



June 14th 2021 — Quantstamp Verified

Naos-Formation

This smart contract audit was prepared by Quantstamp, the protocol for securing smart contracts.

Executive Summary

Type	Ethereum						
Auditors	Sung-Shine Lee, Research Engineer Kacper Bqk, Senior Research Engineer Ed Zulkoski, Senior Security Engineer						
Timeline	2021-05-19 through 2021-06-11						
EVM	Muir Glacier						
Languages	Solidity						
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review						
Specification	None						
Documentation Quality	<div><div></div></div> Medium						
Test Quality	<div><div></div></div> Medium						
Source Code	<table><tr><th>Repository</th><th>Commit</th></tr><tr><td><a href="#">NAOS-Formation</a></td><td>19f4967</td></tr><tr><td>None</td><td>c12527</td></tr></table>	Repository	Commit	<a href="#">NAOS-Formation</a>	19f4967	None	c12527
Repository	Commit						
<a href="#">NAOS-Formation</a>	19f4967						
None	c12527						

Total Issues	16 (13 Resolved)
High Risk Issues	0 (0 Resolved)
Medium Risk Issues	3 (2 Resolved)
Low Risk Issues	5 (4 Resolved)
Informational Risk Issues	7 (6 Resolved)
Undetermined Risk Issues	1 (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 the engagement, a high level overview of the system was provided to the auditing team, but the specification isn't complete in the technical level. We have identified a total of 16 issues, ranging from Medium to Informational Risk. Overall, the system would benefit from adding checks to the return values of external protocol and user inputs. Notably, the system uses deprecated Chainlink api and doesn't check validity of the data, which brings the risk of stale oracle price. The lack of check also made it possible to add the same adapter to the system multiple times. Lastly, due to the existence of `flushActiveVault()` and how it's used, we recommend to be cautious when integrating with external protocols and make sure the assumptions hold. We recommend addressing all issues before using the code in production.

Update: As per [c125272](#), Naos team provided fixes and acknowledgements for the issues. QSP-1 is partially fixed as it is still possible to migrate to the same adapter, if the adapter was not the last adapter.

ID	Description	Severity	Status
QSP-1	Possible to migrate to the same adapter	^ Medium	Mitigated
QSP-2	Unchecked Return Value	^ Medium	Acknowledged
QSP-3	Oracle price could be stale	^ Medium	Fixed
QSP-4	Unchecked function arguments	^ Low	Fixed
QSP-5	Privileged Roles and Ownership	^ Low	Acknowledged
QSP-6	<code>IsHealthy()</code> does not follow the spec completely	^ Low	Fixed
QSP-7	<code>forceTransmute</code> does not check if <code>msg.sender</code> is a new user	^ Low	Fixed
QSP-8	Intended revert not present in <code>flushActiveVault()</code>	^ Low	Fixed
QSP-9	Unlocked Pragma	o Informational	Fixed
QSP-10	Using experimental <code>ABIEncoderV2</code>	o Informational	Fixed
QSP-11	Gas-usage for-loop concerns	o Informational	Fixed
QSP-12	<code>YearnVaultAdapter.deposit</code> has no access-control	o Informational	Fixed
QSP-13	Economic attack vector exists due to <code>flush()</code>	o Informational	Acknowledged
QSP-14	Unnecessary <code>flush()</code>	o Informational	Fixed
QSP-15	TimeToken contract does not seem to be used	o Informational	Fixed
QSP-16	Default <code>flushActivator</code> amount depends on token decimals	? Undetermined	Fixed

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](#) v0.8.0
- [Mythril](#) v0.2.7

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 Possible to migrate to the same adapter

Severity: *Medium Risk*

Status: Mitigated

File(s) affected: [Formation.sol](#)

Description: Using `migrate()` to invoke `_updateActiveVault()` may add add the same adapter multiple times. This would cause accounting errors between Formation and the adapter. While the `migrate()` function is limited to the governance, checks should be in place to prevent incorrect behaviour.

Recommendation: We suggest adding relevant checks to remove such possibilities.

Update: As of [c125272](#), only the last adapter is checked, however formation could still erroneously add previously added adapter other than the last one.

QSP-2 Unchecked Return Value

Severity: *Medium Risk*

Status: Acknowledged

File(s) affected: [YearnVaultAdapter.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 relevant function is checked.

