Quantstamp Security Assessment Certificate

January 29th 2021 — Quantstamp Verified

Reflexer Second Engagement (Assorted)

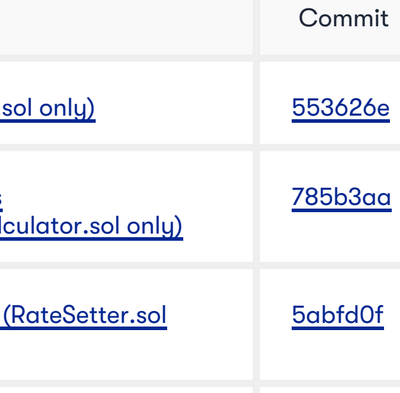
This security assessment was prepared by Quantstamp, the leader in blockchain security

Executive Summary

Type	DeFi
Auditors	Jake Goh Si Yuan, Senior Security Researcher Sebastian Banescu, Senior Research Engineer Fayçal Lalidji, Security Auditor
Timeline	2020-12-07 through 2021-01-25
EVM	Muir Glacier
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	Web Document
Documentation Quality	Medium
Test Quality	Medium
Source Code	

Repository	Commit
ds-token (Delegated.sol only)	553626e
geb-rrfm-calculators (PIRawPerSecondCalculator.sol only)	785b3aa
geb-rrfm-rate-setter (RateSetter.sol only)	5abfd0f
[REMOVED FROM SCOPE] geb-incentives (RollingDistributionIncentives.sol only)	6c6caf5
(Contract GebProxyIncentivesActions within GebProxyActions.sol only)	a6d068f
geb-debt-auction-param-setter (DebtAuctionInitialParameterSetter.sol only)	b8d6f29
ds-pause (protest-pause.sol only)	bab7bd2

Total Issues	**20** (8 Resolved)
High Risk Issues	**1** (1 Resolved)
Medium Risk Issues	**2** (0 Resolved)
Low Risk Issues	**8** (4 Resolved)
Informational Risk Issues	**7** (1 Resolved)
Undetermined Risk Issues	**2** (2 Resolved)



High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or is reasonably likely to lead to moderate financial impact.
Low Risk	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low-impact in view of the client's business circumstances.
Informational	The issue does not post an immediate risk, but is relevant to security best practices or Defence in Depth.
Undetermined	The impact of the issue is uncertain.
Unresolved	Acknowledged the existence of the risk, and decided to accept it without engaging in special efforts to control it.
Acknowledged	The issue remains in the code but is a result of an intentional business or design decision. As such, it is supposed to be addressed outside the programmatic means, such as: 1) comments, documentation, README, FAQ; 2) business processes; 3) analyses showing that the issue shall have no negative consequences in practice (e.g., gas analysis, deployment settings).
Resolved	Adjusted program implementation, requirements or constraints to eliminate the risk.
Mitigated	Implemented actions to minimize the impact or likelihood of the risk.

Summary of Findings

In the second engagement of audit for Reflexer, we have discovered 19 issues of varying levels [High 1, Medium 2, Low 7, Informational 6, Undetermined 3], along with 11 best practices recommendations, that we strongly advise the Reflexer team address through code fix or an official response.
There was a marked drop in the quality level of specification and documentation compared to the last audit, and we encourage the Reflexer team to take some time to boost it up, especially given the relative complexity and potential popularity of the project. The lack of specification and documentation was especially obvious in `PIRawPerSecondCalculator.sol`, which contained complex and proprietary code.
We would also like to highlight the missing input and intermediate result validation, especially present in `GebProxyAction.sol`. Whilst we understand that most of the interactions with external contracts happens within the Reflexer ecosystem, the fact that all of the external contracts were referenced with only an interface and an arbitrary address increases the level of risk, that can be easily and cheaply reduced through checking the return values.
Update:

- The entire `geb-incentives` repo has been removed from scope by the decision of the Reflexer team. We have largely chosen to set issues related to it to status of `Mitigated` or `Removed` where appropriate.
- A separate time-boxed penetration test was undertaken related to the aforementioned opacity in `GebProxyAction.sol` along with a redesigned `geb-incentives` that is discussed in a different report titled "Reflexer Incentives Penetration Test".

ID	Description	Severity	Status
QSP-1	Campaign `modifyParameters` may have unexpected and unaccounted secondary changes	High	Mitigated
QSP-2	Integer Overflow / Underflow	Medium	Acknowledged
QSP-3	Vote delegation mechanism may be flash loan exploitable	Medium	Acknowledged
QSP-4	ERC20 `approve` frontrunning	Low	Acknowledged
QSP-5	Missing input validation	Low	Acknowledged
QSP-6	Gas Usage / `for` Loop Concerns	Low	Mitigated
QSP-7	Possible division by zero	Low	Fixed
QSP-8	Potential misuse of `convertTo18` due to misleading name	Low	Mitigated
QSP-9	Loss of precision due to multiplication after division	Low	Fixed
QSP-10	Lack of assertion and control for calls to external interfaces and arbitrary addresses	Low	Acknowledged
QSP-11	Unlocked and inconsistently based pragma	Informational	Acknowledged
QSP-12	Block Timestamp Manipulation	Informational	Mitigated
QSP-13	Incorrect ERC20 `decimals` type	Informational	Acknowledged
QSP-14	Privileged Roles and Ownership	Informational	Acknowledged
QSP-15	Code duplication and cloning	Informational	Unresolved
QSP-16	Incorrect/Missing Visibility	Informational	Unresolved
QSP-17	Incorrect parameter validation	Undetermined	Mitigated
QSP-18	Possible off-by-one error	Undetermined	Fixed
QSP-19	Missed `historicalCumulativeDeviations` update	Informational	Acknowledged
QSP-20	Loss of precision from `convertTo18`	Low	Unresolved

Quantstamp Audit Breakdown

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

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

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

Methodology

The Quantstamp auditing process follows a routine series of steps:

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

Toolset

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

Setup

Tool Setup:

- Slither v0.6.15

Steps taken to run the tools:

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

Findings

QSP-1 Campaign `modifyParameters` may have unexpected and unaccounted secondary changes

Severity: *High Risk*

Status: Mitigated

File(s) affected: `geb-incentives/GebUniswapRollingDistributionIncentives.sol`

Description: Parameters in a campaign are closely related, and when updating them, other parameters that are dependent on the modified value must also be symmetrically updated, otherwise there may be secondary changes that are unexpected and/or unaccounted for.
For instance, updating `duration` without modifying the `rewardRate` will cause the staking algorithm to distribute an amount different from the expected `reward`. This issue cascades downwards as now the out-of-sync `reward` also means that the contract state variable `globalReward` is affected. Users will receive unexpected amounts.
A similar issue takes place when updating `startTime` or `reward`.

Recommendation: In the Campaign struct, any updates to :

1. `duration` should have a subsequent update to `rewardRate`.
2. `startTime` should have a subsequent update to `finish`.
3. `reward` should have a subsequent update to `rewardRate` and `globalReward`.

Update : The original repo `geb-incentives` is removed from the audit scope.

QSP-2 Integer Overflow / Underflow

Severity: *Medium Risk*

Status: Acknowledged

File(s) affected: `ds-token/delegate.sol`, `geb-proxy-actions/GebProxyActions.sol`, `geb-incentives/GebUniswapRollingDistributionIncentives.sol`

Description: Integer overflow/underflow occur when an integer hits its bit-size limit. Every integer has a set range; when that range is passed, the value loops back around. A clock is a good analogy: at 11:59, the minute hand goes to 0, not 60, because 59 is the largest possible minute. Integer overflow and underflow may cause many unexpected kinds of behavior and was the core reason for the `batchOverflow` attack. Here's an example with `uint8` variables, meaning unsigned integers with a range of `0..255`.

```
function under_over_flow() public {
    uint8 num_players = 0;
    num_players = num_players - 1;  // 0 - 1 now equals 255!
    if (num_players == 255) {
        emit LogUnderflow();        // underflow occurred
    }
    uint8 jackpot = 255;
    jackpot = jackpot + 1;          // 255 + 1 now equals 0!
    if (jackpot == 0) {
        emit LogOverflow();         // overflow occurred
    }
}
```

In particular, the following instances were spotted:

1. [ACKNOWLEDGED] The addition operation on L228 in `ds-token/delegate.sol` uses the primitive operator (+). This could lead to an integer overflow if `nCheckpoints` is equal to the largest integer value representable by the `uint32` type.
2. [ACKNOWLEDGED] The addition operation on L1194 in `geb-proxy-actions/GebProxyActions.sol` uses the primitive operator (+). This could lead to an integer overflow if `deltaDebt` is equal to the largest integer value representable by the `int` type.
3. [FIXED] The multiplication operation on L2597 in `geb-proxy-actions/GebProxyActions.sol` uses the primitive operator (*). This could lead to an integer overflow.
4. [FIXED] The subtraction operation on L2597 in `geb-proxy-actions/GebProxyActions.sol` uses the primitive operator (-). This could lead to an integer underflow if `leverage < 1000`.
5. [REMOVED] The addition operation on L148 in `geb-incentives/GebUniswapRollingDistributionIncentives.sol` uses the primitive operator (+). This could lead to an integer overflow if `val` is high enough.

