol
e130662d4340b84acd3e618ac54a4edd249bb40293bb6331ce972d5471d85755 ./contracts/mock/dex/interfaces/IPancakeFactory.sol
8be38e0d89a7365260aa967e96b55c0e3f74fe42545d81e569982f7fa24f5ded ./contracts/mock/dex/interfaces/IPancakeMigrator.sol
a3318a490116b6c43b9b084654b9f66b4ad92f1cc72ca1961e33dcb6dd7c3875 ./contracts/mock/dex/interfaces/IPancakePair.sol
0a418c60a4e78e0e3c93e04e5e7fc0ae53b6f3f10585d3b05679e0787977ed7d ./contracts/mock/dex/interfaces/IPancakeRouter01.sol
e56fe0c8ef15079e9a43ec552dc7d966c214b9609b63dfd6c66289261cc5b75c ./contracts/mock/dex/interfaces/IPancakeRouter02.sol
83ff5b2e13acd90f2cf24b101631936be3528f4a55e3291535e8c3601614ac3f ./contracts/mock/dex/interfaces/IWETH.sol
4f835090b9de1545ebadc371cbd9210e70a85e0de8660de594da311f48b5135e ./contracts/mock/dex/interfaces/IXERC20.sol
688ff0442d0015bda2214cfcee8d779a4b3a60bf7bd110e95c038b54d62d0c58 ./contracts/mock/dex/interfaces/V1/IUniswapV1Exchange.sol
468a6fa18b3d0e651d87e9704c1807fc671505ec539da3fb370321897402bb7b ./contracts/mock/dex/interfaces/V1/IUniswapV1Factory.sol
8b9106483d640408d51a6d95dbfed51de52d7b0ae222f8d7d625e3d73aaa91ca ./contracts/MasterVault/MasterVault.sol
87265a568b684f37446385f7d4f78f44df557e33009d95372694b0cebbabd767 ./contracts/MasterVault/WaitingPool.sol
4fbb1dd6675c4a22db94f553489b4552a6ac519cc681b381df90ed2678687c25 ./contracts/MasterVault/interfaces/IMasterVault.sol
35afffca92c82221d778fad5e1102e83a135650110b1227e5c5c3df2d2a7e1a6 ./contracts/MasterVault/interfaces/IWaitingPool.sol
275de471313a0b2154adcf8798bba9b44539a3062547e10e31c67b953e22f044 ./contracts/MasterVault/interfaces/IWETH.sol
37c13ce52d34b937ecc8ddd2798f8cb71283362a39b5ab0edddf917248ef545b ./contracts/libraries/AuctionProxy.sol
a113ccccbc3a12b8cddf7437e611a56e9ef7a13ef7a324b5bb97fc06b3af361d ./contracts/interfaces/ClipperLike.sol
9612244c25031a50f95ed1233535a6bc2cb5ffe392132fde3bcde5da913038b0 ./contracts/interfaces/DexV3Like.sol
c8221352a8eb154ce6c83db524edd2967101ef17ddd35777943f96b046674193 ./contracts/interfaces/DogLike.sol
46ba5f33fb1ac7baffa7b36b6e0b7d5a6da0c957bb655b9748bad68c72809e64 ./contracts/interfaces/GemJoinLike.sol
d826d1ed5ba07a1d92aa593c8a528e74ca067a336e48b1419c6797b875441a4b ./contracts/interfaces/GemLike.sol
00f85302bdae03c18c22484bcfe81423a8883cc868798e2118faa668c9bce384 ./contracts/interfaces/IAuctionProxy.sol
c4d2f9e0acc7e54aa7d3a98b7307f5126104a7f24e40670a475b80a2b8c3f723 ./contracts/interfaces/IColanderRewards.sol
f2be3f6d24d70ea3a728880d3f2bb23976c31e61c4993cab9b8ebd16076e52f1 ./contracts/interfaces/IMovingWindowOracle.sol
19ac3b190ed8dd5d07e945a4d80b6c508d99c0c8cd96f24b61f6446414717ede ./contracts/interfaces/IRewards.sol
96c456e29e8056928faee306e9b101d2579b093edd5af62309ba62f1838bebb1 ./contracts/interfaces/IStabilityPool.sol
01fab9eaf6e7261bc22766e3978657457002a78b3201e19aa467865cf7c2674a ./contracts/interfaces/JugLike.sol
0d52a00470f595857304fa1822c94eb28defcf5adbfd5445fce6dccba9e95e42 ./contracts/interfaces/PipLike.sol
e17e4e6caca95aefee2dc6cc0277b9f1ff86a6193b4fe1caa69f4af6a0b999e3 ./contracts/interfaces/SikkaJoinLike.sol
c32254eb6fe7ae250e999c654eb60982e9d74f6fd76392b6349b7290eb98898b ./contracts/interfaces/SikkaLike.sol
10784f4517655afe481a67c2f2b2c4d124eddcb4a3a26bd96f98faba13a1d6ef ./contracts/interfaces/SpotLike.sol
b9d974ba30fd1943b8749a948771ee6e1d843992316cdc77bbf07a96539a2622 ./contracts/interfaces/VatLike.sol
4c47cf017b0089670c98cd4a6c8e7bccc4deef06bb25cb2d6a2ac54f5379e17a ./contracts/ceros/CerosRouter.sol
b2c9dee539a03200e6fdeb8f6be6665820ff7aeac4164343dafdeddd53bc5930 ./contracts/ceros/CeToken.sol
