Response to Preliminary Report

We hope that the preliminary report provided understandable and helpful feedback. During the upcoming engagement phase we will review your resulting modifications. To make this process as efficient as possible and to avoid potential miscommunication, we have prepared the following response document.

Using this document you can:

- Provide feedback if you feel that any of our findings were (partially) unjustified.
- Document the modifications you are making along with corresponding commits.
- Highlight findings that you plan not to address.
- Respond to open questions that were mentioned in the report.

In case any parts of the report, this form or the overall process are unclear, please reach out at any time to schedule a call. This document is for internal use only.

Possible status messages

For each issue below, please indicate the overall status of your response using one of the following status messages:

Code Change
The issue was resolved through a code change. Please provide the brief approach and the commit hash.

Specification Change
The issue was resolved through a specification change. Please reference the change.

Process Change
The issue was addressed using a modified business process. Please explain the modification.

Risk accepted
The associated risk was deemed small enough that no modification is necessary. Hence, no change will be made.

No issue
The reported issue was incorrect. Hence, no change will be made.

New system state

Please provide the commit hash that includes all the modifications made and which should be considered as new version: Link.

Please note that all fixes have been pushed to feature/veSDT-new branch and we expect them to be reviewed and fixed here. Once the fixes' review is done, we'll merge feature/veSDT-new branch into feature/step3. Both of these branches will be frozen until review is finished.

Findings

5.1 Incorrect Index Used to Access `depositorsIndex`

- **Status**: Code Change
- **Description of changes**: In the last claimReward contract deployed, the depositorsIndex is accessed using the depositor address instead of the token address.
- **Commit hash (if applicable)**: link

5.2 Possible to Lock Users' Funds Into `veSDT`

- **Status**: Code Change
- **Description of changes**: Since that the veSDT contract is upgradeable, we have already done it, disallowing an address to lock SDT for another one, even if this address has approved the contract with an infinite allowance to transfer SDT to it.
- **Commit hash (if applicable)**: link

5.3 Broad Function Visibility: `approveWallet`

- **Status**: Risk Accepted
- **Description of changes**: The actual SWW deployed on mainnet has the approveWallet function defined as public. The contract used is the same from curve/angle.
- **Commit hash (if applicable)**:

5.4 Inconsistent Checks When Depositing in `veSDT`

- **Status**: Risk Accepted
- **Description of changes**: When an EOA or a contract call the deposit_for, it can be used to deposit for an existing lock, and so a contract can't create a new lock for itself in any case. It should call the create_lock before and instead it should be whitelisted for calling it.
- **Commit hash (if applicable)**:

5.5 Inconsistent Procedure for Updating `admin`

- **Status**: Risk Accepted
- **Description of changes**: We are going to manage them in the best way.

- **Commit hash (if applicable)**:

5.6 Inconsistent Specification: `deposit_for_from`

- **Status**: Code Change
- **Description of changes**: We fixed the natspec specification for the deposit_for_from contract
- **Commit hash (if applicable)**: #1, #2, #3

5.7 Inconsistent Specification: `initialize`

- **Status**: Code Change
- **Description of changes**: We fixed it in the veSDT upgraded implementation contract.
- **Commit hash (if applicable)**: link

5.8 Mismatch of Specification With the Function Modifier in `AngleLocker`

- **Status**: Risk Accepted
- **Description of changes**: The specification comment is wrong because it mentioned a proxy where it is not declared at the end.
- **Commit hash (if applicable)**:

5.9 Missing Documentation for Parameter

- **Status**: Risk Accepted
- **Description of changes**: The internal init function has not description for the admin parameter.
- **Commit hash (if applicable)**:

5.10 Missing Events for Sensitive Operations

- **Status**: Code Change
- **Description of changes**: Added an event when the ClaimRewards set a new governance address, instead for the other contracts, already deployed, we did not add any events.
- **Commit hash (if applicable)**: link

5.11 Missing Sanity Checks

- **Status**: Risk Accepted
- **Description of changes**: Not fixed, we will manage it carefully to not pass a 0 address or a wrong address. Btw this type of setter function won't be managed using an UI.
- **Commit hash (if applicable)**: link

5.12 Missing Sanity Checks: `AngleLocker`

- **Status**: Risk Accepted

- **Description of changes**: Not fixed, we will manage it carefully to not pass a 0 address or a wrong address. Btw this type of setter function won't be managed using an UI.
- **Commit hash (if applicable)**:

5.13 Non-indexed Events

- **Status**: No Issue
- **Description of changes**: We decided to not include indexed parameters within the events definition, cause they will increase the gas a little bit and also we could fetch externally the same info using theGraph.
- **Commit hash (if applicable)**:

5.14 Possible Gas Optimization in Mappings

- **Status**: Code Change
- **Description of changes**: We changed the claimRewards gauges variable, using an address =>uint256 mapping instead of the address=>bool. We did not optimise it for the SWW and the SdtDistributor contract because they had been already deployed.
- **Commit hash (if applicable)**: link

5.15 Possible to Optimize the Check on Distributor of `tokenReward`

- **Status**: Code Change
- **Description of changes**: It has been fixed, it is checking directly if the distributor is equal to the related accumulator (address(this) for this check). We fixed it only for the CrvAccumulator that needs to be deployed.
- **Commit hash (if applicable)**: link

5.16 Unused Events

- **Status**: Code Change
- **Description of changes**: We have deleted the unused events by the ClaimRewards contract, but not from the SdtDitributorEvents and GaugeController contracts, because they were already deployed.
- **Commit hash (if applicable)**: link

Open Questions

7.1 Motivation of `BaseAccumulator.depositToken`

- **Response to open question**: We defined a depositToken function into the BaseAccumulator, so every accumulator can receive any ERC20 via this function. We preferred to define a function that can emit an event to trace the transfer

externally and in an easy way, instead of just transfer the fund to the accumulator contract using the ERC20.transfer from an EOA or another contract. During the next step, when vaults+strategies will be released, this function will be called by the strategies contract, at every harvest, to allocate a percentage to this.