- in [YearnVaultAdapter.sol](#), L63, 73: return value not checked

Recommendation: Add relevant checks, especially when interacting with external protocols, to ensure integration works properly.

Update: Naos response: "Return value from yearn is not a simple True/False, therefore unable to make a clear cut decision based on the response. Emergency mode can be set to stop the depositing."

QSP-3 Oracle price could be stale

Severity: *Medium Risk*

Status: Fixed

Description: [Formation.sol](#), L661: the price fetching function Chainlink API (`latestAnswer()`) has been declared deprecated by Chainlink and may be outdated. Furthermore, `latestAnswer()` was used but its return value `updatedAt` and `answeredInRound` are not checked, and therefore the data could be arbitrarily old.

Recommendation: Use the current Chainlink API and check that `updatedAt` and `answeredInRound` are recent.

Update: Naos team has updated the deprecated function.

QSP-4 Unchecked function arguments

Severity: *Low Risk*

Status: Fixed

File(s) affected: [YearnVaultAdapter.sol](#), [Transmuter.sol](#), [StakingPools.sol](#), [Formation.sol](#)

Description: Some arguments in functions are not checked against zero. This leaves space for human-error and allows the arguments to be zero, which typically would simply revert, but in some cases it would result in transferring tokens to the `0x0` address and burning them.

1. `StakingPools.constructor` does not check that `_reward` is non-zero.
2. `StakingPools._token` does not check that `_token` is non-zero.
3. `Transmuter.constructor` does not check that `_NToken` and `_Token` are non-zero.
4. `Transmuter.setTransmutationPeriod` does not check that `newTransmutationPeriod` is non-zero.
5. `YearnVaultAdapter.constructor` should check that `_vault` and `_admin` are non-zero.
6. `Formation.constructor` should check that `_token` and `_xtoken` are non-zero.

**Recommendation:** We suggest adding relevant requirement statement to ensure the validity of function arguments.

## QSP-5 Privileged Roles and Ownership

**Severity:** *Low Risk*

**Status:** Acknowledged

**File(s) affected:** `NAOSToken.sol`, `NToken.sol`, `Transmuter.sol`, `StakingPools.sol`, `Formation.sol`, `TimeToken.sol`

**Description:** Smart contracts will often have `owner` variables to designate the person with special privileges to make modifications to the smart contract. The governance of the system holds significant amount of power. For example, through governance, it is possible to mint infinite amount of NTokens and Naos.

1. In `NAOSToken` and `TimeToken.sol`, the minter may mint tokens arbitrarily.
2. In `StakingPools`, the governance may set arbitrary minting rates and proportions for each pool.
3. In `Transmuter.distribute`, any whitelisted address can transfer tokens into the buffer from any other address that has approved the `Transmuter` contract.
4. In `Formation.sol`, the governance can set important addresses such as the `oracle`, `transmuter` or `rewards`, which in the worst case could steal all funds.

**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. Relevant mitigation plans should be communicated to the users too.

**Update:** Naos response: "After deployment, the governance role will be transferred to the NAOS multisig account which consists of NAOS core team members and trusted community members. The setting will follow the decision of the NAOS governance process. "

## QSP-6 `IsHealthy()` does not follow the spec completely

**Severity:** *Low Risk*

**Status:** Fixed

**File(s) affected:** `CDP.sol`

**Description:** The documentation states that "A CDP is healthy if its collateralization ratio is greater than the global collateralization limit.". However, in `L59` of `CDP.sol`, it returns true if the ratio is equal as well.

**Recommendation:** Align the specification with the implementation.

**Update:** The code is deemed correct and comments were updated accordingly.

## QSP-7 `forceTransmute` does not check if `msg.sender` is a new user

**Severity:** *Low Risk*

**Status:** Fixed

**Description:** A fresh address could invoke `forceTransmute()`, in which case the bookkeeping handled by `checkIfNewUser()` will be incorrect.