98a0aa963ac83eedebd4948a9a6efaa8ed4462fdf113760a5c2a87572c21805e ./contracts/ceros/CeVault.sol
923f02e77b8b7b52f554b8bac25aff02c7d413428c8e0c56e35d79df9797b9ef ./contracts/ceros/NonTransferableERC20.sol
96d822a58206bc0ae4c6d478edb5fbfb59e6695562e4abf8b28b3e1cee233bd9 ./contracts/ceros/SikkaProvider.sol
2803930f800c216dc91b6843c8138668e4900e60ec10cbe55daf2093fc7d87fd ./contracts/ceros/sMATIC.sol
0c723c64a72a4cb3b9541638db5525dabe207d0f56e3e7661e371a75ea9dbbd0 ./contracts/ceros/interfaces/IBondToken.sol
a22328f9e44f85d9ed0768dff83da3f7b4616c1cc8fe128ea350f3774b10587f ./contracts/ceros/interfaces/ICerosRouter.sol
48738397f36a3371fc6afe349d00c2664ba301e0c927894f314bd4b2eef49d48 ./contracts/ceros/interfaces/ICertToken.sol
d3bbd6f22dc63a509dd94eef297c98da8eb78c37f30d9a83197a3f2548422173 ./contracts/ceros/interfaces/IDao.sol
6bc2956e3fefa2f771b42bc3ce037469dec12c9fef0d1edc760b25f4f37d0390 ./contracts/ceros/interfaces/IDex.sol
55474bf688deedee3a77e325cc0c65ccd6c8dfd7dfafa70f0a4755bd9c62e916 ./contracts/ceros/interfaces/IMaticPool.sol
8c15735172cb51959adc8d63e2fff017c8e342b530f5ccaa22271104a2b14bee ./contracts/ceros/interfaces/IPriceGetter.sol
3354aee4d4259736ce364be27a46ef4430a642627e387fdbcdf2fc1856e70dc9 ./contracts/ceros/interfaces/ISikkaProvider.sol
97420791581341be38356abc850008cddb2458b0c64ad76987f8302fa120e64f ./contracts/ceros/interfaces/ISwapPool.sol
0576983224096bc4b0f81bbd248a2a44ae39e6f39d1875c208df7e4a3d72a3a3 ./contracts/ceros/interfaces/ISwapRouter.sol
785d84a1bb1482f93bbb504803f7ca0b078c80abd778d5c8bc553b3ab302559f ./contracts/ceros/interfaces/IVault.sol
```

Tests

```
bb8d48ddc82ed196acf00c07062333b8376ace8613c600a48f59f6b4c6eb4e3b ./test/colander.test.js
dc4ff951ab6c284279408759fb652c88ed619af3c700753fc8102b7f729ddb8c ./test/MasterVault/masterVault.test.js
fd7bfbf336abbc9a6f96ffe4d1738e66ef79d0f5aaeccf299ecc6c0f6187b4fc ./test/MasterVault/swapPool.test.js
```

Changelog

- 2022-12-02 - Initial report
- 2023-01-30 - Fix Review

About Quantstamp

Quantstamp is a global leader in blockchain security. Founded in 2017, Quantstamp's mission is to securely onboard the next billion users to Web3 through its best-in-class Web3 security products and services.

Quantstamp's team consists of cybersecurity experts hailing from globally recognized organizations including Microsoft, AWS, BMW, Meta, and the Ethereum Foundation. Quantstamp engineers hold PhDs or advanced computer science degrees, with decades of combined experience in formal verification, static analysis, blockchain audits, penetration testing, and original leading-edge research.

To date, Quantstamp has performed more than 500 audits and secured over $200 billion in digital asset risk from hackers. Quantstamp has worked with a diverse range of customers, including startups, category leaders and financial institutions. Brands that Quantstamp has worked with include Ethereum 2.0, Binance, Visa, PayPal, Polygon, Avalanche, Curve, Solana, Compound, Lido, MakerDAO, Arbitrum, OpenSea and the World Economic Forum.

Quantstamp's collaborations and partnerships showcase our commitment to world-class research, development and security. We're honored to work with some of the top names in the industry and proud to secure the future of web3.

Notable Collaborations & Customers:

- Blockchains: Ethereum 2.0, Near, Flow, Avalanche, Solana, Cardano, Binance Smart Chain, Hedera Hashgraph, Tezos
- DeFi: Curve, Compound, Aave, Maker, Lido, Polygon, Arbitrum, SushiSwap
- NFT: OpenSea, Parallel, Dapper Labs, Decentraland, Sandbox, Axie Infinity, Illuvium, NBA Top Shot, Zora
- Academic institutions: National University of Singapore, MIT