Recommendation: Use `ds-math` or some form of overflow protection in these arithmetic operations.

Update: From the Reflexer team:

1. "The number of checkpoints is bounded by the amount of delegations to a given account. We feel that the max uint256 value is more than enough to store this, with overflows being very unlikely."
2. "deltaDebt is calculated from the caller's Safe, so it's impossible for it to be the max uint256 value (the collateral debt ceiling, safe debt ceiling and the user's funds to open a Safe will prevent this)."

QSP-3 Vote delegation mechanism may be flash loan exploitable

Severity: *Medium Risk*

Status: Acknowledged

File(s) affected: `ds-token/delegate.sol`

Description: Currently, `DSDelegateToken` employs a system of vote delegation that is triggered whenever there is a token movement of some form, including minting and burning. The contract also has the `getCurrentVotes` method which provides the main intended way for other contracts to view and use the vote count in their operations. The method always retrieves the latest result, even when changes have happened in the same block.
This means that if this token is listed on some kind of on-chain lending provider which has the flash loan feature, a malicious actor would have the option to amplify his voting power at little cost.

Recommendation: It is possible to fix this issue entirely by returning the votes count at the end of the block before any transactions have taken place, given that `getPriorVotes` would already serves the start of the block.
However, if there is a desire to keep it as it is, then it would be absolutely vital to highlight this possibility to any potential user. Due to the fact that this exploit can only be done in combination with another component, it would be important to clearly detail out this risk for any incoming integrations.

Update: From Reflexer team: "This is a known limitation of the delegate token contract. Even though (within the same transaction) the user can have flash loaned delegated power, governaqnce proposals use a snapshot at a specific block (the latest transaction, meaning nothing in between is not taken into consideration)."

QSP-4 ERC20 `approve` frontrunning

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `ds-token/delegate.sol`

Description: In Ethereum the order of transactions is not fixed and the evaluation is not commutative. There are attacks that are possible through manipulating this property. It is termed front-running.
This is a well known issue in the ERC20 standard regarding the `approve` method and it's susceptibility to frontrunning. Essentially, attempts to modify a non-zero allowance to a malicious actor may be exploited by the same actor to control a total allowance that is higher than intended.

Exploit Scenario: Imagine two friends — Alice and Bob.

1. Alice decides to allow Bob to spend some of her funds, for example, 1000 tokens. She calls the approve function with the argument equal to 1000.
2. Alice rethinks her previous decision and now she wants to allow Bob to spend only 300 tokens. She calls the approve function again with the argument value equal to 300.
3. Bob notices the second transaction before it is actually mined. He quickly sends the transaction that calls the transferFrom function and spends 1000 tokens.
4. Since Bob is smart, he sets very high fee for his transaction, so that miner will definitely want to include his transaction in the block. If Bob is as quick as he is generous, his transaction will be executed before the Alice's one.
5. In that case, Bob has already spent 1000 Alice's tokens. The number of Alice's tokens that Bob can transfer is equal to zero.
6. Then the Alice's second transaction is mined. That means, that the Bob's allowance is set to 300.

7. Now Bob can spend 300 more tokens by calling the transferFrom function.

Recommendation: This issue should be made explicitly known to users such that they will be able to conservatively verify the two separate transactions' confirmations before proceeding with the next.
At the same time, allowance changes could be limited to only changes between zero and non-zero values only to mitigate it.

Update: From Reflexer team: "This is a well known issue. We opt for leaving this as is. Requiring zero approvals before lowering the approval will create more friction than desired. `increaseAllowance` and `decreaseAllowance` are rarely used."

QSP-5 Missing input validation

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `ds-token/delegate.sol`, `ds-pause/protest-pause.sol`, `geb-incentives/GebUniswapRollingDistributionIncentives.sol`, `geb-debt-auction-param-setter/DebtAuctionInitialParameterSetter.sol`

Description: Input validation is an easy and low cost way of avoiding painful and damaging simple mistakes arising from human and UI errors. We have identified several areas where it would be useful to have them:

1. [ACKNOWLEDGED] `ds-token/delegate.sol::DSDelegateToken.delegate()` does not check if the `delegatee` address is different from `address(0)`.
2. [ACKNOWLEDGED] `ds-token/delgate.sol::DSDelegateToken.delegateBySig()` does not check if the `delegatee` address is different from `address(0)`.
3. [ACKNOWLEDGED] `ds-token/delgate.sol::DSDelegateToken.mint()` does not check if the `guy` address is different from `address(0)`.
4. [ACKNOWLEDGED] `ds-pause/protest-pause.sol::DSProtestPause.constructor()` is not validating the `protesterLifetime_`, `owner_` and `authority_` input parameters.
5. [FIXED] `DebtAuctionInitialParameterSetter.modifyParameters()` does not check if `val < maxUpdateCallerReward` when `parameter == "baseUpdateCallerReward"`. This allows setting `baseUpdateCallerReward` higher than `maxUpdateCallerReward`.
6. [REMOVED] `GebUniswapRollingDistributionIncentives._updateReward()` does not check if the `campaignId` is valid.

Recommendation: Add the necessary validation.

Update: From the Reflexer team, addressing the `Acknowledged` subpoints: "For the delegate functions, we chose not to have input validation (due to the low impact). For `delegate` and `delegateBySig` the caller can just call the functions once again, seting the parameters to the correct delegatee. Delegating to `address(0)` just means giving voting power to an address that cannot vote/propose. `mint` is also low risk, as governance will initially retain power over it until we test the system, verify that only debt auctions can mint and then abdicate the minter".

QSP-6 Gas Usage / `for` Loop Concerns

Severity: *Low Risk*

Status: Mitigated

File(s) affected: `geb-incentives/GebUniswapRollingDistributionIncentives.sol`

Description: Gas usage is a main concern for smart contract developers and users, since high gas costs may prevent users from wanting to use the smart contract. Even worse, some gas usage issues may prevent the contract from providing services entirely. For example, if a `for` loop requires too much gas to exit, then it may prevent the contract from functioning correctly entirely. It is best to break such loops into individual functions as possible.
The `GebUniswapRollingDistributionIncentives.updateReward` modifier is bounded by the `campaignCount` state variable, which is initialized to 30, but which can be set via the `modifyParameters` function. If this modifier runs out of gas it would not allow functions such as `stake` and `withdraw` to be executed successfully.

Recommendation: Place a hard cap on the maximum number of allowed loop iterations. An accurate cap can be determined by performing a gas analysis.

Update : The original repo `geb-incentives` is removed from the audit scope.

QSP-7 Possible division by zero

Severity: *Low Risk*

Status: Fixed

File(s) affected: `geb-proxy-actions/GebProxyActions.sol`

Description: If the `TaxCollectorLike(taxCollector).taxSingle(collateralType)` function call in the `_getGeneratedDeltaDebt` function would return a value equal to zero, it may lead to a division by zero inside that function on L1192: `deltaDebt = toInt(subtract(multiply(wad, RAY), coin) / rate);`
Likewise, if the `SAFEEngineLike(safeEngine).collateralTypes(collateralType)` function call in the `_getRepaidDeltaDebt` function would return a value equal to zero, it may lead to a division by zero inside that function on L1210: `deltaDebt = toInt(coin / rate);`

Recommendation: Use the division function in `ds-math` and/or validate and ensure that the rate cannot be zero.

QSP-8 Potential misuse of `convertTo18` due to misleading name

Severity: *Low Risk*

Status: Mitigated

File(s) affected: `geb-proxy-actions/GebProxyActions.sol`

Description: The `convertTo18` function only converts tokens that have less than 18 decimal digits. It does not affect tokens with more than 18 decimals in any way. Therefore, if the provided tokens has 20 decimals the function will not truncate the last 2 decimals and will simply return the same value. The caller of the function who might be expecting a value with 18 decimals in this case will misinterpret the return value of this function.

Recommendation: Add a require statement to check if the number of decimals of the provided collateral address is higher than 18.

Update: The solution may truncate for values above 18 decimals and cause a loss of precision.

QSP-9 Loss of precision due to multiplication after division

Severity: *Low Risk*

Status: Fixed

File(s) affected: `geb-debt-auction-param-setter/DebtAuctionInitialParameterSetter.sol`

Description: Integer division in Solidity leads to truncated results. Therefore, division should in general be performed after multiplication to minimize the loss of precision. We have identified the following instances where the ordering should be changed:

1. [REMOVED] On L262 in `GebUniswapRollingDistributionIncentives.sol` division is performed before multiplication.
2. [FIXED] On L275-279 in `DebtAuctionInitialParameterSetter.sol`division is performed on L275, before the multiplication on L279.
3. [FIXED] On L300-303 in `DebtAuctionInitialParameterSetter.sol` division is performed on L300, before the multiplication on L303.
4. [FIXED] On L328-343 in `DebtAuctionInitialParameterSetter.sol` division is performed on L328, before the multiplication on L343.

