Quantstamp Security Assessment Certificate

May 10th 2021 — Quantstamp Verified

Badger

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

DRAFT

May 10th 2021

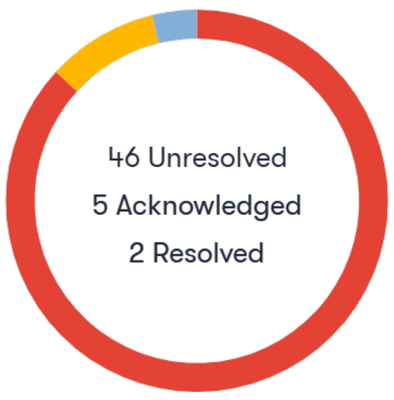
Executive Summary

Type	Bitcoin-centric DeFi System
Auditors	Ed Zulkoski, Senior Security Engineer Jake Goh Si Yuan, Senior Security Researcher Mohsen Ahmadvand, Senior Research Engineer
Timeline	2021-02-22 through 2021-05-10
EVM	Muir Glacier
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Manual Review
Specification	Badger Finance Gitbook
Documentation Quality	Medium
Test Quality	Undetermined

Source Code

Repository	Commit
badger-system	0cf31d2

Total Issues	**53** (2 Resolved)
High Risk Issues	**3** (0 Resolved)
Medium Risk Issues	**5** (1 Resolved)
Low Risk Issues	**7** (0 Resolved)
Informational Risk Issues	**25** (0 Resolved)
Undetermined Risk Issues	**13** (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).
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

During this initial tranche, 24 issues were noted ranging from high to undetermined severity. In addition to the findings below, we noted significant code cloning throughout the repository. We suggest using inheritance as opposed to code cloning where appropriate, which will reduce the overall code size and improve maintainability. Further, the `brownie test` scripts appear to have issues launching a `ganache` instance; logs are provided in the testing section.
During the second tranche, which primarily involved the `badger-sett` and `digg-core` directories, several new issues were noted. These can be found in issues QSP-25 onward, along with new findings appended to the best practices and documentation sections. In addition to the issues presented, we noted significant code-cloning throughout. Issues found may pertain to several contracts (e.g., `BaseStrategy.sol`, `BaseStrategySwapper.sol`, and `Swapper.sol`). We recommend modifying the architecture to reduce cloning.
Note: this report does not include `badger-hunt` and `StrategyPickleMetaFarm.sol`, as they were excluded from the audit. Further, the `yAffiliateWrapperV2.sol` file is taken from a later commit `c4d7e2f`.
Update: The report has been updated to commit 6bc6d41, however this tranche only pertained to `contracts-reference/StabilizeDiggSett.sol` and `contracts-reference/StabilizeStrategyDiggV1.sol`. New findings have been appended to each section including issues QSP-46 through QSP-53. Most notably, the two contracts have no corresponding test cases and limited documentation, making them more difficult to assess.
Update: The report has been updated based on commit 5bcc709. Issues QSP-46 - QSP-53 have been either acknowledged or addressed.

ID	Description	Severity	Status
QSP-1	Double add on `claimed` amount	High	Unresolved
QSP-2	Integer Overflow / Underflow	High	Unresolved
QSP-3	`_deposit` relies on external `vault` contract for critical minting functions	Medium	Unresolved
QSP-4	Unclear `yAffiliateFactoryV2._update` function	Medium	Unresolved
QSP-5	Potential mismatch in decimal values	Medium	Unresolved
QSP-6	`TYPEHASH` constants do not conform to the EIP-712 standard	Low	Unresolved
QSP-7	Potential duplicate events	Low	Unresolved
QSP-8	Unchecked function arguments	Low	Unresolved
QSP-9	`approveRoot` can be invoked multiple times with the same `pendingRoot`	Low	Unresolved
QSP-10	Privileged Roles and Ownership	Informational	Unresolved
QSP-11	`__Ownable_init()` is not invoked	Informational	Unresolved
QSP-12	Unlocked Pragma	Informational	Unresolved
QSP-13	`SettAccessControl` depends upon derived contracts for initialization	Informational	Unresolved
QSP-14	`EnumerableSetUpgradeable.at` does not preserve ordering	Informational	Unresolved
QSP-15	Unclear distribution token management	Informational	Unresolved
QSP-16	`RewardsEscrow.call` does not enforce inline requirements	Informational	Unresolved
QSP-17	Unused and Test events	Informational	Unresolved
QSP-18	`recoverERC20` may transfer to an unintended address if multiple admins are added	Informational	Unresolved
QSP-19	Unclear `Pausable` usage between `StakingRewards` and `StakingRewardsSignalOnly`	Informational	Unresolved
QSP-20	Allowance Double-Spend Exploit	Informational	Unresolved
QSP-21	`yAffiliateFactoryV2.deploy` allows for duplicate deployments	Informational	Unresolved
QSP-22	Undocumented constants in `SimpleTimelock`	Informational	Unresolved
QSP-23	Clone-and-Own	Informational	Unresolved
QSP-24	Unclear `vault` interface	Undetermined	Unresolved
QSP-25	Potential mismatch in decimal values	Medium	Unresolved
QSP-26	`earn` does not respect the `min` on-hand token amount	Low	Unresolved
QSP-27	Unchecked function arguments	Low	Unresolved
QSP-28	`_withdrawSome` assumes `geyser` will fulfill the short amount	Low	Unresolved
QSP-29	Unused events	Informational	Unresolved
QSP-30	All `StrategyCurveGauge*` contracts share the same `getName`	Informational	Unresolved
QSP-31	`Controller`'s `max` and `split` state variables are unused	Informational	Unresolved
QSP-32	`_withdrawSome` does not account for `_preWant`	Informational	Unresolved
QSP-33	`ReentrancyGuard` and `Pausable` are not initialized	Informational	Unresolved
QSP-34	Events will be emitted even if strategy approval has not changed	Informational	Unresolved
QSP-35	Missing inline documentation for complex functions	Informational	Unresolved
QSP-36	Allowance Double-Spend Exploit	Informational	Unresolved
QSP-37	`balance` functionality does not reflect inline documentation	Undetermined	Unresolved
QSP-38	`pause` and `unpause` have different access-control policies	Undetermined	Unresolved
QSP-39	Unclear divide-before-multiply in `_withdraw`	Undetermined	Unresolved
QSP-40	Unclear use of `sharesOfPool` function	Undetermined	Unresolved
QSP-41	Unclear `onlyAfterRebaseStart` modifier	Undetermined	Unresolved
QSP-42	Undocumented and unused "emergency flags"	Undetermined	Unresolved
QSP-43	Unresolved TODO comment	Undetermined	Unresolved

ID	Description	Severity	Status
QSP-44	Unclear distinction between `_crv` and `harvestData.crvHarvested`	? Undetermined	Unresolved
QSP-45	Unclear use of `_initialSharesPerFragment` in conversion view functions	? Undetermined	Unresolved
QSP-46	No test cases for reference contracts	⌃⌃ High	Acknowledged
QSP-47	Oracle data may not be recent	⌃ Medium	Fixed
QSP-48	Unresolved TODO in code	○ Informational	Acknowledged
QSP-49	Clone-and-Own	○ Informational	Acknowledged
QSP-50	Privileged Roles and Ownership	○ Informational	Unresolved
QSP-51	Unclear heuristic for choosing Uniswap vs Sushiswap in `exchange`	? Undetermined	Fixed
QSP-52	Underdocumented logic	? Undetermined	Acknowledged
QSP-53	Potential problematic usage of the `_beforeTokenTransfer` hook	? Undetermined	Acknowledged

Quantstamp Audit Breakdown

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

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

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

Methodology

The Quantstamp auditing process follows a routine series of steps:

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

Toolset

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

Setup

Tool Setup:

- Slither v0.6.14
- Mythril v0.22.16

Steps taken to run the tools:

1. Installed the Slither tool: `pip install slither-analyzer`
2. Run Slither from the project directory: `slither .`
3. Installed the Mythril tool from Pypi: `pip3 install mythril`
4. Ran the Mythril tool on each contract: `myth -x path/to/contract`

Findings

QSP-1 Double add on `claimed` amount

Severity: *High Risk*

Status: Unresolved

File(s) affected: `contracts/badger-geyser/BadgerTree.sol`

Description: Consider the following snippets of code taken from `_tryClaim` L294-295 and `_setClaimed` L269:

```
// within _tryClaim
uint256 claimedAfter = claimedBefore.add(toClaim);
 _setClaimed(account, token, claimedAfter);


// within _setClaimed
claimed[account][token] = claimed[account][token].add(amount);
```

This effectively double counts the `claimedBefore` when updating the `claimed` mapping, as `claimedAfter` already includes the `claimedBefore` amount (which is equivalent to `claimed[account][token]`).

Recommendation: Update the call to `_setClaimed` to pass `toClaim` instead of `claimedAfter`. Alternatively, change `_setClaimed` such that it does not perform addition, but simply sets the stored value to `amount`.

QSP-2 Integer Overflow / Underflow

Severity: *High Risk*

Status: Unresolved

File(s) affected: `contracts-reference/yAffiliateWrapperV2.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 several places in the contract, `+` and `-` are used instead of the SafeMath `.add` and `.sub` functions, which would prevent overflow and underflow from occurring. Most notably, `_transferTokens` is defined as:

```
function _transferTokens(address src, address dst, uint amount) internal {
        balances[src] -= amount;
        balances[dst] += amount;

        emit Transfer(src, dst, amount);
    }
```

This allows the `msg.sender` to underflow their balance, creating an arbitrarily large number of tokens for themself or `dst` in the process.
The issue occurs in several other locations including `transferFrom`, which would allow any user to transfer from any other address. The functions `_mint` and `_burn` are also affected.

Recommendation: Replace **all** uses of `+`, `-`, and `/` with the corresponding SafeMath `.add`, `.sub`, and `div` functions.

QSP-3 `_deposit` relies on external `vault` contract for critical minting functions

Severity: *Medium Risk*

Status: Unresolved

File(s) affected: `contracts-reference/yAffiliateWrapperV2.sol`

Description: The function `_deposit` relies on the return value of an external contract `vault` to decide on a very important parameter of a critical function `_mint`, which is not protected by SafeMath. This is potentially dangerous and could lead to strange behavior, especially with `totalSupply`.

Recommendation: Use SafeMath to protect `_mint` as noted in QSP-2. Ensure that the external `vault` contract is trusted.

QSP-4 Unclear `yAffiliateFactoryV2._update` function

Severity: *Medium Risk*

Status: Unresolved

File(s) affected: `contracts-reference/yAffiliateWrapperV2.sol`

Description: The function `_update` does not have any documentation. It is unclear why it is used. Further, the state variables that are updated (`index` and `bal`) are not used anywhere else in the code.
Additional, `_withdraw` does not check for return value of L214, which means that this could fail while the `msg.sender` has already `_burned` their tokens.