**Recommendation:** Add a `checkIfNewUser(msg.sender)` check to `forceTransmute()`.

**Update:** Naos has fixed the issue as recommended.

## QSP-8 Intended revert not present in `flushActiveVault()`

**Severity:** *Low Risk*

**Status:** Fixed

**Description:** `Formation.sol`, `L440` states that "This function reverts if an emergency exit is active.". The intended revert is not present. Considering this is present for emergency, the lack of this revert may allow the attacker to forcefully push funds into the external protocol. If the external protocol was vulnerable and under attack, this exposes the funds in Formation to the attack as well.

**Recommendation:** Add the revert intended in the comments.

**Update:** Naos has fixed the issue as recommended.

## QSP-9 Unlocked Pragma

**Severity:** *Informational*

**Status:** Fixed

**Description:** Every Solidity file specifies in the header a version number of the format `pragma solidity (^)0.4.*`. 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.

**Update:** Naos has fixed the issue as recommended. The version is locked to 0.6.12.

## QSP-10 Using experimental ABIEncoderV2

**Severity:** *Informational*

**Status:** Fixed

**Description:** The contracts declare the experimental feature ABIEncoderV2. As it is an experimental feature, it is more likely to include unknown bugs in the implementation.

**Recommendation:** Check if the ABIEncoderV2 feature is required for the contracts. Remove the experimental feature if it is not needed.

**Update:** Naos has fixed the issue as recommended.

## QSP-11 Gas-usage for-loop concerns

*Severity: Informational*

**Status:** Fixed

**File(s) affected:** `MultiSigWallet.sol` , `StakingPools.sol`

**Description:** 1. The function `MultiSigWallet.getTransactionIds` iterates over the entire transaction history on L384, even though only transaction with indices in the range `[from, to)` are under consideration. If the wallet has a long transaction history, this may cause an unnecessary amount of storage load (**SLOAD**) operations, and the function may fail due to gas limits. Although the function is declared `view`, `web3` clients may not execute functions beyond the gas-limit, and external contracts may try to invoke the function.

1. `StakingPools._updatePools` could fail if too many pools are added.

**Recommendation:** In `MultiSigWallet.getTransactionIds`, change the first for-loop to only iterate from indices `from` to `to`. In general, perform gas analysis to ensure loops will not exceed acceptable bounds in practice.

**Update:** Naos has fixed the issue as recommended and updated comments accordingly.

## QSP-12 `YearnVaultAdapter.deposit` has no access-control

*Severity: Informational*

**Status:** Fixed

**File(s) affected:** `YearnVaultAdapter.sol`

**Description:** If a user accidentally deposits to this contract instead of the expected `Formation` contract, they will not be credited for their funds.

**Recommendation:** Restrict the function such that it is only callable from the relevant contracts.

**Update:** Naos has fixed the issue as recommended.

## QSP-13 Economic attack vector exists due to `flush()`

*Severity: Informational*

**Status:** Acknowledged

**File(s) affected:** `Formation.sol`

**Description:** As "flush" pushes the funds from Formation to the underlying vault, this forces an exchange of assets from underlying token to the external vault's share token. To be safe from economic exploits, it is essential that the share price of the vault cannot be manipulated in a single transaction.

**Recommendation:** Always verify the behaviour of share price when adding an external protocol. The share price of the external protocol has to be monotonically increasing and cannot be manipulated in a single transaction.

**Update:** Naos response: "The team would choose the DeFi project(s) carefully as the underlying vault. Also, the sentinel could set emergency mode to stop the depositing."

## QSP-14 Unnecessary `flush()`

*Severity: Informational*

**Status:** Fixed

**File(s) affected:** `Formation.sol`

**Description:** The purpose of `flush()` is to push the funds from `Formation` to the external protocol. Thus, it is unclear why `flush()` is activated in `withdraw()` when the specified `amount` is higher than the predefined threshold. When users withdraw more, the action tends to empty the Formation contract and there is no need to activate `flush()` at all.

**Recommendation:** We suggest reviewing the logic and verifying if this is the desired behavior.

**Update:** Naos has removed the activation `flush()` when the user withdraws a higher amount than the predefined threshold.

## QSP-15 TimeToken contract does not seem to be used

*Severity: Informational*

**Status:** Fixed

**File(s) affected:** `TimeToken.sol`

**Description:** The TimeToken contract does not seem to be used in the system. It is also not present in the tests.

**Recommendation:** Remove if the contract is not necessary.

**Update:** The contract has been removed.

## QSP-16 Default `flushActivator` amount depends on token decimals

*Severity: Undetermined*

**Status:** Fixed

**File(s) affected:** `Formation.sol`



**Description:** The `constructor` sets `flushActivator = 100000 ether` with the added comment "change for non 18 digit tokens". It is not clear which tokens will be used, but if their decimal values are significantly different than 18, this amount will be too small or large.