Recommendation: To preserve precision, ensure that all multiplications are completed before proceeding to division.

QSP-10 Lack of assertion and control for calls to external interfaces and arbitrary addresses

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `geb-proxy-actions/GebProxyActions.sol`

Description: Most functions will return a `true` or `false` value upon success. Some functions, like `send()`, are more crucial to check than others. It's important to ensure that every necessary function is checked. It is especially vital when one is making external calls to interfaces or when making low level calls, which is the case most prevalent in most of the contact `GebIncentiveAction` within `GebProxyActions.sol`.
The contract contains several functions which make receive address parameters as inputs, cast them to certain contract interfaces and make calls to those contracts. For example, the public function `generateDebt`does not do any input validation and directly calls the internal function `_generateDebt`. The internal function `_generateDebt` casts its input parameter address manager to the `ManagerLike` interface and then calls several functions such as: `safes(safe)`, `safeEngine()` and `collateralTypes(safe)`. The return values of these 3 functions is not explicitly checked using assert or require statements inside the `_generateDebt` function, which in some cases might be problematic. Note that this is just one example of many. Since the scope of this audit does not include many of the contracts being casted-to and called, e.g. `GebSafeManager`, `SAFEEngine`, we are not certain that the called functions are guaranteed to always return the range of values expected by the calling functions inside `GebProxyAction.sol`.

Recommendation: It is not expensive to add assert or require statements to validate that the return values of functions from external contracts are in the expected range, and ensure that certain unintended execution paths are eliminated entirely. These validations should extend to all external calls, including ones intended for the transfer of `ETH`. This would be our strong recommendation.

Update: From the Reflexer team: "The functions within our system will revert on failure. The operations in the proxy contract depend on the previous ones success to succeed, meaning if anything fails in between, the target contract reverts."

QSP-11 Unlocked and inconsistently based pragma

Severity: *Informational*

Status: Acknowledged

File(s) affected: `All files`

Description: Every Solidity file specifies in the header a version number of the format `pragma solidity (^)0.*.*`. 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".
It is more concerning here than usual as the instances of unlocked pragma are inconsistently based as well, which increases the complexity of maintenance and therefore the probability of errors and exploits occurring.

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

Update: From the Reflexer team: "We acknowledge that many contracts have unlocked pragmas. We think the impact is low."

QSP-12 Block Timestamp Manipulation

Severity: *Informational*

Status: Mitigated

File(s) affected: `geb-incentives/GebUniswapRollingDistributionIncentives.sol`

Description: Projects may rely on block timestamps for various purposes. However, it's important to realize that miners individually set the timestamp of a block, and attackers may be able to manipulate timestamps for their own purposes. If a smart contract relies on a timestamp, it must take this into account.
We have identified the following areas which relies on timestamps that one should be, and make well, aware of :

1. geb-incentives/GebUniswapRollingDistributionIncentives.sol::L142
2. geb-incentives/GebUniswapRollingDistributionIncentives.sol::L150
3. geb-incentives/GebUniswapRollingDistributionIncentives.sol::L156
4. geb-incentives/GebUniswapRollingDistributionIncentives.sol::L193
5. geb-incentives/GebUniswapRollingDistributionIncentives.sol::L210
6. geb-incentives/GebUniswapRollingDistributionIncentives.sol::L277
7. geb-incentives/GebUniswapRollingDistributionIncentives.sol::L278
8. geb-incentives/GebUniswapRollingDistributionIncentives.sol::L282
9. geb-incentives/GebUniswapRollingDistributionIncentives.sol::L287
10. geb-incentives/GebUniswapRollingDistributionIncentives.sol::L310
11. geb-incentives/GebUniswapRollingDistributionIncentives.sol::L337

Recommendation: Either write tests which show that a change of up to 900 seconds does not affect the end result, or warn users that this is a potential issue in publicly available documentation.

Update : The original repo `geb-incentives` is removed from the audit scope.

QSP-13 Incorrect ERC20 `decimals` type

Severity: *Informational*

Status: Acknowledged

File(s) affected: `ds-token/delegate.sol`

Description: The type used in `DSDelegateToken` for the `decimals` is `uint256`, whereas the ERC20 standard expects it in `uint8`.

Recommendation: Either fix the type or explicit state that this is not intended to be complying with ERC20.

Update: From the Reflexer team: "We are aware of the incorrect type, and confirmed it has no practical impact, even for interfaces expecting uint8."

QSP-14 Privileged Roles and Ownership

Severity: *Informational*

Status: Acknowledged

File(s) affected: `ds-token/delegate.sol`

Description: Smart contracts will often have `owner` variables to designate the person with special privileges to make modifications to the smart contract. In particular, the following instances have been spotted:

1. `transfer`, `transferFrom` and `approve` in `DSDelegateToken` can be stopped.
2. Tokens can be `mint` and `burn` to and from any arbitrary address in `DSDelegateToken`.

Recommendation: This centralization of power needs to be made clear to the users, especially depending on the level of privilege the contract allows to the owner.
Update: From the Reflexer team : "Governance will be retaining power over these authorized roles early on to make sure the system works as intended. As soon as we're able to verify that the system works as intended and only debt auctions can mint tokens, we will abdicate the access rights."

QSP-15 Code duplication and cloning

Severity: *Informational*

Status: Unresolved

File(s) affected: `geb-proxy-actions/GebProxyActions.sol`

Description: Code clones make the code less maintainable and prone to human error, because modifying the functionality in one clone instance will cause the rest of the clones to behave differently, which may not be intentional. The following functions are clone instances located in multiple contracts:

1. [FIXED] `[subtract, toInt, toRad, convertTo18, _getGeneratedDeltaDebt, _getRepaidDeltaDebt, _getRepaidAlDebt, transfer, approveSAFEModification, denySAFEModification, openSAFE, transferSAFEOwnership, transferSAFEOwnershipToProxy, allowSAFE, allowHandler, transferCollateral, transferInternalCoins, modifySAFECollateralization, quitSystem, enterSystem, moveSAFE, freeETH, exitETH and openLockETHAndGenerateDebt]` functions inside the `GebProxyIncentivesActions` contract are cloned in the `GebProxyLeverageActions` and `GebProxyActions` contracts.
2. [FIXED] The `[_generateDebt, _lockETH, _repayDebt, _repayDebtAndFreeETH, ethJoin_join, ethJoin_join, generateDebt]` functions inside the `GebProxyIncentivesActions` contract are cloned in the `GebProxyLeverageActions` contract. The `[lockETH, safeLockETH, repayDebt, safeRepayDebt, repayAllDebt, safeRepayAllDebt, lockETHAndGenerateDebt]` functions inside the `GebProxyLeverageActions` contract are cloned in the `GebProxyActions` contract.
3. [UNRESOLVED] With the exception of one concrete parameter value (i.e. `msg.sender` as the 3rd parameter of the call to `_removeLiquidityUniswap`), the code inside the `exitAndRemoveLiquidity` function is cloned in the `exitRemoveLiquidityRepayDebt` and `exitRemoveLiquidityRepayDebtFreeETH` functions, where `address(this)` is used instead of `msg.sender`.

Recommendation: Use inheritance to avoid function clones. This way the cloned functions can be defined in the super-contract and reused by the sub-contracts. Code re-use may be employed to avoid clones of code snippets across multiple functions. This way the common code can be implemented in an internal function that is called by the other functions that currently contain code clones.

QSP-16 Incorrect/Missing Visibility

Severity: *Informational*

Status: Unresolved

File(s) affected: `All`

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

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

The following instances were spotted in our audit:

1. [UNRESOLVED] ds-pause/protest-pause.sol::setOwner should be set to external.
2. [UNRESOLVED] ds-pause/protest-pause.sol::setAuthority should be set to external.
3. [FIXED] ds-pause/protest-pause.sol::setProtester should be set to external.
4. [FIXED] ds-pause/protest-pause.sol::setDelay should be set to external.
5. [FIXED] ds-pause/protest-pause.sol::setDelayMultiplier should be set to external.
6. [FIXED] ds-pause/protest-pause.sol::scheduleTransaction(address,bytes32,bytes,uint256) should be set to external.
7. [FIXED] ds-pause/protest-pause.sol::scheduleTransaction(address,bytes32,bytes,uint256,string) should be set to external.