Recommendation: Clarify the `_update` function. If `bal` and `index` are not needed, remove the function and state variables.

QSP-5 Potential mismatch in decimal values

Severity: *Medium Risk*

Status: Unresolved

File(s) affected: `contracts-reference/yAffiliateWrapperV2.sol`

Description: In the `_update` function, the `decimals` variable is used to compute `_ratio = _diff * 10**decimals / totalSupply`. However, in `currentContribution`, the constant `1e18` is used: `return 1e18 * IERC20(vault).balanceOf(address(this)) / IERC20(vault).totalSupply();`. It is not clear if `1e18` is always correct, or if `decimals` should be used.

Recommendation: Clarify that decimal conversions happen correctly.

QSP-6 `TYPEHASH` constants do not conform to the EIP-712 standard

Severity: *Low Risk*

Status: Unresolved

File(s) affected: `contracts-reference/yAffiliateWrapperV2.sol`

Description: From the EIP-712 standard:

> The atomic types are `bytes1` to `bytes32`, `uint8` to `uint256`, `int8` to `int256`, `bool` and `address`. These correspond to their definition in Solidity. Note that there are no aliases `uint` and `int`.

However, `DOMAIN_TYPEHASH` is defined using `uint`: `keccak256("EIP712Domain(string name,uint chainId,address verifyingContract)")`.
This also occurs for `PERMIT_TYPEHASH`.

Recommendation: Change `uint` to `uint256` in both `DOMAIN_TYPEHASH` and `PERMIT_TYPEHASH`.

QSP-7 Potential duplicate events

Severity: *Low Risk*

Status: Unresolved

File(s) affected: `contracts/badger-geyser/RewardsEscrow.sol`, `contracts/badger-timelock/SimpleTimelock.sol`

Description: In `RewardsEscrow.sol` in `approveRecipient` and `revokeRecipient`, the `recipient` may already be approved or revoked permissions-wise, and therefore the event emitted would be misleading.
In `SimpleTimelock.sol`, a similar issue exists for `approveTransfer` and `revokeTransfer`.

Recommendation: Do not emit an event if there are no state updates in the contract.

QSP-8 Unchecked function arguments

Severity: *Low Risk*

Status: Unresolved

File(s) affected: `contracts/digg-oracles/MedianOracle.sol`, `contracts/badger-sett/SettAccessControl.sol`, `contracts/badger-geyser/BadgerTree.sol`, `contracts/badger-timelock/GovernanceTimelock.sol`, `contracts/badger-timelock/SmartTimelock.sol`, `contracts/badger-geyser/BadgerGeyser.sol`, `contracts/badger-geyser/StakingRewards.sol`, `contracts/badger-geyser/StakingRewardsSignalOnly.sol`, `contracts-reference/yAffiliateWrapperV2.sol`

Description: There are several functions that should validate their input arguments against common faulty values:

1. In `contracts/digg-oracles/MedianOracle.sol`:
   - In both `constructor` and `setReportDelaySec`, it should be ensured that `reportDelaySec < reportExpirationTimeSec_`.
   - Similarly, `setReportExpirationTimeSec` should ensure that `reportExpirationTimeSec_ > reportDelaySec`.
2. In `contracts/badger-sett/SettAccessControl.sol`, the functions `setStrategist`, `setKeeper`, and `setGovernance` should all check that the address argument is non-zero.
3. In `contracts/badger-geyser/BadgerTree.sol`:
   - The `initialize` function should check that the three address arguments are non-zero.
   - `getClaimableFor` should first check that `token.length == cumulativeAmounts.length`.
4. In `contracts/badger-timelock/GovernanceTimelock.sol`:
   - The `constructor` should ensure that `admin_` and `guardian_` are non-zero.
   - `setPendingAdmin` should ensure that `pendingAdmin_` is non-zero.
5. In `contracts/badger-timelock/SmartTimelock.sol`, `initialize` should check that address arguments are non-zero.
6. In `contracts/badger-geyser/BadgerGeyser.sol`:
   - `initialize` should check that address arguments are non-zero.
   - `initialize` does not validate for whether `globalStartTime_` is more than or equal to the current blocktime, which means possibly starting something in the past. It is not clear if this should be allowed.
   - `addDistributionToken` does not check if `token` is already in `distributionTokens`.
7. In both `contracts/badger-geyser/StakingRewards.sol` and `contracts/badger-geyser/StakingRewardsSignalOnly.sol`:
   - `initialize` should check that address arguments are non-zero.
   - `setRewardsDuration` should ensure that `_rewardsDuration != 0`. Note that if set to zero, `notifyRewardAmount` will fail due to division-by-zero.
8. In `contracts-reference/yAffiliateWrapperV2.sol`:
   - The `constructor` and `setAffiliate` should check that all address arguments are non-zero.

- `setGovernance` should check that `_gov != msg.sender`.
- `setAffiliate` should check that `_affiliate != affiliate`.

Recommendation: Add the corresponding `require` statements to each function.

QSP-9 `approveRoot` can be invoked multiple times with the same `pendingRoot`

Severity: *Low Risk*

Status: Unresolved

File(s) affected: `contracts/badger-geyser/BadgerTree.sol`

Description: The function `approveRoot` does not update any of the state variables checked in the `require` statements from L215-220 (i.e., `pendingMerkleRoot`, `pendingMerkleContentHash`, `pendingCycle`, `lastProposeStartBlock`, and `lastProposeEndBlock`). This allows repeat calls to the function with the same arguments, and consequently a duplicate `RootUpdated` event to be emitted.
Although this function requires privileged access, errors in external scripts/bots may cause this "duplicate approval" scenario to occur. It is unclear if the duplicate `RootUpdated` event would affect surrounding infrastructure.

Recommendation: Clear the `pending*` variables once they have been approved.

QSP-10 Privileged Roles and Ownership

Severity: *Informational*

Status: Unresolved

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 `badger-geyser` contracts inherently rely upon off-chain privileged "bots" which may be either faulty or unresponsive at any point.

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.

QSP-11 `__Ownable_init()` is not invoked

Severity: *Informational*

Status: Unresolved

File(s) affected: `contracts/badger-timelock/vesting/SingleTokenVestingNonRevocable.sol`, `contracts/badger-timelock/vesting/SmartVesting.sol`

Description: The function `__SingleTokenVestingNonRevocable_init` does not invoke `__Ownable_init`, meaning that the `owner` will never be set. This affects both `SingleTokenVestingNonRevocable` and `SmartVesting`, however there does not appear to be any functions that are `onlyOwner`.

Recommendation: Invoke `__Ownable_init` in `__SingleTokenVestingNonRevocable_init`, or inherit directly from `Initializable` if ownership is not needed.

QSP-12 Unlocked Pragma

Severity: *Informational*

Status: Unresolved

File(s) affected: `All Contracts`

Description: Every Solidity file specifies in the header a version number of the format `pragma solidity (^)0.6.*`. The caret (^) before the version number implies an unlocked pragma, meaning that the compiler will use the specified version *and above*, hence the term "unlocked".

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

QSP-13 `SettAccessControl` depends upon derived contracts for initialization

Severity: *Informational*

Status: Unresolved

File(s) affected: `contracts/badger-sett/SettAccessControl.sol`

Description: Three variables (`governance`, `strategist`, and `keeper`) are declared but not set in some `initialize` function. The variables must be set downstream, as in `BaseStrategy.sol`. It would be more modular to include an `initialize` function in `SettAccessControl` to set the variables. Further, this reduces the likelihood of forgetting to initialize one or more of the variables.

Recommendation: It would be more modular to include an `initialize` function in `SettAccessControl` to set the variables. Further, this reduces the likelihood of forgetting to initialize one or more of the variables.

QSP-14 `EnumerableSetUpgradeable.at` does not preserve ordering

Severity: *Informational*

Status: Unresolved

File(s) affected: `contracts/badger-geyser/BadgerGeyser.sol`

Description: From `EnumerableSetUpgradeable` L178-179, we have the following comment: "Note that there are no guarantees on the ordering of values inside the array, and it may change when more values are added or removed." If the `distributionTokens` set is updated, the order returned by `getDistributionTokens` may not be preserved.
While we do not see any issues in the solidity code related to the ordering, we include this finding in case external scripts may rely upon ordering.

Recommendation: Ensure external scripts/bots are not dependent on the set ordering.

QSP-15 Unclear distribution token management

Severity: Informational

Status: Unresolved

File(s) affected: `contracts/badger-geyser/BadgerGeyser.sol`

Description: In the comment block for `signalTokenLock`, it states: "@param amount Number of distribution tokens to lock. These are transferred from the caller.". However, no such transfer occurs. If a transfer is intended here, it should likely occur from the `msg.sender` to the `BadgerTree` contract (not `BadgerGeyser`), as that is the contract where users claim rewards.

Recommendation: Either update the comment block, or include a `transfer` if intended.

QSP-16 `RewardsEscrow.call` does not enforce inline requirements

Severity: Informational

Status: Unresolved

File(s) affected: `contracts/badger-geyser/RewardsEscrow.sol`

Description: In the comment block on L46, we have: "@notice Allows the timelock to call arbitrary contracts, as long as it does not reduce it's locked token balance". This is not enforced in the function `call`, unlike in similar functions such as `SmartTimelock.call`.

Recommendation: Since the token address is unspecified in this contract, it is likely more appropriate to update the comment block rather than add a post-condition to `call` (unlike in `SmartTimelock` where the token is known).

QSP-17 Unused and Test events

Severity: Informational

Status: Unresolved

File(s) affected: `contracts/badger-geyser/RewardsFaucet.sol`, `contracts/badger-geyser/DiggRewardsFaucet.sol`, `contracts/badger-geyser/StakingRewards.sol`, `contracts/badger-geyser/StakingRewardsSignalOnly.sol`

Description: In the two `StakingRewards*` contracts, the event `Test` should likely be renamed in production. This appears to correspond to `NotifyRewardsAmount` in the similar `RewardsFaucet` contract.
Further, the events `Staked`, `Withdrawn`, and `Recovered` are unused in the `RewardsFaucet` and `DiggRewardsFaucet` contracts. These appear to be cloned from the `StakingRewards` contract.

Recommendation: Remove the events or add the corresponding functionality. Update the name of the `Test` event.

QSP-18 `recoverERC20` may transfer to an unintended address if multiple admins are added

Severity: Informational

Status: Unresolved

File(s) affected: `contracts/badger-geyser/StakingRewards.sol`, `contracts/badger-geyser/StakingRewardsSignalOnly.sol`

Description: The function `recoverERC20` uses `getRoleMember(DEFAULT_ADMIN_ROLE, 0)` to obtain an admin address to send the tokens. If multiple admins are added to the role, it is not guaranteed that the initial admin (as set during the `initialize` function) will be the address to receive funds. This may occur since `EnumerableSetUpgradeable.at` does not guarantee an ordering on values.