- **Description, in case of resulting changes**:
- **Commit hash (if applicable)**:

Notes

8.1 Admin's Weight on a Gauge Can Be Overwritten

- **Status**: Risk Accepted
- **Description of changes**: The weight, for a gauge already included into the GauceController won't likely change, if it would happen we will take care of managing it.
- **Commit hash (if applicable)**:

8.2 All Gauges Should Be Trusted

- **Status**: Risk Accepted
- **Description of changes**: Since that every gauge will be added by the governance, into the SdtDistributor, so we can be sure that, during the distribution rewards for SDT, it will be split and sent only between trusted gauges (sdToken LGV4).
- **Commit hash (if applicable)**:

8.3 Calculation of Seconds in Years

- **Status**: No Issue
- **Description of changes**: It is because 364 is divisible by zero, and it would be more precise for doing the calculus in 5.2
- **Commit hash (if applicable)**:

8.4 Dust Amounts Not Accepted in veSDT

- **Status**: Risk Accepted
- **Description of changes**: We did not account for the dust amounts related to lockers that are locking a really small amount of SDT i.e. 0.000000000126144000 SDT
- **Commit hash (if applicable)**:

8.5 Event Can be Emitted Multiple Times

- **Status**: No Issue
- **Description of changes**: We know about that, but it can see like a non issue at the end, because these events will be emitted not so often.
- **Commit hash (if applicable)**:

8.6 Hardcoded Function Parameter ``unlock_time``

- **Status**: Code Change
- **Description of changes**: We deleted the internal _deposit_for_from() function, in the last version of the veSDT contract, already upgraded, so it is not an issue anymore. For the other internal function, _deposit_for(), it is not hardcoding the unlock time whenever called.
- **Commit hash (if applicable)**: #1, #2

8.7 Incorrect Checks in Tests

- **Status**: Risk Accepted
- **Description of changes**: We are going to refactor most of the tests, they are updated right now, to support all use cases of the entire architecture, from step1 to the actual step3. So then every action flow related to the platform will be supported.
- **Commit hash (if applicable)**:

8.8 Outdated Compiler Version

- **Status**: Risk Accepted
- **Description of changes**: We used the 0.8.7 in all contracts already deployed, and in the upcoming ones strictly related to this step, like the crv liquid locker depositor. Instead, we are planning to use the most updated version for the contracts related to the next step.
- **Commit hash (if applicable)**:

8.9 Possible Reentrancy in lockToken for Special Tokens

- **Status**: Risk Accepted
- **Description of changes**: We will take care to adapt this control to avoid reentrancy attacks, if it would need to support ERC777 standard token or any standard that would support callbacks as sdToken.
- **Commit hash (if applicable)**:

8.10 Proper Declaration of Constant Variables

- **Status**: Code Change
- **Description of changes**: We defined it as constant in the last ClaimRewards contract, already deployed.
- **Commit hash (if applicable)**: link

8.11 Reward Distribution should be called Periodically for All Gauges

- **Status**: Risk Accepted
- **Description of changes**: For resolving the first issue, we would take care of calling the distributeMulti for every gauge enabled, to avoid that a percentage of SDT allocated for a gauge, for the related period won't be distributed, and we don't have to rescue them. For the latest, we can automate a bot that will trigger the distributeMulti for all gauges just after the new vote weight from gauges can be updated.

- **Commit hash (if applicable)**:

8.12 Tautology in if Condition

- **Status**: Code Change
- **Description of changes**: It has been fixed in the last version of the CrvDepositor (outside of this audit scope for now) contract that needs to be deployed.
- **Commit hash (if applicable)**: link

8.13 Typo in Mappings Specification

- **Status**: Risk Accepted
- **Description of changes**: The typo remains because it has been deployed before this review.
- **Commit hash (if applicable)**:

8.14 ClaimRewards Functions should be called only with enabled gauges

- **Status**: Risk Accepted
- **Description of changes**: We preferred to manage this use case defining a require instead of just skipping the non enabled gauge without reverting the tx. Since a user will likely interact with this contract using the StakeDAO UI, it would be able to pass the correct gauge address and manage only the ones approved. Also Web3 wallet like MM could advise that the tx would revert (thanks to the require defined) if there would be any issue with the addresses passed as input parameters.
- **Commit hash (if applicable)**:

8.15 safeApprove Usage

- **Status**: Code Change
- **Description of changes**: It has been fixed in the CrvDepositor (outside of the audit scope for now) contract. We decided to use the safeIncreaseAllowance instead of the safeApprove.
- **Commit hash (if applicable)**: link