8. [UNRESOLVED] ds-pause/protest-pause.sol::attachTransactionDescription should be set to external.
9. [FIXED] ds-pause/protest-pause.sol::protestAgainstTransaction should be set to external.
10. [FIXED] ds-pause/protest-pause.sol::abandonTransaction should be set to external.
11. [FIXED] ds-pause/protest-pause.sol::executeTransaction should be set to external.
12. [FIXED] ds-pause/protest-pause.sol::getTransactionDelays should be set to external.
13. [UNRESOLVED] ds-token/delegate.sol::decimals should be set to constant.
14. [UNRESOLVED] ds-token/delegate.sol::approve should be set to external.
15. [UNRESOLVED] ds-token/delegate.sol::push should be set to external.
16. [UNRESOLVED] ds-token/delegate.sol::pull should be set to external.
17. [UNRESOLVED] ds-token/delegate.sol::move should be set to external.
18. [UNRESOLVED] ds-token/delegate.sol::mint(uint256) should be set to external.
19. [UNRESOLVED] ds-token/delegate.sol::burn(uint256) should be set to external.
20. [UNRESOLVED] ds-token/delegate.sol::mint(address,uint256) should be set to external.
21. [UNRESOLVED] ds-token/delegate.sol::burn(address,uint256) should be set to external.
22. [UNRESOLVED] ds-token/delegate.sol::delegate should be set to external.
23. [UNRESOLVED] ds-token/delegate.sol::delegateBySig should be set to external.
24. [UNRESOLVED] ds-token/delegate.sol::getPriorVotes should be set to external.
25. [FALSE POSITIVE] geb-debt-auction-param-setter/DebtAuctionInitialParameterSetter.sol::lastUpdateTime should be set to external.
26. [UNRESOLVED] geb-debt-auction-param-setter/DebtAuctionInitialParameterSetter.sol::getCallerReward should be set to external.
27. [FALSE POSITIVE] geb-debt-auction-param-setter/DebtAuctionInitialParameterSetter.sol:: should be set to external.
28. [FIXED] geb-debt-auction-param-setter/DebtAuctionInitialParameterSetter.sol::getNewDebtBid should be set to external.
29. [FIXED] geb-debt-auction-param-setter/DebtAuctionInitialParameterSetter.sol::getRawProtocolTokenAmount should be set to external.
30. [FIXED] geb-debt-auction-param-setter/DebtAuctionInitialParameterSetter.sol::getPremiumAdjustedProtocolTokenAmount should be set to external.
31. [FIXED] geb-proxy-actions/GebProxyActions.sol::transfer|L1359 should be set to external.
32. [FIXED] geb-proxy-actions/GebProxyActions.sol::ethJoin_join|L1363 should be set to external.
33. [FIXED] geb-proxy-actions/GebProxyActions.sol::approveSAFEModification|L1376 should be set to external.
34. [FIXED] geb-proxy-actions/GebProxyActions.sol::denySAFEModification|L1383 should be set to external.
35. [FIXED] geb-proxy-actions/GebProxyActions.sol::transferSAFEOwnershipToProxy|L1406 should be set to external.
36. [FIXED] geb-proxy-actions/GebProxyActions.sol::allowSAFE|L1429 should be set to external.
37. [FIXED] geb-proxy-actions/GebProxyActions.sol::allowHandler|L1438 should be set to external.
38. [FIXED] geb-proxy-actions/GebProxyActions.sol::quitSystem|L1473 should be set to external.
39. [FIXED] geb-proxy-actions/GebProxyActions.sol::enterSystem|L1481 should be set to external.
40. [FIXED] geb-proxy-actions/GebProxyActions.sol::moveSAFE|L1489 should be set to external.
41. [UNRESOLVED] geb-proxy-actions/GebProxyActions.sol::lockETH|L1497 should be set to external.
42. [UNRESOLVED] geb-proxy-actions/GebProxyActions.sol::freeETH|L1505 should be set to external.
43. [FIXED] geb-proxy-actions/GebProxyActions.sol::exitETH|L1523 should be set to external.
44. [UNRESOLVED] geb-proxy-actions/GebProxyActions.sol::generateDebt|L1539 should be set to external.
45. [UNRESOLVED] geb-proxy-actions/GebProxyActions.sol::repayDebt|L1549 should be set to external.
46. [FIXED] geb-proxy-actions/GebProxyActions.sol::openLockETHAndGenerateDebt|L1570 should be set to external.
47. [FIXED] geb-proxy-actions/GebProxyActions.sol::repayDebtAndFreeETH|L1582 should be set to external.
48. [FIXED] geb-proxy-actions/GebProxyActions.sol::openLockETHGenerateDebtProvideLiquidityUniswap|L1595 should be set to external.
49. [FIXED] geb-proxy-actions/GebProxyActions.sol::lockETHGenerateDebtProvideLiquidityUniswap|L1619 should be set to external.
50. [FIXED] geb-proxy-actions/GebProxyActions.sol::openLockETHGenerateDebtProvideLiquidityStake|L1643 should be set to external.
51. [UNRESOLVED] geb-proxy-actions/GebProxyActions.sol::lockETHGenerateDebtProvideLiquidityStake|L1670 should be set to external.
52. [FIXED] geb-proxy-actions/GebProxyActions.sol::provideLiquidityUniswap|L1697 should be set to external.
53. [FIXED] geb-proxy-actions/GebProxyActions.sol::generateDebtAndProvideLiquidityUniswap|L1707 should be set to external.
54. [FIXED] geb-proxy-actions/GebProxyActions.sol::stakeInMine|L1726 should be set to external.
55. [FIXED] geb-proxy-actions/GebProxyActions.sol::generateDebtAndProvideLiquidityStake|L1731 should be set to external.
56. [REMOVED] geb-proxy-actions/GebProxyActions.sol::harvestReward|L1753 should be set to external.
57. [REMOVED] geb-proxy-actions/GebProxyActions.sol::getLockedReward|L1760 should be set to external.
58. [FIXED] geb-proxy-actions/GebProxyActions.sol::exitMine|L1767 should be set to external.
59. [FIXED] geb-proxy-actions/GebProxyActions.sol::withdrawFromMine|L1776 should be set to external.
60. [FIXED] geb-proxy-actions/GebProxyActions.sol::withdrawAndHarvest|L1783 should be set to external.
61. [FIXED] geb-proxy-actions/GebProxyActions.sol::removeLiquidityUniswap|L1793 should be set to external.
62. [FIXED] geb-proxy-actions/GebProxyActions.sol::withdrawAndRemoveLiquidity|L1798 should be set to external.
63. [FIXED] geb-proxy-actions/GebProxyActions.sol::withdrawRemoveLiquidityRepayDebt|L1805 should be set to external.
64. [REMOVED] geb-proxy-actions/GebProxyActions.sol:: withdrawRemoveLiquidityRepayDebtFreeETH|L1818 should be set to external.
65. [FIXED] geb-proxy-actions/GebProxyActions.sol::exitAndRemoveLiquidity|L1831 should be set to external.
66. [FIXED] geb-proxy-actions/GebProxyActions.sol::exitRemoveLiquidityRepayDebt|L1840 should be set to external.
67. [REMOVED] geb-proxy-actions/GebProxyActions.sol::exitRemoveLiquidityRepayDebtFreeETH|L1854 should be set to external.

68. [REMOVED] geb-incentives/uniswap/GebUniswapRollingDistributionIncentives.sol::stake should be set to external.
69. [REMOVED] geb-incentives/uniswap/GebUniswapRollingDistributionIncentives.sol::campaignListLength should be set to external.
70. [REMOVED] geb-incentives/uniswap/GebUniswapRollingDistributionIncentives.sol::val should be set to external.
71. [FIXED] geb-rrfm-calculators/PIRawPerSecondCalculator.sol::getLastIntegralTerm should be set to external.
72. [FIXED] geb-rrfm-rate-setter/RateSetter.sol::getCallerReward should be set to external.
73. [FIXED] geb-rrfm-rate-setter/RateSetter.sol::updateRate should be set to external.
74. [FIXED] geb-rrfm-rate-setter/RateSetter.sol::getRedemptionAndMarketPrices should be set to external.

Recommendation: Check if the recommendation in the list is apt, and correct if so.

QSP-17 Incorrect parameter validation

Severity: *Undetermined*

Status: Mitigated

File(s) affected: `ds-pause/protest-pause.sol`

Description: The `DSProtestPause.constructor` is validating the `protestEnd` state variable, which is not an input parameter to the constructor.

Recommendation: Clarify if this is intended. Otherwise, validate the right input parameter, namely `protesterLifetime_`.

Update: This issue is considered `Mitigated` only as although the incorrect parameter is removed from validation, `protestorLifetime_` does not seem to be validated in its stead.

QSP-18 Possible off-by-one error

Severity: *Undetermined*

Status: Fixed

File(s) affected: `geb-debt-auction-param-setter/src/DebtAuctionInitialParameterSetter.sol`

Description: Whenever a maximum or cap value is specified for a given parameter, the general expectation is that the maximum can be reached. The require statement on `DebtAuctionInitialParameterSetter.sol::L90` prevents the `baseUpdateCallerReward_` value from being equal to the `maxUpdateCallerReward_` value.