Recommendation: Clarify if this scenario could be problematic in practice. If necessary, store the initial admin address separate from the underlying Set data structure to distinguish it.

QSP-19 Unclear `Pausable` usage between `StakingRewards` and `StakingRewardsSignalOnly`

Severity: Informational

Status: Unresolved

File(s) affected: `contracts/badger-geyser/StakingRewards.sol`, `contracts/badger-geyser/StakingRewardsSignalOnly.sol`

Description: Although the two contracts `StakingRewards` and `StakingRewardsSignalOnly` are similar for most functionality, the `Pausable` policy is significantly different. In particular the contract `StakingRewards` allows the functions `withdraw`, `getReward`, `notifyRewardAmount`, `recoverERC20`, and `setRewardsDuration` to be invoked when `_paused == true`, unlike in `StakingRewardsSignalOnly`.

Recommendation: Clarify if the proper functions have a `whenNotPaused` modifier.

QSP-20 Allowance Double-Spend Exploit

Severity: Informational

Status: Unresolved

File(s) affected: `contracts-reference/yAffiliateWrapperV2.sol`

Description: As it presently is constructed, the contract is vulnerable to the allowance double-spend exploit, as with other ERC20 tokens.

Exploit Scenario:

1. Alice allows Bob to transfer `N` amount of Alice's tokens (`N>0`) by calling the `approve()` method on `Token` smart contract (passing Bob's address and `N` as method arguments)
2. After some time, Alice decides to change from `N` to `M` (`M>0`) the number of Alice's tokens Bob is allowed to transfer, so she calls the `approve()` method again, this time passing Bob's address and `M` as method arguments
3. Bob notices Alice's second transaction before it was mined and quickly sends another transaction that calls the `transferFrom()` method to transfer `N` Alice's tokens somewhere
4. If Bob's transaction will be executed before Alice's transaction, then Bob will successfully transfer `N` Alice's tokens and will gain an ability to transfer another `M` tokens
5. Before Alice notices any irregularities, Bob calls `transferFrom()` method again, this time to transfer `M` Alice's tokens.

Recommendation: The exploit (as described above) is mitigated through use of functions that increase/decrease the allowance relative to its current value, such as `increaseAllowance()` and `decreaseAllowance()`.
Pending community agreement on an ERC standard that would protect against this exploit, we recommend that developers of applications dependent on `approve()` / `transferFrom()` should keep in mind that they have to set allowance to 0 first and verify if it was used before setting the new value. Teams who decide to wait for such a standard should make these recommendations to app developers who work with their token contract.

QSP-21 `yAffiliateFactoryV2.deploy` allows for duplicate deployments

Severity: Informational

Status: Unresolved

File(s) affected: `contracts-reference/yAffiliateWrapperV2.sol`

Description: The `deploy` function allows multiple deployments with the same exact arguments. It is not clear if this is intended.

Recommendation: Clarify if this scenario is allowed. If not, add `require` statements preventing duplicate deployments.

QSP-22 Undocumented constants in `SimpleTimelock`

Severity: Informational

Status: Unresolved

File(s) affected: `contracts/badger-timelock/SimpleTimelock.sol`

Description: The two addresses `0xB65cef03b9B89f99517643226d76e286ee999e77` and `0x8dE82C4C968663a0284b01069DDE6EF231D0Ef9B`, along with the constant `7350000 ether` are undocumented.

Recommendation: Clarify that these constants are correct. Add inline documentation regarding these constants.

QSP-23 Clone-and-Own

Severity: Informational

Status: Unresolved

File(s) affected: `contracts-reference/yAffiliateWrapperV2.sol`

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.
In particular, L28-L67 appear to be cloned from old OpenZeppelin code.

Recommendation: Rather than the clone-and-own approach, a good industry practice is to use the Truffle framework for managing library dependencies. This eliminates the clone-and-own risks yet allows for following best practices, such as, using libraries.

QSP-24 Unclear `vault` interface

Severity: Undetermined

Status: Unresolved

File(s) affected: `contracts-reference/yAffiliateWrapperV2.sol`

Description: The `yAffiliateTokenV2` contract relies upon the `IyVault` interface in several functions. In particular, the `_deposit` function invokes `_mint(msg.sender, IyVault(vault).deposit(amount, address(this)));`, and `_withdraw` invokes `IyVault(vault).withdraw(amount, msg.sender, maxLoss);`.
It is not clear which concrete vault contract will be called from these functions, but of note, the yVault contract does not appear to adhere to this interface.

Recommendation: Clarify which `Vault` contracts support the `IyVault` interface.

QSP-25 Potential mismatch in decimal values

Severity: Medium Risk

Status: Unresolved

File(s) affected: `contracts/badger-sett/Sett.sol`

Description: In `getPricePerFullShare`, it is assumed that 18 decimals are used, but this may not always be the case.
This relates to QSP-5, however we include a new issue as part of the second tranche.

Recommendation: Ensure that token-related decimal arithmetic is consistent.

QSP-26 `earn` does not respect the `min` on-hand token amount

Severity: *Low Risk*

Status: Unresolved

File(s) affected: `contracts/badger-sett/Sett.sol`

Description: The function `available` "sets minimum required on-hand to keep small withdrawals cheap"; in effect, this function is used such that only 95% of tokens will be sent to the `controller`, and 5% will remain for withdrawals. However, `available` only considers tokens in `Sett` itself (not in the controller), so if the `earn` function is called twice in a row, only 5% * 5% = 0.25% of funds will remain in the `Sett`. This process can be repeated an arbitrary number of times.

Recommendation: In `earn`, replace `uint256 _bal = available();` with ` uint256 _bal = balance();

QSP-27 Unchecked function arguments

Severity: *Low Risk*

Status: Unresolved

File(s) affected: `contracts/badger-sett/UnlockScheduler.sol` `contracts/badger-sett/Sett.sol` `contracts/badger-sett/Controller.sol` `contracts/badger-sett/strategies/Swapper.sol` `contracts/badger-sett/strategies/BaseStrategy.sol` `contracts/badger-sett/strategies/badger/StrategyBadgerRewards.sol`, `contracts/badger-sett/BadgerRewardsManager.sol`, `strategies/harvest/StrategyHarvestMetaFarm.sol`, `strategies/curve/StrategyCurveGaugebase.sol`, `contracts/digg-core/UFragments.sol`

Description: 1. In `contracts/badger-sett/UnlockScheduler.sol`, `initialize` should check that all address arguments are non-zero.

1. In `contracts/badger-sett/Sett.sol`:
    - `initialize` should check that all address arguments are non-zero. Further, it is not clear why `strategist` is set to `address(0)` and cannot be updated after initialization.
    - `setGuardian` and `setController` should ensure argumentst are non-zero.
    - `setMin` should ensure that `_min <= max`.
2. In `contracts/badger-sett/Controller.sol`:
    - `initialize` should check that all address arguments are non-zero.
    - `setOneSplit` and `setRewards` should check that address arguments are non-zero.
    - `setSplit` should vallidate that `split <= max`.
3. In `contracts/badger-sett/strategies/Swapper.sol`, `_processFee` should ensure that `feeBps <= MAX_FEE`.
4. In `contracts/badger-sett/strategies/BaseStrategy.sol`:
    - `__BaseStrategy_init` should check that all address-arguments are non-zero.
    - `_processWithdrawalFee` should check that `IController(controller).rewards()` is non-zero to avoid burning.
    - `_processFee` should validate that `feeBps <= MAX_FEE`, and that `amount` and `recipient` are non-zero.
5. In `contracts/badger-sett/strategies/badger/StrategyBadgerRewards.sol`, `initialize` should check that each fee is `<= MAX_FEE`, and that address arguments are non-zero. This `MAX_FEE` check should generally occur through most strategies.
6. `contracts/badger-sett/BadgerRewardsManager.sol`, `initialize` should check that all address arguments are non-zero.
7. In `strategies/harvest/StrategyHarvestMetaFarm.sol`, `setFarmPerformanceFeeGovernance` and `setFarmPerformanceFeeStrategist` should validate input against `MAX_FEE`.
8. In `strategies/curve/StrategyCurveGaugebase.sol` on L78, we have the comment "// 1000" associated with `keepCRV`. This should likely be validated against `MAX_FEE`.
9. In `contracts/digg-core/UFragments.sol`, `setMonetaryPolicy` does not ensure that `monetaryPolicy` is non-zero (nor does it check if the new policy is different than the existing one).

Recommendation: Add the corresponding `require` statements to each function.

QSP-28 `_withdrawSome` assumes `geyser` will fulfill the short amount

Severity: *Low Risk*

Status: Unresolved

File(s) affected: `strategies/uni/StrategyBadgerLpMetaFarm.sol`

Description: This method assumes that the `_toWithdraw` taken from `geyser` will make up for the short fall, but that may not necessarily be true. It is only the optimal behavior.

Recommendation: Check for actual amount available in `geyser` instead.

QSP-29 Unused events

Severity: *Informational*

Status: Unresolved

File(s) affected: `contracts/badger-sett/BadgerRewardsManager.sol` `contracts/badger-sett/strategies/BaseStrategy.sol`

Description: 1. In `contracts/badger-sett/BadgerRewardsManager.sol`, the event `Call` declared on L25 is not emitted anywhere. It is not clear if this should be removed, or if this event should be omitted somewhere.

1. In `contracts/badger-sett/strategies/BaseStrategy.sol`, all events from L35-43 (i.e., the `Withdraw*` and `Set*` events) are not used.

Recommendation: Either remove the event declarations, or add event emissions where appropriate.

QSP-30 All `StrategyCurveGauge*` contracts share the same `getName`

Severity: Informational

Status: Unresolved

File(s) affected: `contracts/badger-sett/strategies/curve/StrategyCurveGaugeBase.sol` `contracts/badger-sett/strategies/curve/StrategyCurveGaugeRenBtcCrv.sol` `contracts/badger-sett/strategies/curve/StrategyCurveGaugeSbtcCrv.sol` `contracts/badger-sett/strategies/curve/StrategyCurveGaugeTbtcCrv.sol`

Description: All four contracts return "StrategyCurveGauge" for `getName`. This may potentially cause confusion when users interact with these contracts on-chain.

Recommendation: Override `getName` in the derived contracts.

QSP-31 `Controller`'s `max` and `split` state variables are unused

Severity: Informational

Status: Unresolved

File(s) affected: `contracts/badger-sett/Controller.sol`

Description: These variables are not used throughout any functions. Note that they appear to be leftover from the yearn Controller contract, where they are used in the `yearn` function.

Recommendation: Either update the relevant functions to use these variables, or remove them from the contract.

QSP-32 `_withdrawSome` does not account for `_preWant`

Severity: Informational

Status: Unresolved

File(s) affected: `contracts/badger-sett/strategies/harvest/StrategyHarvestMetaFarm.sol`

Description: If `_preWant >= _amount`, it would appear that the function could immediately return `_amount`. Additionally, the `_wrappedToWithdraw` amount does not take into account existing `_preWant` tokens.