**Recommendation:** Clarify which tokens will be used. Ensure that `setFlushActivator` is used if `token.decimals != 18`.

**Update:** The `flushActivator` argument is added as an input parameter for the user to adjust the value. A recommended value added in the comment for user’s reference.

Adherence to Specification

- The behaviour stated in `Formation.sol`, `L440`is not enforced.

Code Documentation

1. In `MultiSigWallet.sol` on L324, "filers" should be "filters".
2. In `StakingPools.sol` on L187, there is a comment "// FIXME", however it is unclear if any issue still exists here.
3. The comment block in `Transmuter.sol` on L12-34 appears copied from one of the OpenZeppelin `ERC20`-related contracts, and does not help describe `Transmuter` itself. In general, this file requires significantly more inline documentation.
4. In `Formation.sol` on L157, "movemetns" should be "movements".
5. The comment on `Formation.sol#666` should say "Checks that caller is an eoa." (remove "not").
6. The function `CDP.isHealthy` has the comment "A CDP is healthy if its collateralization ratio is greater than the global collateralization limit.", but the function checks for `>=`.

Adherence to Best Practices

- the modifier `Transmuter.checkIfNewUser` changes state due to assignments. The name would suggest that it is a query, although it is not.
- `StakingPools.sol#124`, `Formation.sol#231`, it's unclear why `_pendingGovernance` is necessary
- `StakingPools.sol#187`: `fixme`
- `Formation.sol`, L191-194, L394: commented out code
- `IChainlink.sol` uses deprecated Chainlink API.
- Favor using `uint256` instead of `uint`.
- `MultiSigWallet.isConfirmed` should explicitly return false after the for-loop.
- In `NToken.sol`, `setWhitelist` and `setBlacklist` should emit events to facilitate tracking state variables.
- In `Formation.sol`, the commented code on L191-194,394 should be removed.\* . In `Formation.sol` on 325, the `require` statement should have an error message.
- The internal function `Formation._expectCaller` is not used anywhere in the project and could be removed.