Recommendation: Clarify that it is intended via code comments and user facing documentation, or, replace the strict inequality with a “or-equal” inequality.

QSP-19 Missed `historicalCumulativeDeviations` update

Severity: *Informational*

Status: Acknowledged

File(s) affected: `geb-rrfm-calculators/calculator/PIRawPerSecondCalculator.sol`

Description: In `PIRawPerSecondCalculator.modifyParameters` the modified value of `priceDeviationCumulative` is not pushed to the `historicalCumulativeDeviations`. The expected behavior was that the value would be appended, as the precedent was set in the `constructor`.

Recommendation: Explicitly document this if it is intended behaviour, otherwise rectify it to the expected.

Update: From the Reflexer team: "This is as intended. `modifyParameters` is not expected to affect historical cumulative deviations." We have also switched the severity level to `Informational` based on this new information.

QSP-20 Loss of precision from `convertTo18`

Severity: *Low Risk*

Status: Unresolved

File(s) affected: `geb-proxy-actions/GebProxyActions.sol`

Description: Currently, the main implementation is of the form `amt / 10 ** (decimals - 18)` which truncates useful information from the `amt` and causes a loss of precision. Created from QSP-8 fix.

Recommendation: Address the potential loss of precision through public documentation and/or enforce lossless conversions.

Automated Analyses

Slither

We have analyzed and combed through all 1022 results returned and found that almost all were false-positives. The other true-positive results were no higher than `INFO` level, and we have subsequently embedded them either into their own `INFO` level issue such as QSP-16, or under best practices.

Adherence to Specification

1. It is not clear what the value of the `wad` input parameter of the `_getGeneratedDeltaDebt` function in `GebProxyActions.sol` represents. Also the unit of the return value of this function is not clear.

Code Documentation

Given that this is the second Reflexer audit we have performed, we have an understanding and a gauge of the level of documentation that we can come to expect with the Reflexer team. In this audit, there has been a marked drop in the documentation level for the following reasons:

1. There is no documentation on `https://docs.reflexer.finance` regarding most of the contracts here.
2. There is no README on most of the repos that involve a non-trivial amount of complexity.
3. Complex and important logic such as the `PIRawPerSecondCalculator` has no specification or definition, and it is extremely difficult to ensure that the behavior observed is expected.
4. Inline documentation is missing frequently, functions missing code comments at the beginning of the function, which describe the purpose, input params and outputs, some examples(not exhaustive as it would be too long to list) :
    . `DSDelegateToken._moveDelegates()`
    . `DSDelegateToken._writeCheckpoint()`
    . `DSTokenFactory.make()`
    . `DSDelegateTokenFactory.make()`
    . All functions in `DSTokenBase`
    . All functions in `DSToken`

Adherence to Best Practices

1. The following local variables were identified as unused and can and should be safely removed:
    . geb-proxy-actions/GebProxyActions.sol::L1800 `lpToken`
    . geb-proxy-actions/GebProxyActions.sol::L1808 `lpToken`
    . geb-proxy-actions/GebProxyActions.sol::L1821 `lpToken`

2. Even though we have not identified an over-/under-flow in functions such as `DSDelegateToken.getPriorVotes()`, it is recommended that `ds-math` be used as a defense in depth measure against any future changes that could lead to such issues.
3. The name of the following function may be misleading: `DSDelegateToken.getPriorVotes()` actually returns the number of votes at the given block number. A more suggestive name would be `getVotesAtBlockNumber()`.
4. It is recommended that the number of bits always be specified after the `uint` and `int` types, i.e. `uint256`, `int8`, etc. We have identified several files/contracts where this is not the case, e.g.:
    . `GebProxyActions.sol`
    . `protest-pause.sol`
    . `PIRawPerSecondCalculator.sol`
    . `DebtAuctionInitialParameterSetter.sol`
    . `RateSetter.sol`
    . `GebUniswapRollingDistributionIncentives.sol`

5. There is a typo in the name of the following function `_getRepaidAlDebt -> _getRepaidAllDebt` for all the 3 declarations inside the `GebProxyAction.sol` file.
6. The use of magic numbers should be avoided. Such numbers should be replaced by named constants. Identified instances of magic numbers:
    . L2597 in `GebProxyActions.sol` where the magic number `1000` is used.
    . L163 in `protest-pause.sol` where the magic number `1000` is used.
    . L262 in `GebUniswapRollingDistributionIncentives.sol` where the magic number `100` is used twice.
    . L275, L298 in `PIRawPerSecondCalculator.sol` where the magic number `10**9` is used.
    . L2116 in `GebProxyActions.sol` where the magic numbers `1000`, `997` and `1` are used. It is unclear what these numbers represent.

7. Named constants should indicate that semantics of the constant, not its value. The semantics of the constant should indicate what the meaning of the value is. For example, the `uint256 constant public THOUSAND = 1000` is defined in `GebUniswapRollingDistributionIncentives`. However, this name does not confer any semantics for what the value means, which in this contract we assume to be the value of 1% with 3 decimal digits. Therefore, a more suggestive name for this constant would have been `ONE_PERCENT`. A similar named constant is also defined in the `DebtAuctionInitialParameterSetter` contract.
8. Error messages should be descriptive enough for the end-user to be able to infer the root cause of a tx failure. For some error messages this is the case, however, there are some other error messages which may be harder to decipher. Some examples include: `“invalid-time-elapsed”`
9. Event parameters or type address should be indexed to facilitate filtering. The following instances of unindexed event parameters were identified:
    . The `account` parameter of the `DelayReward` event
    . The `caller` parameter of the `WithdrewExtraRewardTokens` event
    . The `addr` parameter of the `DebtAuctionInitialParameterSetter.ModifyParameter` event
    . The `account` parameter of the `DebtAuctionInitialParameterSetter.AddAuthorization` event
    . The `account` parameter of the `DebtAuctionInitialParameterSetter.RemoveAuthorization` event
    . The `feeReceiver` of the `DebtAuctionInitialParameterSetter.RewardCaller` event

. The `finalFeeReceiver` of the `DebtAuctionInitialParameterSetter.FailRewardCaller` event

. All events in `DSProtestPause`

10. Using `delegates[address(0)]` in `DSDelegateToken.burn` and `DSDelegateToken.mint` is unnecessary and can be replaced by `address(0)`.
11. Unnecessary `require(proxy.owner() == address(this), "ds-protest-pause-illegal-storage-change")` in `DSProtestPause.executeTransaction` since `DSPauseProxy.executeTransaction` is a restricted function and will throw if the caller address is different than `DSProtestPause` contract.

Test Results

Test Suite Results

Almost all of the test suites were able to run besides `GebProxyActions` which output the following error message:

```
fatal: cannot change to 'lib/geb-safe-manager/lib/geb-deploy/lib/ds-auth/src': No such file or directory
child_process.js:650
    throw err;
    ^
Error: Command failed: git -C lib/geb-safe-manager/lib/geb-deploy/lib/ds-auth/src rev-parse HEAD
fatal: cannot change to 'lib/geb-safe-manager/lib/geb-deploy/lib/ds-auth/src': No such file or directory
    at checkExecSyncError (child_process.js:629:11)
    at Object.execFileSync (child_process.js:647:13)
    at run (/nix/store/3ij6qraxzaa0xyxx874g17blq2x09mgl-dapp-0.31.1/libexec/dapp/dapp-remappings:64:35)
    at ls.forEach.name (/nix/store/3ij6qraxzaa0xyxx874g17blq2x09mgl-dapp-0.31.1/libexec/dapp/dapp-remappings:35:16)
    at Array.forEach (<anonymous>)
    at findRemappings (/nix/store/3ij6qraxzaa0xyxx874g17blq2x09mgl-dapp-0.31.1/libexec/dapp/dapp-remappings:18:43)
    at ls.forEach.name (/nix/store/3ij6qraxzaa0xyxx874g17blq2x09mgl-dapp-0.31.1/libexec/dapp/dapp-remappings:51:5)
    at Array.forEach (<anonymous>)
    at findRemappings (/nix/store/3ij6qraxzaa0xyxx874g17blq2x09mgl-dapp-0.31.1/libexec/dapp/dapp-remappings:18:43)
    at ls.forEach.name (/nix/store/3ij6qraxzaa0xyxx874g17blq2x09mgl-dapp-0.31.1/libexec/dapp/dapp-remappings:51:5)
```