Recommendation: Consider revising the `_withdrawSome` logic to reduce the amount of withdrawal necessary.

QSP-33 `ReentrancyGuard` and `Pausable` are not initialized

Severity: Informational

Status: Unresolved

File(s) affected: `contracts/badger-sett/BadgerRewardsManager.sol`, `contracts/badger-sett/Sett.sol`, `contracts/badger-sett/UnlockScheduler.sol`

Description: In `BadgerRewardsManager.sol`, the function `__ReentrancyGuard_init` is not called during initialization, and therefore `ReentrancyGuardUpgradeable._status` will not be initially set to `_NOT_ENTERED`. Fortunately, this does not seem to alter the logic associated with the `nonReentrant` modifier.
`Sett.sol`, does not initialize `Pausable`.
`UnlockScheduler.sol` does not initialize `ReentrancyGuardUpgradeable`.

Recommendation: Invoke `__ReentrancyGuard_init` from `BadgerRewardsManager.initialize`.

QSP-34 Events will be emitted even if strategy approval has not changed

Severity: Informational

Status: Unresolved

File(s) affected: `contracts/badger-sett/BadgerRewardsManager.sol`

Description: The functions `approveStrategy` and `revokeStrategy` will emit events regardless of if the `isApprovedStrategy[recipient]` actually changes.

Recommendation: Only emit an event if the underlying state changes.

QSP-35 Missing inline documentation for complex functions

Severity: Informational

Status: Unresolved

File(s) affected: `strategies/harvest/StrategyHarvestMetaFarm.sol`, `strategies/curve/StrategyCurveGaugebase.sol`, `strategies/sushi/StrategySushiBadgetWbtc.sol`

Description: The function `_withdrawAll` makes use of several undocumented interfaces and external calls related to `metaFarm`, `vaultFarm`, and `harvestVault`. This function should

contain more documentation in order to better assess correctness.
As one example question for `StrategyHarvestMetaFarm._withdrawAll`, why are we sending *all* of the farm to the rewards address and not harvesting the gains to `BadgerTree` or distributing to stakers?
In `StrategyCurveGaugebase.harvest`, it is not clear why why `_crv` was declared when it's the exact same as `_afterCrv`, nor is it clear exactly what `keepCrv` refers to.
In `StrategySushiBadgetWbtc._withdrawAll`, it is not clear why everything being moved towards `rewards()` when it is also the place where `governanceFees` are moved towards.

Recommendation: In general, more inline documentation should be added to all strategy contracts.

QSP-36 Allowance Double-Spend Exploit

Severity: *Informational*

Status: Unresolved

File(s) affected: `contracts/digg-core/UFragments.sol`

Description: As it presently is constructed, the contract is vulnerable to the allowance double-spend exploit, as with other ERC20 tokens.

Exploit Scenario:

1. Alice allows Bob to transfer `N` amount of Alice's tokens (`N>0`) by calling the `approve()` method on `Token` smart contract (passing Bob's address and `N` as method arguments)
2. After some time, Alice decides to change from `N` to `M` (`M>0`) the number of Alice's tokens Bob is allowed to transfer, so she calls the `approve()` method again, this time passing Bob's address and `M` as method arguments
3. Bob notices Alice's second transaction before it was mined and quickly sends another transaction that calls the `transferFrom()` method to transfer `N` Alice's tokens somewhere
4. If Bob's transaction will be executed before Alice's transaction, then Bob will successfully transfer `N` Alice's tokens and will gain an ability to transfer another `M` tokens
5. Before Alice notices any irregularities, Bob calls `transferFrom()` method again, this time to transfer `M` Alice's tokens.

Recommendation: The exploit (as described above) is mitigated through use of functions that increase/decrease the allowance relative to its current value, such as `increaseAllowance()` and `decreaseAllowance()`.
Pending community agreement on an ERC standard that would protect against this exploit, we recommend that developers of applications dependent on `approve()` / `transferFrom()` should keep in mind that they have to set allowance to 0 first and verify if it was used before setting the new value. Teams who decide to wait for such a standard should make these recommendations to app developers who work with their token contract.

QSP-37 `balance` functionality does not reflect inline documentation

Severity: *Undetermined*

Status: Unresolved

File(s) affected: `contracts/badger-sett/Sett.sol`

Description: The comment states that it "Sums the balance in the Sett, the Controller, and the Strategy". However, the function only seems to sum the Sett and Controller.

Recommendation: Clarify either the function or comment.

QSP-38 `pause` and `unpause` have different access-control policies

Severity: *Undetermined*

Status: Unresolved

File(s) affected: `contracts/badger-sett/Sett.sol`, `contracts/badger-sett/strategies/BaseStrategy.sol`

Description: The function `pause` can be called by both the `guardian` and `governance`, whereas `unpause` can only happen by `governance`. While this may not be an issue, since we did not see related documentation to this discrepancy, we include the issue as a precaution.

Recommendation: Clarify if the access-control policy is correct. Add inline documentation.

QSP-39 Unclear divide-before-multiply in `_withdraw`

Severity: *Undetermined*

Status: Unresolved

File(s) affected: `contracts/badger-sett/DiggSett.sol`

Description: We have the following in `_withdraw`:

```
// uint256 _sharesToRedeem = (shares().mul(_bDiggToBurn)).div(totalSupply());
uint256 _sharesToRedeem = (shares().div(totalSupply())).mul(_bDiggToBurn);
```

Typically, it is recommended to multiply before dividing to mitigate truncation issues. It is not clear why the opposite approach is chosen.

Recommendation: Clarify the `_sharesToRedeem` calculation with inline documentation.

QSP-40 Unclear use of `sharesOfPool` function

Severity: *Undetermined*

Status: Unresolved

File(s) affected: `interfaces/digg/IDiggStrategy.sol contracts/badger-sett/strategies/digg/StrategyDiggRewards.sol`

Description: `IDiggStrategy` defines `sharesOfPool`, however this function is only defined in `StrategyDiggRewards` and returns 0, with the comment "No strategy positions for the native rewards strategy". It is not clear why this function is defined or if it is implemented correctly.

Recommendation: Remove the function if it does not serve any purpose.

QSP-41 Unclear `onlyAfterRebaseStart` modifier

Severity: *Undetermined*

Status: Unresolved

File(s) affected: `contracts/digg-core/UFragments.sol`

Description: Since `rebaseStartTime` is always set to zero in the `initialize` function, it is not clear why the check `require(now >= rebaseStartTime);` is needed.

Recommendation: Clarify the use of `onlyAfterRebaseStart`.

QSP-42 Undocumented and unused "emergency flags"

Severity: *Undetermined*

Status: Unresolved

File(s) affected: `contracts/badger-sett/strategies/harvest/StrategyHarvestMetaFarm.sol`

Description: The two booleans `_emergencyTestTrasferFlag` and `_emergencyFullTrasferFlag` currently have no uses.

Recommendation: Clarify the use of these two variables; remove if no longer needed.

QSP-43 Unresolved TODO comment

Severity: *Undetermined*

Status: Unresolved

File(s) affected: `strategies/harvest/StrategyHarvestMetaFarm.sol`

Description: On L120 associated with `balanceOfPool`, it states "TODO: If this is wrong, it will overvalue our shares (we will get LESS for each share we redeem) This means the user will lose out.".

Recommendation: This comment should be resolved (and tested correct).

QSP-44 Unclear distinction between `_crv` and `harvestData.crvHarvested`

Severity: *Undetermined*

Status: Unresolved

File(s) affected: `strategies/curve/StrategyCurveGaugebase.sol`

Description: In `harvest`, it is not clear if it is valid for L149 to use `_crv` instead of `harvestData.crvHarvested` to reward, given that the actual harvest was the latter.

Recommendation: Clarify if this is intended and add inline documentation.

QSP-45 Unclear use of `_initialSharesPerFragment` in conversion view functions

Severity: *Undetermined*

Status: Unresolved

File(s) affected: `contracts/digg-core/UFragments.sol`

Description: In `scaledSharesToShares` and `sharesToScaledShares`, it is not clear why `_initialSharesPerFragment` is used instead of `_sharesPerFragment`.

Recommendation: Clarify these functions with inline documentation.

QSP-46 No test cases for reference contracts

Severity: *High Risk*

Status: Acknowledged

File(s) affected: `StabilizeDiggSett.sol, StabilizeStrategyDiggV1.sol`

Description: There are no test cases associated with the two reference contracts. This makes it more difficult to assess the correctness of the code and significantly increases the chance of unforeseen issues.

Recommendation: Implement a test suite for the two contracts. Use coverage tools to ensure the tests cover a high percentage of the code.

Update: From the developers -- Added some minimal tests however Badger has a large testing suite that can test this strategy within a controlled environment. This is required as the strategy depends on some Badger controlled contracts. The expectation is that they will integrate this into their testing suite before launch.

QSP-47 Oracle data may not be recent

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `StabilizeStrategyDiggV1.sol`

Description: The contract uses `priceOracle.latestRoundData` but does not check if the price has been recently updated. This may lead to incorrect values used during WBTC/Digg exchanges.

Recommendation: Use the `updatedAt` and `answeredInRound` values returned from `latestRoundData`. Consider using multiple price oracles if redundancy and reliably fresh data is of high importance.

Update: The `updatedAt` value is used; they still do not use the `answeredInRound` value

QSP-48 Unresolved TODO in code

Severity: *Informational*

Status: Acknowledged

File(s) affected: `StabilizeStrategyDiggV1.sol`

Description: On L1182, we have "// TODO, place holder for digg treasury".

Recommendation: Resolve any remaining TODOs.

Update: TODO has been updated but there is still work to be done by Badger team to create a contract that can interface with this strategy.

QSP-49 Clone-and-Own

Severity: *Informational*

Status: Acknowledged

File(s) affected: `StabilizeDiggSett.sol`, `StabilizeStrategyDiggV1.sol`

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, a good industry practice is to use the Truffle framework for managing library dependencies. This eliminates the clone-and-own risks yet allows for following best practices, such as, using libraries.

Update: From the developers -- No change as clone and own was used purposefully to make this strategy easier for Badger team to understand integration.

QSP-50 Privileged Roles and Ownership

Severity: *Informational*

Status: Unresolved

File(s) affected: `StabilizeStrategyDiggV1.sol`

Description: Governance is allowed to transfer out wBTC from the strategy contract. Further, an initialization parameter `strategyLockedUntil` will disallow withdrawals up until a specified time.

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 developers -- No change as users understand that Badger governance has superuser functionality. Also governance is behind timelock (my understanding).

QSP-51 Unclear heuristic for choosing Uniswap vs Sushiswap in `exchange`

Severity: *Undetermined*

Status: Fixed

File(s) affected: `StabilizeStrategyDiggV1.sol`

Description: In L1348-1352, the function `exchange` determines a split between using Uniswap and Sushiswap, but only considers the number of Digg tokens in each protocol (note that this is regardless of whether Digg is the `_inputToken` or `_outputToken`). It is not clear why the balance of wBTC is not also considered, as the total reserves of both tokens will influence the price.