Test Results

Test Suite Results

```
Formation
constructor
  when token is the zero address
    ✓ reverts
  when xtoken is the zero address
    ✓ reverts
  when governance is the zero address
    ✓ reverts
  when sentinel is the zero address
    ✓ reverts
  when flushActivator is set to zero
    ✓ reverts
update Formation addys and variables
set governance
  when caller is not current governance
    ✓ reverts
  when caller is current governance
    ✓ reverts when setting governance to zero address
    ✓ updates rewards (77ms)
set transmuter
  when caller is not current governance
    ✓ reverts
  when caller is current governance
    ✓ reverts when setting transmuter to zero address
    ✓ updates transmuter
set rewards
  when caller is not current governance
    ✓ reverts
  when caller is current governance
    ✓ reverts when setting rewards to zero address
    ✓ updates rewards
set performance fee
  when caller is not current governance
    ✓ reverts
  when caller is current governance
    ✓ reverts when performance fee greater than maximum
    ✓ updates performance fee
set collateralization limit
  when caller is not current governance
    ✓ reverts
  when caller is current governance
    ✓ reverts when performance fee less than minimum
    ✓ reverts when performance fee greater than maximum
```

```
    ✓ updates collateralization limit (44ms)
vault actions
migrate
  when caller is not current governance
    ✓ reverts
  when caller is current governance
    when adapter is zero address
      ✓ reverts
    when adapter is same as current active vault
      ✓ reverts
    when adapter token mismatches
      ✓ reverts
    when conditions are met
      ✓ increments the vault count
      ✓ sets the vaults adapter
recall funds
  from the active vault
    ✓ reverts when not an emergency, not governance, and user does not have permission to recall funds from active vault
    ✓ governance can recall some of the funds (207ms)
    ✓ governance can recall all of the funds (149ms)
  in an emergency
    ✓ anyone can recall funds (109ms)
    ✓ after some usage (238ms)
  from an inactive vault
    ✓ anyone can recall some of the funds to the contract (53ms)
    ✓ anyone can recall all of the funds to the contract (54ms)
  in an emergency
    ✓ anyone can recall funds (71ms)
flush funds
  when the Formation is not initialized
    ✓ reverts
  when there is at least one vault to flush to
    when there is one vault
      ✓ flushes funds to the vault
    when there are multiple vaults
      ✓ flushes funds to the active vault
deposit and withdraw tokens
  ✓ deposited amount is accounted for correctly (41ms)
  ✓ deposits token and then withdraws all (105ms)
  ✓ reverts when withdrawing too much (38ms)
  ✓ reverts when cdp is undercollateralized (78ms)
  ✓ deposits, mints, repays, and withdraws (169ms)
  ✓ deposits 5000 DAI, mints 1000 nUSD, and withdraws 3000 DAI (123ms)
flushActivator
  ✓ deposit() flushes funds if amount >= flushActivator (67ms)
  ✓ deposit() does not flush funds if amount < flushActivator (48ms)
repay and liquidate tokens
  ✓ repay with dai reverts when nothing is minted and transmuter has no nUsd deposits
  ✓ liquidate max amount possible if trying to liquidate too much (173ms)
  ✓ liquidates funds from vault if not enough in the buffer (278ms)
  ✓ liquidates the minimum necessary from the formation buffer (238ms)
  ✓ deposits, mints nUsd, repays, and has no outstanding debt (180ms)
  ✓ deposits, mints, repays, and has no outstanding debt (139ms)
  ✓ deposits, mints nUsd, repays with nUsd and DAI, and has no outstanding debt (193ms)
  ✓ deposits and liquidates DAI (193ms)
mint
  ✓ reverts if the Formation is not whitelisted
is whitelisted
  ✓ reverts if the Formation is blacklisted (57ms)
  ✓ reverts when trying to mint too much
  ✓ reverts if the ceiling was breached (49ms)
  ✓ mints successfully to depositor (99ms)
flushActivator
  ✓ mint() flushes funds if amount >= flushActivator (203ms)
  ✓ mint() does not flush funds if amount < flushActivator (182ms)
harvest
  ✓ harvests yield from the vault (101ms)
  ✓ sends the harvest fee to the rewards address (98ms)
  ✓ does not update any balances if there is nothing to harvest (49ms)

NaosToken
  ✓ grants the admin role to the deployer
  ✓ grants the minter role to the deployer
mint
  when unauthorized
    ✓ reverts
  when authorized
    ✓ mints tokens

StakingPools
  when reward token address is the zero address
    ✓ reverts
set governance
  ✓ only allows governance
  when caller is governance
    ✓ prevents getting stuck
    ✓ sets the pending governance
    ✓ updates governance upon acceptance
    ✓ emits GovernanceUpdated event
set reward rate
  ✓ only allows governance to call
  when caller is governance
    ✓ updates reward rate
    ✓ emits RewardRateUpdated event
create pool
  ✓ only allows governance to call
  when caller is governance
    ✓ only allows none-zero token address
    ✓ emits PoolCreated event
  when reusing token
    ✓ reverts
set pool reward weights
  ✓ only allows governance to call
  when caller is governance
    ✓ reverts when weight array length mismatches
  with one pool
    ✓ updates the total reward weight
    ✓ updates the reward weights
  with many pools
    ✓ updates the total reward weight
    ✓ updates the reward weights (38ms)
deposit tokens
  with no previous deposits
    ✓ increments total deposited amount
    ✓ increments deposited amount
    ✓ transfers deposited tokens
    ✓ does not reward tokens
  with previous deposits
    ✓ increments total deposited amount
    ✓ increments deposited amount
    ✓ transfers deposited tokens
withdraw tokens
  with previous deposits
    ✓ decrements total deposited amount
    ✓ decrements deposited amount
    ✓ transfers deposited tokens
claim tokens
  with deposit
    ✓ mints reward tokens
    ✓ clears unclaimed amount
  with multiple deposits
    ✓ mints reward tokens
    ✓ clears unclaimed amount
get stake unclaimed amount
  with deposit
    ✓ properly calculates the balance
  with multiple deposits
    ✓ properly calculates the balance

Transmuter
  when NToken is the zero address
```

```
    ✓ reverts
when token is the zero address
    ✓ reverts
stake()
    ✓ stakes 1000 nUsd and reads the correct amount (39ms)
    ✓ stakes 1000 nUsd two times and reads the correct amount (65ms)
unstake()
    ✓ reverts on depositing and then unstaking balance greater than deposit
    ✓ deposits and unstakes 1000 nUSD (60ms)
    ✓ deposits 1000 nUSD and unstaked 500 nUSD (62ms)
distributes correct amount
    ✓ deposits 100000 nUSD, distributes 1000 DAI, and the correct amount of tokens are distributed to depositor (76ms)
    ✓ two people deposit equal amounts and recieve equal amounts in distribution (123ms)
    ✓ deposits of 500, 250, and 250 from three people and distribution is correct (180ms)
transmute() claim() transmuteAndClaim()
    ✓ transmutes the correct amount (135ms)
    ✓ burns the supply of nUSD on transmute() (111ms)
    ✓ moves DAI from pendingdivs to inbucket upon staking more (111ms)
    ✓ transmutes and claims using transmute() and then claim() (147ms)
    ✓ transmutes and claims using transmuteAndClaim() (137ms)
    ✓ transmutes the full buffer if a complete phase has passed (153ms)
    ✓ transmutes the staked amount and distributes overflow if a bucket overflows (596ms)
transmuteClaimAndWithdraw()
    ✓ has a staking balance of 0 nUSD after transmuteClaimAndWithdraw()
    ✓ returns the amount of nUSD staked less the transmuted amount
    ✓ burns the correct amount of transmuted nUSD using transmuteClaimAndWithdraw()
    ✓ successfully sends DAI to owner using transmuteClaimAndWithdraw()
exit()
    ✓ transmutes and then withdraws nUSD from staking
    ✓ transmutes and claimable DAI moves to realised value
    ✓ does not claim the realized tokens
forceTransmute()
    ✓ User 'depositor' has nUSD overfilled, user 'minter' force transmutes user 'depositor' and user 'depositor' has DAI sent to his address (149ms)
    ✓ User 'depositor' has nUSD overfilled, user 'minter' force transmutes user 'depositor' and user 'minter' overflow added inbucket (138ms)
    ✓ you can force transmute yourself (157ms)
    ✓ you can force transmute yourself even when you are the only one in the transmuter (106ms)
    ✓ reverts when you are not overfilled (51ms)
Multiple Users displays all overfilled users
    ✓ returns userInfo (113ms)
distribute()
    ✓ must be whitelisted to call distribute (39ms)
    ✓ increases buffer size, but does not immediately increase allocations (90ms)
userInfo()
    ✓ distribute increases allocations if the buffer is already > 0 (93ms)
    ✓ increases buffer size, and userInfo() shows the correct state without an extra nudge (97ms)
```