The other tests ran successfully:

```
Running 10 tests for src/test/RateSetter.t.sol:RateSetterTest
[PASS] test_get_redemption_and_market_prices() (gas: 43192)
[PASS] test_first_update_rate_with_warp() (gas: 166720)
[PASS] test_null_rate_needed_submit_different() (gas: 250402)
[PASS] test_correct_setup() (gas: 16994)
[PASS] test_oracle_relayer_bounded_rate() (gas: 255611)
[PASS] test_modify_parameters() (gas: 594177)
[PASS] test_two_updates() (gas: 252753)
[PASS] test_disable() (gas: 8006)
[PASS] test_first_update_rate_no_warp() (gas: 164752)
[PASS] testFail_update_before_period_passed() (gas: 162102)

Running 4 tests for src/test/pause-integration.t.sol:Integration
[PASS] test_voteQuorum_direct_integration() (gas: 7990119)
[PASS] test_voteQuorum_dsRecursiveRoles_integration() (gas: 9343739)
[PASS] test_governance_transition() (gas: 15761531)
[PASS] test_multisig_dsRecursiveRoles_integration() (gas: 11788268)

Running 5 tests for src/test/pause-integration.t.sol:IntegrationVotingScenarios
[PASS] test_flash_proposal() (gas: 729071)
[PASS] test_smallVote() (gas: 445426)
[PASS] test_abandonTransaction() (gas: 648079)
[PASS] test_pass_proposal_without_majority() (gas: 867945)
[PASS] test_pass_older_proposal() (gas: 892200)

Running 1 tests for src/test/pause-integration.t.sol:UpgradeVoteQuorum
[PASS] test_quorum_upgrade() (gas: 13566053)

Running 3 tests for src/test/pause.t.sol:Abandon
[PASS] testFail_abandon_unscheduled_transaction() (gas: 17396)
[PASS] test_abandon_scheduled_tx() (gas: 111878)
[PASS] testFail_call_from_unauthorized() (gas: 99743)

Running 6 tests for src/test/pause.t.sol:Admin
[PASS] testFail_cannot_set_owner_without_delay() (gas: 3877)
[PASS] test_set_owner_with_delay() (gas: 342458)
[PASS] testFail_cannot_set_authority_without_delay() (gas: 130872)
[PASS] test_set_authority_with_delay() (gas: 450120)
[PASS] test_set_delay_with_delay() (gas: 321414)
[PASS] testFail_cannot_set_delay_without_delay() (gas: 3901)

Running 3 tests for src/test/pause.t.sol:Constructor
[PASS] test_delay_set() (gas: 2710899)
[PASS] test_owner_set() (gas: 2711003)
[PASS] test_authority_set() (gas: 2711015)

Running 8 tests for src/test/pause.t.sol:Execute
[PASS] testFail_double_execution() (gas: 118554)
[PASS] testFail_codeHash_mismatch() (gas: 92502)
[PASS] testFail_delay_not_passed() (gas: 91411)
[PASS] test_succeeds_when_called_from_unauthorized() (gas: 117863)
[PASS] testFail_exec_plan_with_proxy_ownership_change() (gas: 115673)
[PASS] test_succeeds_when_called_from_authorized() (gas: 113467)
[PASS] test_succeeds_when_delay_passed() (gas: 113489)
[PASS] testFail_execution_too_late() (gas: 96091)

Running 8 tests for src/test/pause.t.sol:Schedule
[PASS] test_schedule_populates_scheduled_transactions_mapping() (gas: 90592)
[PASS] test_schedule_duplicate_transaction_after_execution() (gas: 205614)
[PASS] testFail_schedule_eta_above_max_delay() (gas: 15801)
[PASS] test_schedule_duplicate_transaction_after_abandon() (gas: 199798)
[PASS] testFail_schedule_eta_too_soon() (gas: 13846)
[PASS] testFail_call_from_unauthorized() (gas: 18032)
[PASS] testFail_schedule_duplicate_transaction() (gas: 100562)
[PASS] testFail_schedule_above_max_scheduled_txs() (gas: 691260)

Running 4 tests for src/test/protest-pause-integration.t.sol:Integration
[PASS] test_voteQuorum_direct_integration() (gas: 8959327)
[PASS] test_voteQuorum_dsRecursiveRoles_integration() (gas: 10312947)
[PASS] test_governance_transition() (gas: 16799148)
[PASS] test_multisig_dsRecursiveRoles_integration() (gas: 12757255)

Running 5 tests for src/test/protest-pause-integration.t.sol:IntegrationVotingScenarios
[PASS] test_flash_proposal() (gas: 731849)
[PASS] test_smallVote() (gas: 448204)
[PASS] test_abandonTransaction() (gas: 678755)
[PASS] test_pass_proposal_without_majority() (gas: 870723)
[PASS] test_pass_older_proposal() (gas: 916956)

Running 1 tests for src/test/protest-pause-integration.t.sol:UpgradeVoteQuorum
[PASS] test_quorum_upgrade() (gas: 14557308)

Running 3 tests for src/test/protest-pause.t.sol:Abandon
[PASS] test_drop_scheduled_transaction() (gas: 132265)
[PASS] testFail_abandon_unscheduled_transaction() (gas: 17977)
[PASS] testFail_call_from_unauthorized() (gas: 119153)

Running 12 tests for src/test/protest-pause.t.sol:Admin
[PASS] testFail_cannot_set_owner_without_delay() (gas: 3945)
[PASS] test_set_protester_with_delay() (gas: 440283)
[PASS] testFail_set_delay_multiplier_above_max() (gas: 426963)
[PASS] test_set_owner_with_delay() (gas: 457823)
[PASS] testFail_set_delay_above_max() (gas: 426899)
[PASS] testFail_cannot_set_authority_without_delay() (gas: 130938)
[PASS] test_set_authority_with_delay() (gas: 565573)
[PASS] testFail_cannot_set_delay_multiplier_without_delay() (gas: 3859)
[PASS] test_set_delay_with_delay() (gas: 436801)
[PASS] test_set_delay_multiplier_with_delay() (gas: 436887)
[PASS] testFail_cannot_set_protester_without_delay() (gas: 4712)
[PASS] testFail_cannot_set_delay_without_delay() (gas: 3857)

Running 3 tests for src/test/protest-pause.t.sol:Constructor
[PASS] test_delay_set() (gas: 3677374)
[PASS] test_owner_set() (gas: 3677456)
[PASS] test_authority_set() (gas: 3677468)

Running 8 tests for src/test/protest-pause.t.sol:Execute
[PASS] testFail_double_execution() (gas: 143441)
[PASS] testFail_codeHash_mismatch() (gas: 114449)
```

```
[PASS] testFail_delay_not_passed() (gas: 113358)
[PASS] test_succeeds_when_called_from_unauthorized() (gas: 140177)
[PASS] test_succeeds_when_called_from_authorized() (gas: 135759)
[PASS] test_succeeds_when_delay_passed() (gas: 135781)
[PASS] testFail_execution_too_late() (gas: 118105)
[PASS] testFail_exec_transaction_with_proxy_ownership_change() (gas: 137912)

Running 6 tests for src/test/protest-pause.t.sol:Protest
[PASS] testFail_protest_after_protesterLifetime() (gas: 124486)
[PASS] test_protest_scheduled_tx_max_delay_bound() (gas: 590703)
[PASS] test_protest_scheduled_tx() (gas: 246600)
[PASS] testFail_call_from_unauthorized() (gas: 114622)
[PASS] testFail_protest_scheduled_tx_twice() (gas: 161438)
[PASS] testFail_protest_after_protestEnd() (gas: 146372)

Running 8 tests for src/test/protest-pause.t.sol:Schedule
[PASS] test_schedule_populates_scheduled_transactions_mapping() (gas: 109957)
[PASS] test_schedule_duplicate_transaction_after_execution() (gas: 247294)
[PASS] testFail_schedule_eta_above_max_delay() (gas: 14168)
[PASS] test_schedule_duplicate_transaction_after_abandon() (gas: 239528)
[PASS] testFail_schedule_eta_too_soon() (gas: 12235)
[PASS] testFail_call_from_unauthorized() (gas: 18011)
[PASS] testFail_schedule_duplicate_transaction() (gas: 118132)
[PASS] testFail_schedule_above_max_scheduled_txs() (gas: 885205)



Running 19 tests for src/test/DebtAuctionInitialParameterSetter.t.sol:DebtAuctionInitialParameterSetterTest
[PASS] test_getRawProtocolTokenAmount() (gas: 9322)
[PASS] testFail_set_params_invalid_sys_coin_price() (gas: 40537)
[PASS] test_set_params_get_self_rewarded() (gas: 147590)
[PASS] test_set_params_in_accounting_engine() (gas: 135948)
[PASS] testFail_getNewDebtBid_invalid_price() (gas: 11683)
[PASS] testFail_getNewDebtBid_null_price() (gas: 10746)
[PASS] test_setup() (gas: 52618)
[PASS] testFail_set_params_null_sys_coin_price() (gas: 39682)
[PASS] test_set_params_get_other_rewarded() (gas: 147624)
[PASS] testFail_getRawProtocolTokenAmount_invalid_price() (gas: 11838)
[PASS] test_set_params_after_delay() (gas: 194892)
[PASS] testFail_getRawProtocolTokenAmount_null_price() (gas: 10791)
[PASS] test_getNewDebtBid() (gas: 8660)
[PASS] testFail_set_params_null_prot_price() (gas: 39704)
[PASS] testFail_set_params_invalid_prot_price() (gas: 40491)
[PASS] testFail_getPremiumAdjustedProtocolTokenAmount_null_price() (gas: 10812)
[PASS] testFail_getPremiumAdjustedProtocolTokenAmount_invalid_price() (gas: 11793)
[PASS] test_getPremiumAdjustedProtocolTokenAmount() (gas: 10448)
[PASS] testFail_set_params_before_period_elapsed() (gas: 134389)

Running 50 tests for src/test/GebUniswapRollingDistributionIncentives.t.sol:GebUniswapRollingDistributionIncentivesTest
[PASS] testGetLockedReward() (gas: 1469145)
[PASS] testExitAfterVesting() (gas: 748375)
[PASS] testFailModifyParametersStartPast() (gas: 314796)
[PASS] testGetReward2() (gas: 33799532)
[PASS] testRewardCalculation0() (gas: 566023)
[PASS] testRewardCalculation1() (gas: 790414)
[PASS] testFailNewCampaignOverlappingCampaigns() (gas: 321402)
[PASS] testFailNewCampaignStartPast() (gas: 5809)
[PASS] testRewardCalculation2() (gas: 1323773)
[PASS] testExitMidVesting() (gas: 748161)
[PASS] testStake() (gas: 135601)
[PASS] testRewardCalculation5() (gas: 583796)
[PASS] testFailNewCampaignRewardZero() (gas: 5810)
[PASS] testExit() (gas: 713340)
[PASS] testUpdateRewardBounds() (gas: 64112541)
[PASS] testNewCampaign() (gas: 347095)
[PASS] testFailModifyParametersLongerRewardDelayAfterCampaignStarted() (gas: 316580)
[PASS] testFailModifyParametersUnauthorized() (gas: 6879)
[PASS] testFailNewCampaignInvalidInstantExitPercenntage() (gas: 5067)
[PASS] testGetReward() (gas: 1470447)
[PASS] testFailWithdrawUnauthorized() (gas: 317752)
[PASS] testFailWithdrawNoExtraBalance() (gas: 369636)
[PASS] testFailModifyParametersUnexistent() (gas: 6834)
[PASS] testFailModifyParametersInvalidExitPercentage() (gas: 314882)
[PASS] testFailInvalidTimeElapsed() (gas: 679836)
[PASS] testModifyParameters() (gas: 13386)
[PASS] testFailGetLockedRewardNoBalance() (gas: 711748)