Recommendation: Consider revising the split logic, or document the design decision.

QSP-52 Underdocumented logic

Severity: *Undetermined*

Status: Acknowledged

File(s) affected: `StabilizeDiggSett.sol`, `StabilizeStrategyDiggV1.sol`

Description:

- `StabilizeDiggSett.sol:1499` -- why is strategist set to address(0)?
- `StabilizeDiggSett.sol:1504` -- why is min set to 9500?
- `StabilizeDiggSett.sol:1674` -- it is not clear what is meant by the comment: `// No rebalance implementation for lower fees and faster swaps`

- `StabilizeDiggSett.sol:1680-1689` -- it is not clear what this if-clause is doing.
- `StabilizeStrategyDiggV1.sol`-- the roles of guardian, keeper, strategist, controller, and governance and their permissions need to be explicitly documented.
- the `StabilizeStrategyDiggV1:_postDeposit` -- function contains no op.
- `StabilizeStrategyDiggV1:_withdrawAll();` -- contains no code.
- `StabilizeStrategyDiggV1:1219` -- use of fee variable for block number.
  - `strategyLockedUntil = _feeConfig[4]; // Deployer can optionally lock strategy from withdrawing until a certain blocknumber`
- `StabilizeStrategyDiggV1:1332` -- `exchange` function needs a better description of the intended actions.
- `StabilizeStrategyDiggV1:1408` -- is zero an acceptable value for `_mPDigg` and `_mPWBTC`?Better document the intended logic and follow the inlined recommendations of each item (when provided)

Recommendation: Better document the intended logic and follow the inlined recommendations of each item (when provided).

Update: From the developers --

- No change to line 1499 - as cloned from BadgerSett reference code
- No change to line 1504 - as cloned from BadgerSett reference code
- No change to line 1680 - as cloned from BadgerSett reference code
- No change to description of guardian, keeper, strategist, controller and governance as they are defined by Badger docs
- No change to _postDeposit as it is kept for reference
- No change to _withdrawAll() as it is kept for reference
- Changed DiggV1 line 1219, strategyLockedUntil now has its own parameter
- exchange function in DiggV1 now has a better description
- Updated comments to show number range of _mPDigg and _mPWBTC, and inlined each item

QSP-53 Potential problematic usage of the `_beforeTokenTransfer` hook

Severity: *Undetermined*

Status: Acknowledged

File(s) affected: `StabilizeDiggSett.sol`

Description: It is not clear what the `_beforeTokenTransfer` hook is supposed to do. Moreover, it doesn't follow the rule of calling super `super._beforeTokenTransfer`.

Recommendation: From the Rules of Hooks (https://docs.openzeppelin.com/contracts/3.x/extending-contracts), there's a few guidelines you should follow when writing code that uses hooks in order to prevent issues. They are very simple, but do make sure you follow them:

- Whenever you override a parent's hook, re-apply the virtual attribute to the hook. That will allow child contracts to add more functionality to the hook.
- Always call the parent's hook in your override using super. This will make sure all hooks in the inheritance tree are called: contracts like ERC20Pausable rely on this behavior.

Update: From the developers -- No change as this was cloned from BadgerSett reference code.

Automated Analyses

Slither

- Slither warns that `BaseStrategy.want` is not initialized in this contract. Although all derived contracts initialize the state variable, it may be less error-prone to initialize the variable in `__BaseStrategy_init`. Many of the return values of external calls to the UniswapV2Router are not checked, however the results of the function calls are often implicitly checked by calling `token.balanceOf(address(this))`.

Mythril

- Mythril warned of several uses of `tx.origin`; this was also noted in the Best Practices section.

Code Documentation

1. In `contracts/badger-timelock/vesting/SingleTokenVestingNonRevocable.sol` on L12, the comment "Optionally revocable by the owner." does not appear relevant to any functionality of the contract.
2. In `contracts/badger-timelock/SmartVesting.sol`, the comment on L25: "// address beneficiary, uint256 start, uint256 cliffDuration, uint256 duration, bool revocable" could likely be removed. In particular, "bool revocable" does not appear relevant to the code.
3. In `contracts/badger-geyser/RewardsFaucet.sol`, the comment block on L14-24 starts with "Digg Rewards Faucet", and should be updated for the general `RewardsFaucet`.
4. In `contracts/badger-geyser/BadgerGeyser.sol` on L58-61, the docs discuss 3 parameters but function `initialize` has 5 parameters.
5. Regarding `contracts/badger-geyser/BadgerGeyser.sol`, it should be noted in user documentation that there is no `unstakeFor` analog to `stakeFor`. Users should ensure that if the `stakedFor` `user` address is not an externally-owned account, then it will have the functionality to call `unstakeFor` at a later time.

After the 2nd tranche:

1. In `contracts/badger-sett/Sett.sol` on L285 we have: "Add blockLock to transfers, users cannot transfer tokens in the same block as a deposit or withdrawal." This is currently allowed if the transfer happens first in the block, so it should technically read: "... in the same block after a deposit or withdrawal."

2. In `contracts/badger-sett/Sett.sol`, the comment on `deposit` and `depositAll` should also mention that they are callable by approved addresses.
3. In `contracts/badger-sett/Sett.sol` on L214, it is not clear why "Permissionless operation" is stated, since the function is restricted to `_onlyAuthorizedActors`.
4. In `contracts/badger-sett/Sett.sol`, the docs state that it is "Used to swap any borrowed reserve over the debt limit to liquidate to 'token'", but the only thing it does here is to transfer to `controller` a non-`token` ERC20 in any arbitrary amount.
5. In `contracts/badger-sett/Controller.sol` on L187, "Wtihdraw" should be "Withdraw".
6. In `contracts/badger-sett/strategies/Swapper.sol` on L45, the comment should say "sushiswap" instead of "Uniswap". The same issue exists on L91.
7. In `contracts/badger-sett/strategies/BaseStrategy.sol` on L157, the comment "...to Withdraw partial funds..." does not seem appropriate here for the `withdrawAll` function.
8. In `contracts/badger-sett/strategies/sushi/StrategySushiBadgerWbtc.sol` on L173, "hrvest" should be "harvest".
9. In `contracts/badger-sett/Controller.sol`, it is not immediately clear what a `converter` refers to (e.g., uniswap?). Inline documentation should clarify this.
10. In `contracts/badger-sett/Controller.sol`, the comment block on L155-156 appears to be incorrectly copied from L148-149.
11. In `contracts/badger-sett/strategies/BaseStrategy.sol`, it is not clear why `withdrawalMaxDeviationThreshold` is set to 50.
12. In `contracts/badger-sett/strategies/BaseStrategy.sol`, the docs on L195-196 should use the "@notice" format used elsewhere.
13. In `strategies/harvest/StrategyHarvestMetaFarm.sol`, it does not list changes from versions 1.0 and 1.1 as in other contracts. This also occurs in `strategies/sushi/StrategySushiBadgetWbtc.sol`.
14. In `strategies/harvest/StrategyHarvestMetaFarm.sol` on L200, "unproessed" should be "unprocessed".
15. In `strategies/harvest/StrategyHarvestMetaFarm.sol`, the comment in tend: "simple do nothing besides emit an event" may not be true if `IERC20Upgradeable(farm).balanceOf(address(this))` is non-zero before the function call.
16. In `strategies/harvest/StrategyHarvestMetaFarm.sol`, the function `_fromHarvestVaultTokens` is not commented unlike the similar `_toHarvestVaultTokens` above.
17. In `strategies/sushi/StrategySushiBadgetWbtc.sol` on L196, "Withdrawl" should be "Withdrawal". This also occurs in the other sushiswap strategy contracts.
18. In `contracts/digg-core/UFragments.sol`, due to lack of specification or documentation, we cannot verify if `DECIMALS`, `SCALED_SHARES_EXTRA_DECIMALS`, or `MAX_FRAGMENTS_SUPPLY` are valid.
19. In `contracts/digg-core/UFragments.sol`, the comment on L218 states that it returns false otherwise, but the function instead reverts.
20. In `contracts/digg-core/UFragments.sol`, the comment block for `transferFrom` does not declare `@return` unlike other functions.
21. In `contracts/digg-core/UFragments.sol`, the functions `increaseAllowance` and `decreaseAllowance` should probably document the fact that if `subtractedValue > oldValue`, then it drops to 0.

As of commit `6bc6d41` we noted the following:

1. Price functions appear to be normalized to 18 decimal places, however this should be explicitly commented in the code.

Adherence to Best Practices

1. Favor using the explicit `uint256` instead of `uint` throughout.
2. In `contracts/badger-sett/SettAccessControlDefended.sol`, the function `SettAccessControlDefended._defend` is declared to return a `bool` but has no return statement. Further, the function uses `tx.origin`; consider using OpenZeppelin's Address.isContract.
3. In `contracts/badger-geyser/BadgerGeyser.sol`, the variable `MAX_PERCENTAGE = 100` is not used.
4. In the two `contracts/badger-geyser/*RewardsFaucet.sol` contracts, it is not clear why the events `RewardAdded` and `RewardDurationUpdated` are needed, as these fields are included in `NotifyRewardsAmount`.
5. In `contracts-reference/yAffiliateWrapperV2.sol`, the conditional on L120, `if (_diff > 0)` can be removed, as the `_bal > bal` check above ensures `_diff = _bal - bal` will be non-zero.
6. In `contracts-reference/yAffiliateWrapperV2.sol`, the `safe32` function is not used and could be removed.
7. In `contracts/badger-geyser/BadgerGeyser.sol` on L39, `EnumerableSetUpgradeable.AddressSet distributionTokens;` has no explicit visibility.
8. In `contracts/badger-geyser/BadgerGeyser.sol` on L44-45, it is not clear if there is any benefit to indexing timestamp in the `Staked` and `Unstaked`, given that `blockNumber` is also included.
9. In `contracts/badger-geyser/BadgerGeyser.sol`, certain functions such as `_onlyAfterStart` and `_onlyAdmin` could be modifiers instead.
10. In `contracts/badger-geyser/BadgerTree.sol`, the function `encodeClaim` could be inherited from `claimEncoder`.
11. In `contracts/badger-geyser/BadgerTree.sol` on L142, "specifiedrewards" should be "specified rewards".
12. The contract `contracts/badger-timelock/SimpleTimelockWithVoting.sol` could inherit from `contracts/badger-timelock/SimpleTimelock.sol`.
13. In `contracts-reference/yAffiliateWrapperV2.sol` on L79, `decimals` should be defined as a `uint8` according to the ERC20 specification.
14. In `contracts-reference/yAffiliateWrapperV2.sol`, the function `currentContribution` should use SafeMath's `.div` function, as a division-by-zero could occur if `totalSupply == 0`.
15. In `contracts-reference/yAffiliateWrapperV2.sol`, there is no explicit visibility for `affiliateTokens`, `isTokenAffiliate`, `tokenAffiliates`, and `isAffiliate`.
16. In `contracts-reference/yAffiliateWrapperV2.sol`, it is likely that `_affiliates` and `_yAffiliateTokens` have visibility set wrongly because they have getter functions (`yAffiliateTokens()`, `affiliates()`). These should likely change from public to internal given the underscore in the name.