[PASS] testRewardCalculation4() (gas: 1403743)
[PASS] testFailWithdrawMoreThanBalance() (gas: 126378)
[PASS] testStakeFor() (gas: 134891)
[PASS] testFailModifyParametersCampaignStarted() (gas: 315791)
[PASS] testRollingMaxCampaigns() (gas: 20710006)
[PASS] testFailModifyParametersInexistentCampaign() (gas: 314745)
[PASS] testConstructor() (gas: 9200)
[PASS] testFailModifyParametersZeroReward() (gas: 314770)
[PASS] testFailWithdrawZero() (gas: 125142)
[PASS] testFailStakeLowerApproval() (gas: 86943)
[PASS] testFailStakeZero() (gas: 37967)
[PASS] testWithdraw() (gas: 161072)
[PASS] testFailModifyCampaignParametersUnauthorized() (gas: 316662)
[PASS] testModifyReduceMaxCampaigns() (gas: 10524164)
[PASS] testRewardCalculation3() (gas: 919837)
[PASS] testFailNewCampaignDurationZero() (gas: 5875)
[PASS] testModifyCampaignParameters() (gas: 351373)
[PASS] testFailNotifyRewardAmountUnauthorized() (gas: 6908)
[PASS] testFailModifyParametersDurationZero() (gas: 314822)
[PASS] testCancelCampaign() (gas: 674392)
[PASS] testWithdrawExtraRewardTokens() (gas: 365209)
[PASS] testFailCancelledCampaignGetReward() (gas: 472806)
[PASS] testFailCancelCampaignAlreadyStarted() (gas: 317203)

Running 35 tests for src/test/GebUniswapSingleDistributionIncentives.t.sol:GebUniswapSingleDistributionIncentivesTest
[PASS] testGetLockedReward() (gas: 1238079)
[PASS] testRewardCalculation0() (gas: 312543)
[PASS] testRewardCalculation1() (gas: 427009)
[PASS] testFailNotifyRewardTooHigh() (gas: 99312)
[PASS] testFailModifyParametersDurationAfterSetUp() (gas: 103860)
[PASS] testRewardCalculation2() (gas: 427008)
[PASS] testStake() (gas: 147520)
[PASS] testRewardCalculation5() (gas: 323726)
[PASS] testExit() (gas: 450346)
[PASS] testFailInvalidSlot() (gas: 415118)
[PASS] testFailModifyParametersInvalidStartTime() (gas: 4923)
[PASS] testFailModifyParametersUnauthorized() (gas: 7842)
[PASS] testGetReward() (gas: 1156069)
[PASS] testFailWithdrawUnauthorized() (gas: 105434)
[PASS] testFailNotifyRewardAmountAfterPeriod() (gas: 110660)
[PASS] testFailModifyParametersStartTimeAfterSetUp() (gas: 103886)
[PASS] testFailWithdrawNoExtraBalance() (gas: 157340)
[PASS] testFailInvalidTimeElapsed() (gas: 469366)
[PASS] testModifyParameters() (gas: 38897)
[PASS] testFailGetLockedRewardNoBalance() (gas: 468863)
[PASS] testRewardCalculation4() (gas: 687040)
[PASS] testFailWithdrawMoreThanBalance() (gas: 157518)
[PASS] testStakeFor() (gas: 146699)
[PASS] testFailModifyParametersAfterStart() (gas: 4878)
[PASS] testConstructor() (gas: 26039)
[PASS] testFailWithdrawZero() (gas: 156282)
[PASS] testFailStakeLowerApproval() (gas: 98774)
[PASS] testFailStakeZero() (gas: 49885)
[PASS] testFailModifyParametersInvalidInstantExitPercentage() (gas: 4895)
[PASS] testWithdraw() (gas: 192212)
[PASS] testRewardCalculation3() (gas: 410477)
[PASS] testFailStakeBeforeStart() (gas: 49201)
[PASS] testFailNotifyRewardAmountUnauthorized() (gas: 6625)
[PASS] testWithdrawExtraRewardTokens() (gas: 152891)
[PASS] testNotifyRewardAmount() (gas: 116243)


ds-token

Running 12 tests for src/test/base.t.sol:DSTokenBaseTest
[PASS] testTransferCost() (gas: 30321)
[PASS] testAllowanceStartsAtZero() (gas: 7293)
[PASS] testFailTransferFromSelfNonArbitrarySize() (gas: 10369)
[PASS] testFailTransferWithoutApproval() (gas: 35463)
[PASS] testFailWrongAccountTransfers() (gas: 5908)
[PASS] testSetupPrecondition() (gas: 4907)
[PASS] testApproveSetsAllowance() (gas: 41780)
[PASS] testChargesAmountApproved() (gas: 68368)
[PASS] testFailChargeMoreThanApproved() (gas: 64079)
[PASS] testValidTransfers() (gas: 42717)
[PASS] testFailInsufficientFundsTransfers() (gas: 34563)
[PASS] testTransferFromSelf() (gas: 41009)

Running 54 tests for src/test/delegate.t.sol:DSDelegateTokenTest
[PASS] testApproveWillNotModifyAllowance() (gas: 89409)
[PASS] testFailTrustWhenStopped() (gas: 13397)
[PASS] testBurnAuth() (gas: 56274)
[PASS] testTransferCost() (gas: 33251)
[PASS] testMintGuy() (gas: 36666)
[PASS] testFailPullWhenStopped() (gas: 42975)
[PASS] testFailPullWithoutTrust() (gas: 9742)
[PASS] testMoveWithTrust() (gas: 63903)
[PASS] testMintAuth() (gas: 41999)
[PASS] testAllowanceStartsAtZero() (gas: 7271)
[PASS] testFailMoveWithoutTrust() (gas: 10648)
```

```
[PASS] testFailTransferFromWhenStopped() (gas: 17194)
[PASS] testFailMoveWhenStopped() (gas: 41109)
[PASS] testPullWithTrust() (gas: 62992)
[PASS] testBurnGuyAuth() (gas: 87764)
[PASS] testFailTransferFromSelfNonArbitrarySize() (gas: 11495)
[PASS] testFailUntrustedTransferFrom() (gas: 16376)
[PASS] testFailTransferWhenStopped() (gas: 13472)
[PASS] testFailTransferWithoutApproval() (gas: 40421)
[PASS] testFailWrongAccountTransfers() (gas: 7981)
[PASS] testFailMintWhenStopped() (gas: 13510)
[PASS] testFailTransferOnlyTrustedCaller() (gas: 69793)
[PASS] testSetupPrecondition() (gas: 4953)
[PASS] testPriorVotesReturnsLatestCheckpoint() (gas: 148632)
[PASS] testBurnself() (gas: 17573)
[PASS] testFailPriorVotesInvalidBlock() (gas: 4115)
[PASS] testBurnGuyWithTrust() (gas: 86086)
[PASS] testApproveSetsAllowance() (gas: 42871)
[PASS] testFailBurnGuyNoAuth() (gas: 70616)
[PASS] testFailMintGuyNoAuth() (gas: 8587)
[PASS] testMint() (gas: 15473)
[PASS] testPush() (gas: 100531)
[PASS] testPriorVotesReturnsCorrectVotingBalance() (gas: 436352)
[PASS] testFailMintGuyWhenStopped() (gas: 14461)
[PASS] testTrusting() (gas: 51586)
[PASS] testFailBurnWhenStopped() (gas: 13555)
[PASS] testMintGuyAuth() (gas: 42965)
[PASS] testFailMintNoAuth() (gas: 7565)
[PASS] testCheckpoints() (gas: 384484)
[PASS] testTrustedTransferFrom() (gas: 68957)
[PASS] testChargesAmountApproved() (gas: 73559)
[PASS] testFailPushWhenStopped() (gas: 13513)
[PASS] testFailChargeMoreThanApproved() (gas: 70083)
[PASS] testFailBurnNoAuth() (gas: 40108)
[PASS] testPriorVotesReturnsZeroIfNoCheckpoints() (gas: 148668)
[PASS] testMintself() (gas: 16596)
[PASS] testFailBurnGuyWithoutTrust() (gas: 40443)
[PASS] testBurn() (gas: 16471)
[PASS] testValidTransfers() (gas: 45715)
[PASS] testPriorVotesReturnsZeroForNoCheckpoints() (gas: 6159)
[PASS] testFailInsufficientFundsTransfers() (gas: 38483)
[PASS] testOneCheckpointPerBlock() (gas: 289934)
[PASS] testTransferFromSelf() (gas: 38304)
[PASS] testApproveWillModifyAllowance() (gas: 92624)

Running 47 tests for src/test/token.t.sol:DSTokenTest
[PASS] testApproveWillNotModifyAllowance() (gas: 87698)
[PASS] testFailTrustWhenStopped() (gas: 13395)
[PASS] testBurnAuth() (gas: 52478)
[PASS] testTransferCost() (gas: 31387)
[PASS] testMintGuy() (gas: 34779)
[PASS] testFailPullWhenStopped() (gas: 42951)
[PASS] testFailPullWithoutTrust() (gas: 9769)
[PASS] testMoveWithTrust() (gas: 61993)
[PASS] testMintAuth() (gas: 40089)
[PASS] testAllowanceStartsAtZero() (gas: 7381)
[PASS] testFailMoveWithoutTrust() (gas: 10697)
[PASS] testFailTransferFromWhenStopped() (gas: 17126)
[PASS] testFailMoveWhenStopped() (gas: 41041)
[PASS] testPullWithTrust() (gas: 61149)
[PASS] testBurnGuyAuth() (gas: 83967)
[PASS] testFailTransferFromSelfNonArbitrarySize() (gas: 11569)
[PASS] testFailUntrustedTransferFrom() (gas: 16513)
[PASS] testFailTransferWhenStopped() (gas: 13471)
[PASS] testFailTransferWithoutApproval() (gas: 38522)
[PASS] testFailWrongAccountTransfers() (gas: 7990)
[PASS] testFailMintWhenStopped() (gas: 13530)
[PASS] testFailTransferOnlyTrustedCaller() (gas: 67981)
[PASS] testSetupPrecondition() (gas: 4974)
[PASS] testBurnself() (gas: 15686)
[PASS] testBurnGuyWithTrust() (gas: 82204)
[PASS] testApproveSetsAllowance() (gas: 42891)
[PASS] testFailBurnGuyNoAuth() (gas: 68772)
[PASS] testFailMintGuyNoAuth() (gas: 8562)
[PASS] testMint() (gas: 13697)
[PASS] testPush() (gas: 96825)
[PASS] testFailMintGuyWhenStopped() (gas: 14348)
[PASS] testTrusting() (gas: 51738)
[PASS] testFailBurnWhenStopped() (gas: 13530)
[PASS] testMintGuyAuth() (gas: 40965)
[PASS] testFailMintNoAuth() (gas: 7697)
[PASS] testTrustedTransferFrom() (gas: 67136)
[PASS] testChargesAmountApproved() (gas: 71539)
[PASS] testFailPushWhenStopped() (gas: 13511)
[PASS] testFailChargeMoreThanApproved() (gas: 68204)
[PASS] testFailBurnNoAuth() (gas: 38309)
[PASS] testMintself() (gas: 14754)
[PASS] testFailBurnGuyWithoutTrust() (gas: 38389)
[PASS] testBurn() (gas: 14606)
[PASS] testValidTransfers() (gas: 43873)
[PASS] testFailInsufficientFundsTransfers() (gas: 36539)
[PASS] testTransferFromSelf() (gas: 36418)
[PASS] testApproveWillModifyAllowance() (gas: 90869)

Running 16 tests for src/test/per-second/PIRawPerSecondCalculator.t.sol:PIRawPerSecondCalculatorTest
[PASS] test_get_new_rate_no_warp_zero_current_integral() (gas: 88160)
[PASS] test_two_small_positive_deviations() (gas: 484912)
[PASS] testFail_update_same_period_warp() (gas: 188413)
[PASS] testFail_redemption_way_higher_than_market() (gas: 33229)
[PASS] test_first_small_negative_deviation() (gas: 257259)
[PASS] test_first_small_positive_deviation() (gas: 274397)
[PASS] test_normalized_pi_result() (gas: 383226)
[PASS] testFail_update_invalid_market_price() (gas: 177307)
[PASS] test_big_delay_positive_deviation() (gas: 464329)
[PASS] test_correct_setup() (gas: 87069)
[PASS] testFail_update_same_period_no_warp() (gas: 181891)
[PASS] test_modify_parameters() (gas: 76120)
[PASS] test_correct_proportional_calculation() (gas: 254275)
[PASS] test_first_update_rate_no_deviation() (gas: 192394)
[PASS] test_both_gains_zero() (gas: 244397)
[PASS] test_get_new_rate_no_proportional_no_integral() (gas: 124316)
```