After the 2nd tranche:

1. In `contracts/badger-sett/BadgerRewardsManager.sol` on L83 of `_onlyApprovedSetts`, the `== true` is not needed.
2. In `contracts/badger-sett/BadgerRewardsManager.sol`, there are inconsistent error messages in the `_only*` functions. For example, `_onlyAdmin` has the message "onlyAdmin" whereas `_onlyDistributor` uses "DISTRIBUTOR_ROLE".
3. The contract `contracts/badger-sett/BadgerRewardsManager.sol` inherits from `ReentrancyGuardUpgradeable` and `PausableUpgradeable` but does not use either

contract's modifiers.

4. In `contracts/badger-sett/Controller.sol`, the function `getExpectedReturn` does not have a consistent naming scheme for using underscores for parameters.
5. In `contracts/badger-sett/Controller.sol`, `withdrawAll` should check that `_token != address(0)` as done on L124.
6. In `contracts/badger-sett/Controller.sol`, there is a minor mismatched naming convention between `inCaseTokensGetStuck` and `inCaseStrategyTokenGetStuck` -- the first uses "Tokens" whereas the second uses "Token".
7. Several contracts such as `contracts/badger-sett/SettAccessControl.sol` inherit from helper contracts such as `Initializable` or `Pausable`, but this functionality is not used. For example, base contracts that inherit from `Initializable` should contain an `initialize` function, rather than relying on derived contracts to have this functionality.
8. In `contracts/badger-sett/SettAccessControl.sol`, the functions `setStrategist`, `setKeeper`, and `setGovernance` should likely emit an event.
9. In `contracts/badger-sett/strategies/BaseStrategy.sol`, it is not clear why the function `deposit` does not simply reject if `_want <= 0`.
10. The `_add_max_liquidity*` functions defined in multiple contracts should check that `token0 != token1`.
11. In `contracts/badger-sett/strategies/BaseStrategy.sol`, the `_only*` functions could be defined as modifiers.
12. In `strategies/harvest/StrategyHarvestMetaFarm.sol`, the event emitted on L292 says `Harvest(0, block.number)` but it seems that 0 should be replaced with `totalFarmHarvested` or `farmToRewards`.
13. In `strategies/sushi/StrategySushiDiggWbtcLpOptimizer.sol`, the `pid` is not explicitly declared unlike other sushiswap strategy contracts.
14. In `contracts/digg-core/UFragments.sol`, there are unused "deprecated" parameters on L53-54 that should be removed.
15. In `contracts/digg-core/UFragments.sol`, it is not clear why the variables on L81-82 are public.
16. In `contracts/digg-core/UFragments.sol`, the `TOTAL_SHARES` comments on L73 state that it uses a value that fits in a uint128 but then uses `MAX_UINT256`.
17. In the three sushiswap strategies such as `strategies/sushi/StrategySushiDiggWbtcLpOptimizer.sol`, in `_withdrawSome`, the actual amount withdrawn is more accurately `_postWant.sub(_preWant)` as `_postWant > _amount` is plausible.

As of commit `6bc6d41` we noted the following:

1. Several functions such as `_onlyAnyAuthorizedParties` could be declared as modifiers.
2. Functions such as `_burn` implicitly make contract logic checks in utility functions such as `SafeMath.sub`. It would be more readable to have checks such as `require(_balances[account]>=amount)` within `_burn` itself.
3. In `StabilizeStrategyDiggV1.sol` on L1229-1238, there are multiple hardcoded token addresses; consider using constants.
4. The function `_onlyAnyAuthorizedParties` is not used anywhere and could be removed.
5. In `StabilizeStrategyDiggV1.sol`, there is redundant code on L1355-1374 that could be abstracted into a helper function.

Test Results

Test Suite Results

The test suite script appears to fail launching a `ganache` instance. Logs are provided below.

```
Launching 'ganache-cli --port 8545 --gasLimit 12000000 --accounts 100 --hardfork istanbul --mnemonic brownie --fork https://mainnet.infura.io/v3/$WEB3_INFURA_PROJECT_ID --chainId 1'...
INTERNALERROR> Traceback (most recent call last):
INTERNALERROR>   File "/Users/ezulkosk/audits/badger-system/venv/lib/python3.7/site-packages/_pytest/main.py", line 240, in wrap_session
INTERNALERROR>     session.exitstatus = doit(config, session) or 0
INTERNALERROR>   File "/Users/ezulkosk/audits/badger-system/venv/lib/python3.7/site-packages/_pytest/main.py", line 295, in _main
INTERNALERROR>     config.hook.pytest_collection(session=session)
INTERNALERROR>   File "/Users/ezulkosk/audits/badger-system/venv/lib/python3.7/site-packages/pluggy/hooks.py", line 286, in __call__
INTERNALERROR>     return self._hookexec(self, self.get_hookimpls(), kwargs)
INTERNALERROR>   File "/Users/ezulkosk/audits/badger-system/venv/lib/python3.7/site-packages/pluggy/manager.py", line 93, in _hookexec
INTERNALERROR>     return self._inner_hookexec(hook, methods, kwargs)
INTERNALERROR>   File "/Users/ezulkosk/audits/badger-system/venv/lib/python3.7/site-packages/pluggy/manager.py", line 87, in <lambda>
INTERNALERROR>     firstresult=hook.spec.opts.get("firstresult") if hook.spec else False,
INTERNALERROR>   File "/Users/ezulkosk/audits/badger-system/venv/lib/python3.7/site-packages/pluggy/callers.py", line 208, in _multicall
INTERNALERROR>     return outcome.get_result()
INTERNALERROR>   File "/Users/ezulkosk/audits/badger-system/venv/lib/python3.7/site-packages/pluggy/callers.py", line 80, in get_result
INTERNALERROR>     raise ex[1].with_traceback(ex[2])
INTERNALERROR>   File "/Users/ezulkosk/audits/badger-system/venv/lib/python3.7/site-packages/pluggy/callers.py", line 187, in _multicall
INTERNALERROR>     res = hook_impl.function(*args)
INTERNALERROR>   File "/Users/ezulkosk/audits/badger-system/venv/lib/python3.7/site-packages/_pytest/main.py", line 306, in pytest_collection
INTERNALERROR>     session.perform_collect()
INTERNALERROR>   File "/Users/ezulkosk/audits/badger-system/venv/lib/python3.7/site-packages/_pytest/main.py", line 522, in perform_collect
INTERNALERROR>     hook.pytest_collection_finish(session=self)
INTERNALERROR>   File "/Users/ezulkosk/audits/badger-system/venv/lib/python3.7/site-packages/pluggy/hooks.py", line 286, in __call__
INTERNALERROR>     return self._hookexec(self, self.get_hookimpls(), kwargs)
INTERNALERROR>   File "/Users/ezulkosk/audits/badger-system/venv/lib/python3.7/site-packages/pluggy/manager.py", line 93, in _hookexec
INTERNALERROR>     return self._inner_hookexec(hook, methods, kwargs)
INTERNALERROR>   File "/Users/ezulkosk/audits/badger-system/venv/lib/python3.7/site-packages/pluggy/manager.py", line 87, in <lambda>
INTERNALERROR>     firstresult=hook.spec.opts.get("firstresult") if hook.spec else False,
INTERNALERROR>   File "/Users/ezulkosk/audits/badger-system/venv/lib/python3.7/site-packages/pluggy/callers.py", line 203, in _multicall
INTERNALERROR>     gen.send(outcome)
INTERNALERROR>   File "/Users/ezulkosk/audits/badger-system/venv/lib/python3.7/site-packages/brownie/test/managers/runner.py", line 258, in pytest_collection_finish
INTERNALERROR>     brownie.network.connect(CONFIG.argv["network"])
INTERNALERROR>   File "/Users/ezulkosk/audits/badger-system/venv/lib/python3.7/site-packages/brownie/network/main.py", line 50, in connect
INTERNALERROR>     rpc.launch(active["cmd"], **active["cmd_settings"])
INTERNALERROR>   File "/Users/ezulkosk/audits/badger-system/venv/lib/python3.7/site-packages/brownie/network/rpc.py", line 136, in launch
INTERNALERROR>     raise RPCProcessError(cmd, uri)
INTERNALERROR> brownie.exceptions.RPCProcessError: Unable to launch local RPC client.
INTERNALERROR> Command: ganache-cli
INTERNALERROR> URI: http://127.0.0.1:8545
```

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

5e1f736b73011bd95ff24ee8d65c4545ebce51afe0f6989a606fba32c5fc90a7 ./contracts/digg-core/UFragmentsPolicy.sol

2b3d5d20b4337b16dd9965529a1f337c0d4842288492e4025c00f28942c94dfd ./contracts/digg-core/Orchestrator.sol

2261b50b74acbdbf0f64ff3ede692011eb6fb83e0ac8e6d9af81ef211fba6f6f ./contracts/digg-core/UFragments.sol

ea6b6d6a803759c2c786147748ae14a0128548795d7b5b29fe268bc8570526c9 ./contracts/digg-core/lib/SafeMathInt.sol

5c6e056051acd09cd87db055fda46b833f23debad96eb1bcf50c889f151b2493 ./contracts/digg-core/lib/UInt256Lib.sol

37351dcf0da2af72b999a0385288870e260008466bfdc9806ee216d75433e9dc ./contracts/digg-core/mocks/MockUFragmentsPolicy.sol

d1cb5e8d48be47ffd09ad175e71a6c1017f10953d42dc08c584bd51177bfde62 ./contracts/digg-core/mocks/ConstructorRebaseCallerContract.sol

d485b5f758e1419c9b9c617a2720956cc1f3658157097236d475b42247c9a87d ./contracts/digg-core/mocks/SafeMathIntMock.sol

7da06c7f2fd4d529144e80b2e87610a46fa0042fd98ada6b7461c0ea50de5f4d ./contracts/digg-core/mocks/RebaseCallerContract.sol

a96b8f4efb819c83af0325b726fd907b0b391a4b4df86cbc477eee10e8784680 ./contracts/digg-core/mocks/UInt256LibMock.sol

383acce84457d91fbac5f5497dd548f40d0890c14dc81964db5e877f18089b42 ./contracts/digg-core/mocks/MockUFragments.sol

932d8752575b2641e8757bf91e3a5cd9313c9c0421e172fbb56d93789c81016f ./contracts/digg-core/mocks/MockOracle.sol

9f770d3f07d48c58036ae273f0d127f189847b81e26c892e011569926d25e62e ./contracts/digg-core/mocks/MockDownstream.sol

c6c027c46e3be287b922cd42ce6f2d94d25f03c924ba9077735f4c55ffab74f1 ./contracts/badger-claw/ExpiringMultiPartyCreator.sol

a0d5af1d1b92857bc17171dc1d08ac25ac58fd362e454ee2b5bd939be4ddae6b ./contracts/badger-claw/ExpiringMultiParty.sol

12342c71e24faacdd6e4ebe1ed83eeb13f12b1911922606fcba732374e60aede ./contracts/badger-sett/Sett.sol

f5af289aa8b07428455da552b9ba2cda5868cf1fbdff4b5e22df19b8e7dee116 ./contracts/badger-sett/SettV1.sol

4592f9eb8e41b62267298237c58015ded8f4434066652864cc84c212f7414d08 ./contracts/badger-sett/Controller.sol

d0ed1e83ccb7c5344a0d4c5223c9b77f2ee851ad7654ad9aca1001dc8c68ad68 ./contracts/badger-sett/SettAccessControl.sol

7427cb78c86f7acaf6f9d9ff074dd6827f2020f48c54f4ffacf0b71d6e8f3856 ./contracts/badger-sett/SettAccessControlDefended.sol

a971e1117a9503b81d71eab5f7ebc3e8181240f866b2f075da31992613aa8c05 ./contracts/badger-sett/BadgerRewardsManager.sol

79ac609c3389dee3cfc162577aa0287b66a1e668164ec47a4674cdfad640f8b8 ./contracts/badger-sett/UnlockScheduler.sol

498263495accff103662f502034ea68a0bf5eeee16384075a745085ea480228b ./contracts/badger-sett/DiggSett.sol

ca928fe5df97799828aa39a076e265111c3522b6472468cffe768f0e1c04ebfd ./contracts/badger-sett/strategies/BaseStrategy.sol

96338d72a14c7feedf9ce0a4dc575a3001f423a154f68ca2c7f86fa52245dc59 ./contracts/badger-sett/strategies/Swapper.sol

feb42b1c45b9952aff7101d27c8820beb5a2a95053a56250ffc42dd4fa9abdd4 ./contracts/badger-sett/strategies/PancakeSwapper.sol

73281bc7b9a11157b20518cce7f03b01bc8a659520f8f5810fbd6024573fb0cf ./contracts/badger-sett/strategies/BaseStrategySwapper.sol

9f4f3cdc7f586579166295ca4f25d7eff3f7b0a7aa8b7c70dfc5bfd7dca6728d ./contracts/badger-sett/strategies/badger/StrategyBadgerRewards.sol

1978d76154aa49fc9f9cc6ea119b2f9733c46dfd445993c99d3ab4f35e7b7387 ./contracts/badger-sett/strategies/pickle/StrategyPickleMetaFarm.sol

d4fdf38dccabc15aba8dabe54e199fc483649bcba7b9aa15df45b9e0f8338bda ./contracts/badger-sett/strategies/sushi/StrategySushiDiggWbtcLpOptimizer.sol

3fd6bfed1a839c215c92cdc1aad179bb243308ea2d6dafa963651088f0c6fd2e ./contracts/badger-sett/strategies/sushi/StrategySushiBadgerWbtc.sol

043e0ebe73f7db7a58caa5da1c9f2c1bd07e0413bb873844a6467b45fca3cfc2 ./contracts/badger-sett/strategies/sushi/StrategySushiLpOptimizer.sol

8b04955c10b39a30ef854e64a3847df3b827a5b214220c9d98fea9ef4eb6cb67 ./contracts/badger-sett/strategies/uni/StrategyDiggLpMetaFarm.sol

9c8ea7974717d715a2d645f65934cb8b16a32ea5f27e3c68a68fd8cb8f4fb99f ./contracts/badger-sett/strategies/uni/StrategyBadgerLpMetaFarm.sol

cac6db8750aed9c598336709c6413e05f07985f32563313763b624d7afdb4af0 ./contracts/badger-sett/strategies/digg/StrategyDiggRewards.sol

1843ff46f59213399a5ac7da139b4ef890e3a1603f127cb8dde622fd31171d05 ./contracts/badger-sett/strategies/harvest/StrategyHarvestMetaFarm.sol

eb17a5383c5a8b5ad16c2a79d467ff7af83fa8bcfea5cf9c49c1dbf1c7a03acf ./contracts/badger-sett/strategies/curve/StrategyCurveGaugeBase.sol

3754452191e3e46eecd7df59ce7b8f636af9fe28e80c76ea7916037d45ab86b5 ./contracts/badger-sett/strategies/curve/StrategyCurveGaugeSbtcCrv.sol

c6aa85e4f1e9b1e1c45618837337be5a3f609475b009192344636eecdea3d344 ./contracts/badger-sett/strategies/curve/StrategyCurveGaugeRenBtcCrv.sol

367d51ccaeb7651c17547e687d2fac6e45852f9f6632272de142bb7685ea5be1 ./contracts/badger-sett/strategies/curve/StrategyCurveGaugeTbtcCrv.sol

eedd02004769ba323cb79846bfc54adf2bdd9c3c15a55a8ded89bbb32d1f205e ./contracts/badger-sett/strategies/pancake/StrategyPancakeLpOptimizer.sol

2f4d6dbbd2707a74aacbdc5457a1e797c84dab820a0a9ebb0af97164cf92e28b ./contracts/badger-bridge/BadgerBridgeAdapter.sol

fed20f5d64da1fd8e3f920755929f72933d37fd72279e2402d0a5b7ba006a701 ./contracts/badger-bridge/mocks/Gateway.sol

5daf758aec6e4392edfcd2bb93f511bd3ffbafec3ffe9ae917ca462a13ff8867 ./contracts/badger-hunt/DiggDistributor.sol

dfddd27c399787127259881bf87b4fcf3ffed4c3c2bac247ed832aa4c94d0a77 ./contracts/badger-hunt/BadgerHunt.sol

5df951f1d572a6da278c230ae200193f2feb73a8baf3f4e9df6b01cb9257435f ./contracts/badger-hunt/DiggSeeder.sol

0ed819889cf13b79bdaa2d6be8c334ed0656b57bb699f2396d9254391e9441c0 ./contracts/badger-hunt/AirdropDistributor.sol

b4f2da5b3eb142be6a728486de6449f5364635c8d0d03b0ecbb3c3e3c459052b ./contracts/badger-hunt/HoneypotMeme.sol

7006ea802f407d9408943b6259092a0a5f9192bc9ede4aa9e08420ef05d7397c ./contracts/badger-hunt/MerkleDistributor.sol

c5b845f9f97e97959179b2cc55983abf8d0322bb24bf1d10fc8a98773e99a5a4 ./contracts/digg-oracles/ConstantOracle.sol

3f9f85a152b1c4c641d196cce499be7c7181c3b713581a44d5e77d3f966dd3c9 ./contracts/digg-oracles/MedianOracle.sol

e5b56d0d83cb6f03135d56f6dede0a9d7bded93b4452d4403f865bf2c87734b8 ./contracts/digg-oracles/lib/Select.sol

137ff3b87a7cd4d0b749f329aec0f375f0b6cfcac00ebccf69576bd3ca43f87f ./contracts/digg-oracles/test/DynamicOracle.sol

8b542b1d1fe8f98661ce198a8077ce0e4f1335253c3594b0bdc24c9c0adbc097 ./contracts/digg-oracles/mocks/SelectMock.sol

67c0c171d669620d8ec03bd188026e99a8fc7d5d957e8c13b1684fe664b57d48 ./contracts/mock/MockToken.sol

5e3e3594706f1a0b902885ac0729d4545b58338d727a6a5c486a54a7a3e96fbc ./contracts/mock/Ping.sol

fa5d858a73c7f3d1fb07ab1dab0a103b0c4e10fa0348414817c8d7be58e3deea ./contracts/mock/ForceEther.sol

ff72b04e79fa2c052b3e455be64b222c73eefb91b140b1f82f4b8e62b387c86b ./contracts/badger-swap/SwapStrategyRouter.sol

87934e874bdc8ed5626a2ebbcf9758e00b501e1cf5e849d9bccb25db4be25050 ./contracts/badger-swap/strategies/CurveSwapStrategy.sol

5719b122d82fec7d8642a280b3b33e0d9ef74622aef6477db13e4bcad27f18ca ./contracts/badger-timelock/OtcEscrow.sol

18ff15ed243cd4a55574234f55fcb921ae58050e34455280c45c26c6a8a10464 ./contracts/badger-timelock/SimpleTimelock.sol

12c4c4b30c7c6ca3004c94a21b34dbfc3d17fec89a4469599fd996d56ab2cc14 ./contracts/badger-timelock/SmartTimelock.sol

702ce34b44b89a28cee5fb01e26963e8cb58d2c57dfc28b9cfdb194118daf944 ./contracts/badger-timelock/GovernanceTimelock.sol

674aa48d4ed5f56ede5d55ecd284d5b9c7d04aa1e347adfc2fea5ec82a3df39b ./contracts/badger-timelock/SimpleTimelockWithVoting.sol

67b213c710375973906b950dd911678e95fe0ca50ad2c5bbe74ca316aa4f26a4 ./contracts/badger-timelock/Executor.sol

d934398ac63a970fad4351fc8b388bd02a24bdc4f32678a9a648af53682a8115 ./contracts/badger-timelock/SmartVesting.sol

af7e4f912de6e99cc61e540d8d6e5e2247f7baa0caba95fc12f1423e6d5a75ee ./contracts/badger-timelock/vesting/SingleTokenVestingNonRevocable.sol

a7f88cf4b2e9321aa6e8b90f2923f6bfc367d18d1f598243d7b0706c95c1f5c2 ./contracts/badger-timelock/mock/StakingMock.sol

63c547d2d6c53a5d35480432fb1602806b67381a99894766afcf4fcd832fdc5c ./contracts/badger-timelock/mock/TokenRelease.sol

e6a7d80ca67c8244736fdd6403ac09b2829db89a42255f2aa766ebd30a7b6788 ./contracts/badger-timelock/mock/TokenGifter.sol

b0ad3f09953606274de048fbd313b365ad9d460ed1aa6c36bc5f32b9510bddd1 ./contracts/badger-timelock/mock/EthGifter.sol

6fc388e928c121768a4b9647e5647a253f7ce0452afebccce768da04b95c0b6e ./contracts/badger-geyser/RewardsEscrow.sol

eef0df5fa02222368af57cfde2feb9c5f0d020a45ce97d3371bc4408bf6124c4 ./contracts/badger-geyser/StakingRewardsSignalOnly.sol

5230b809fa3e49594c186f0dccc07a275a1800eb03f0f2611e082efc040a6a41 ./contracts/badger-geyser/BadgerTreeV2.sol

d659d5eaa666398f1bb5d40d51639c97502a8737a37e0bbca2d3d5ad6282330a ./contracts/badger-geyser/BadgerTree.sol