Code Coverage

We were unable to perform code coverage due to the relative limitation of `dapp` right now. It is only able to return line-by-line results which will prove far too time-consuming to verify with all of the files involved.

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

```
8284946872596895d9640861aac378e1ed9ffed8f437430a290116c2e2e5c4f6 ./ds-token/src/delegate.sol
774ee7a9904b6a4c9532ba81545ee37195077c7ac6de83a1b05927eecfa34ab4 ./geb-rrfm-calculators/calciulator/PIRawPerSecondCalculator.sol
fe950ad6c2874edaadd33f0d948ef8f302ea48b11548fb8057f42ec898cc4d76 ./geb-rrfm-rate-setter/src/RateSetter.sol
8b2487add3919c941e19202f45903de080e34e32428bd3d2b86a2bdd908217af ./geb-proxy-actions/src/GebProxyActions.sol
8ee7c2d87d1d2e4558bc447fcd0d6ffe955af6b1b74317d7f6863953e3d8ff86 ./geb-debt-auction-param-setter/src/DebtAuctionInitialParameterSetter.sol
91cf3022715a27888efccb74bee680ae3b33de37a5de8f3df662a6095fc7dce1 ./ds-pause/protest-pause.sol
e5c7239d462f5badf277c648f0c0981fdb9e160658a0251c09568c0c162aecc7 ./geb-incentives/src/uniswap/GebUniswapRollingDistributionIncentives.sol
```

Changelog

- 2020-12-21 - Initial report
- 2021-01-11 - `ds-token` has been updated from the initial commit of `cea76c6` to `553626e`
- 2021-01-11 - `geb-rrfm-calculators` has been updated from the initial commit of `c55dfc8` to `785b3aa`
- 2021-01-11 - `geb-rrfm-rate-setter` has been updated from the initial commit of `6af5b82` to `5abfd0f`
- 2021-01-11 - `geb-proxy-actions` has been updated from the initial commit of `3f25c84` to `a6d068f`
- 2021-01-11 - `geb-debt-auction-param-setter` has been updated from the initial commit of `f252e67` to `b8d6f29`
- 2021-01-11 - `ds-pause` has been updated from the initial commit of `a609b7e` to `bab7bd2`
- 2021-01-11 - `geb-incentives` has been removed from scope entirely by the decision of the Reflexer team. The initial commit was `6c6caf5`. See `Overall Assessment` for more.

About Quantstamp

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

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

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

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