45b7be18db4a250c7d7c7c5f31dd035c347c8b2bbfe59857e256925e91342ff7 ./contracts/badger-geyser/RewardsFaucet.sol

6ce55521d7198a7a99b40d0a65ae60fd67accbfb15975717a5ac8903d3dda5d0 ./contracts/badger-geyser/StakingRewards.sol

10119d72e0824c89725145e860edae9b41a57ac2379365dad544e3baad47f7e5 ./contracts/badger-geyser/ContributorLogger.sol

87d004b54e79e001c7209e832e733fe0195687e5094762715a85e04efa564113 ./contracts/badger-geyser/ClaimEncoder.sol

c03686d8357e6f0d1059f312144675813b61196be0bfed250b220a625899e3da ./contracts/badger-geyser/BadgerGeyser.sol

d78572571c326bf3065125f8090a8dc752060bd96557c72c331eb9ab7d106b8b ./contracts/badger-geyser/DiggRewardsFaucet.sol

596ea113b5019e1ec13dc135a7b7d00de316e51d4b21bd9a4a657ceca3739733 ./contracts/yearn/BaseSimpleWrapperUpgradeable.sol

7c526f450843c71b59b0aa26bc0d27ac30371cef180e01c05904294174042f46 ./contracts/yearn/VipCappedGuestListWrapperUpgradeable.sol

95f734e51c9bf963cc0c6557af63168eb39b09a973991e519435565cd1e30b79 ./contracts/yearn/VipCappedGuestList.sol

60d128b8056ac1f1ba4ff0fe36f09c79e54ce850c1c3c74786d8d726435e5deb ./contracts/yearn/BaseWrapperUpgradeable.sol

99928d2dd5923e568c22a6dc8b8e92dabf44c1cf9eb031fbbee9b43a1ca31868 ./contracts/yearn/AffiliateTokenGatedUpgradeable.sol

28bc9e81bd36c33ccd4389ee78f2eb9931487b8d431976890d7a404ef139cc76 ./contracts/yearn/SimpleWrapperGatedUpgradeable.sol

Tests

7ef02db23c90b74db8f1be94e06e4401c45861a13a8c4f7b1e5852c9849c4358 ./tests/test_recorder.py

e5b91c368c769a2c5577f59b7db28092764e88da50742837679b14ab96248ac4 ./tests/conftest.py

556ac0dbb9915cdd0a408440a269f96f772f5aadc3dd5161c6d8e931469c325c ./tests/StrategyConfig.py

e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855 ./tests/__init__.py

aabf8b434a91292a340ddb7b1bb5f143ae041c5a349ad7bbe31c00989950dee0 ./tests/helpers.py

3a0e67d1170b9f3f2adddb19df0fe685b980c20460a736132e5cfa6083d51258 ./tests/airdrop/test_gitcoin_round_8_flow.py

0826878874bebc4d0d7dc828faf3ca5a9da61b0a3c22f1cfb372799a27e4d39c ./tests/sett/test_claw_simulation.py

615dd6c66ce45b810e2e1156dc5b273cacad14924a95d2d582f78640bb9794e7 ./tests/sett/test_bsc_flow_live.py

33286992b214c5c3989fb1d7f82a3332cb304fb266bc38dc6049ec00963b1cbe ./tests/sett/test_upgrade_crv_setts.py

61c3de6ca59e627bf71f52de9108cd81214304274ec946ecac9ea02dac7d1f1d ./tests/sett/test_strategy_flow.py

ca72089095794df3f77514cbc07e0be5a93d4fd0ae266793da47beaec3db85aa ./tests/sett/test_bsc_flow.py

af0e76362c0d2a4685be155867938a7b10afe773fef4b74dd3db533d1f9a3ec5 ./tests/sett/test_digg_simulation.py

82da59af881f6d5e39ca0be626e926b7fd844f1dac871316685621caf385e60e ./tests/sett/test_simulation.py

61822665f533ffaac643b24bd0c8b194f7f33b49ead133dbfac7e7ebf6404d25 ./tests/sett/test_digg_strategy_flow.py

9b89515f5aaab837c96fef39765ea97c4faf40a943accc5bee1121ebfb764caf ./tests/sett/test_strategy_permissions.py

d6694b7c5dac5bbb31fcb1bfd78a81a48c28d753b4a07c3d11cec2dd34623c37 ./tests/sett/fixtures/SushiClawUSDCMiniDeploy.py

0e235f265d027755d8658228dce3376c8acd7967ca88bd6dbadcb0d42bc1e7e3 ./tests/sett/fixtures/BadgerRewardsMiniDeploy.py

ac9482c7ee983c0f049baed4375d8e1836535c47de64c6d2f80f9cd0798efedc ./tests/sett/fixtures/BadgerLpMetaFarmMiniDeploy.py

f4c97641b745c126988526b211149cf5d48895dd678a5a462d24ee69b6ac6248 ./tests/sett/fixtures/UniDiggWbtcLpMiniDeploy.py

e1a99250712be40f1b47e6de5fcb129eb63800908d26d77ca93d600f2391c0f1 ./tests/sett/fixtures/SushiDiggWbtcLpOptimizerMiniDeploy.py

c8bdfc6d5b1b89f13e893fb0ee83ba90274b1e789a3af0748958f1e0f23e0eae ./tests/sett/fixtures/__init__.py

2e52621c6cb836056dd080fb885d8eca9232502f988beb4ba5e4e45fbcf93325 ./tests/sett/fixtures/SushiBadgerLpOptimizerMiniDeploy.py

1977e5fed1f402b335b67bf43b56fa29fdbb52c2d5c073aae4c56bd60e7cda88 ./tests/sett/fixtures/DiggSettMiniDeployBase.py

```
c05d8eb418615a36864d2c9fae1623f0657db905d574902f238d5cef4a793d8e ./tests/sett/fixtures/SushiBadgerWBtcMiniDeploy.py
e33d6028afeedef58d29a94ab1446d4d8f6f4104991b9d5dc556a22e82295b3f ./tests/sett/fixtures/HarvestMetaFarmMiniDeploy.py
4978cb2ca818a6fe2efdbd96e487f924fcacc7142e67f50849e2a9837e569348 ./tests/sett/fixtures/DiggRewardsMiniDeploy.py
64960175f146be7844a9ea83bc0bdb8db883d8953db3e24ca338f5fe0799c2be ./tests/sett/fixtures/SettMiniDeployBase.py
5f7d83b7679e900e5c06f918c803db48589f8668c53884c487bbb9345e482203 ./tests/sett/fixtures/PancakeMiniDeploy.py
1c18fd02d06ad00fb0767940ec2aaf7f47be202df995371f7413e961c216d829 ./tests/sett/fixtures/CurveGaugeSBtcMiniDeploy.py
4ec49578893332456c85bdb77aefbee4a1382af551eb0fcceff22044f669e304 ./tests/sett/fixtures/CurveGaugeRenBtcMiniDeploy.py
de6faaad25faf858570038865b9178ff3bc32761f73749c1dea89f3a14755452 ./tests/sett/fixtures/CurveGaugeTBtcMiniDeploy.py
6f0a3107ba9a9a4574c497b083d0a022d9d8272f5e690d1c6d96aa836a9f896c ./tests/claw/test_claw.py
c3f957bd93bc41cc6cb814839b3140f486931da12e5d6111d30a518de43895c4 ./tests/smart_timelock/fixtures.py
7fefbc8fa96b525748f4aa14d53d1225794a2c368fe34147dae08083d67fef7a ./tests/smart_timelock/vesting/test_vesting_unit.py
af9c2aca6393d662122fb657d75092d333f9f4c9dead51aba8ad3f17dccd24b7 ./tests/smart_timelock/timelock/test_access_control.py
a290f88792ad81b31303501323d902153241fb5a53024bb4c0cd3493002a02bf ./tests/smart_timelock/timelock/test_staking.py
45c032bfb9566bc847b6d201212b3d5d4fe54ca9dc20d979193dbfa91398032a ./tests/smart_timelock/timelock/test_eth_transfers.py
75ffc72b4b79b85791f266d4a312f4860e6c8ea354c7a9205f351c72c0932922 ./tests/rewards_tree/test_rewards_flow.py
0d22d84b026c82d70f44b90937f0715e5cdf4216442ffd674c4f867fddcb0374 ./tests/rewards_tree/test_rewards_tree.py
5fe0863eae0adacc9683c8404d6f1373e646f2b2ad61284bf8c1635e8a126908 ./tests/rewards_tree/test_rewards_multi_cycle.py
e1957ab61531c1c808ef26caf44f4b27a6c8bf1f9c1b2f7e47fb45952f7c40f1 ./tests/honeypot/test_meme_honeypot.py
c85721f3a5b50be46d11f7eb3e4cb4fc7f0a9a36478feb71a446f352e67fd967 ./tests/integration/test_fund_release.py
066e389e19071a0f103ec5ff2e6137faa80023f5d9ae13484079bce21aa1c97a ./tests/integration/test_upgrades.py
6c1fd7ceea1d026f20957ead1c46f64a9d490cb5c332d4188861bd6b620f37cd ./tests/integration/test_deploy.py
bc80a3684f3a0f6c3ed88877383ef657b3bd4ade462c6b9d23144ef58ae6e983 ./tests/bridge-integration/adapter_test.js
6872eca76764d6a4c77722ee265a6d59f32855158547c114463f0ec954076e43 ./tests/bridge/test_bridge.py
a477afe3c3e3bdfa90d337e41fb12112c174e802e58b717e9a04dc20c031051d ./tests/hunt/test_badger_hunt.py
```

Changelog

- 2021-03-04 - Initial report
- 2021-03-22 - Updated report for second tranche
- 2021-04-20 - Updated report for commit 6bc6d41
- 2021-05-10 - Updated report for commit 5bcc709

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.

Timeliness of content

The content contained in the report is current as of the date appearing on the report and is subject to change without notice, unless indicated otherwise by Quantstamp; however, Quantstamp does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following publication.

Notice of confidentiality

This report, including the content, data, and underlying methodologies, are subject to the confidentiality and feedback provisions in your agreement with Quantstamp. These materials are not to be disclosed, extracted, copied, or distributed except to the extent expressly authorized by Quantstamp.

Links to other websites

You may, through hypertext or other computer links, gain access to web sites operated by persons other than Quantstamp, Inc. (Quantstamp). Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites' owners. You agree that Quantstamp are not responsible for the content or operation of such web sites, and that Quantstamp shall have no liability to you or any other person or entity for the use of third-party web sites. Except as described below, a hyperlink from this web site to another web site does not imply or mean that Quantstamp endorses the content on that web site or the operator or operations of that site. You are solely responsible for determining the extent to which you may use any content at any other web sites to which you link from the report. Quantstamp assumes no responsibility for the use of third-party software on the website and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any outcome generated by such software.

Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Blockchain technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.