137 passing (2m)



## Code Coverage

While there are already plenty of tests in the test suite, the coverage data shows that the security could improve from adding more tests. Specifically, the coverage is critically low on the important contracts (e.g. [CDP.sol](#)). We recommend to add more tests to cover all the statements and branch.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
<b>contracts/</b>	70.78	55.09	64.23	69.66	
Formation.sol	86.47	72.62	82.05	86.78	... 667,668,669
MultiSigWallet.sol	0	0	0	0	... 393,394,395
MultiSigWalletWithTimelock.sol	0	0	0	0	... 115,123,124
NAOSToken.sol	100	100	100	100	
NToken.sol	88.46	75	84.62	89.66	89,100,101
StakingPools.sol	87.64	83.33	83.33	86.81	... 282,311,312
Transmuter.sol	93.65	69.57	91.67	93.94	... 446,448,450
<b>contracts/adapters/</b>	100	50	100	100	
YearnVaultAdapter.sol	100	50	100	100	
<b>contracts/interfaces/</b>	100	100	100	100	
IChainlink.sol	100	100	100	100	
ICurveMetaFactory.sol	100	100	100	100	
IDetailedERC20.sol	100	100	100	100	
IERC20Burnable.sol	100	100	100	100	
IMintableERC20.sol	100	100	100	100	
ITransmuter.sol	100	100	100	100	
IVaultAdapter.sol	100	100	100	100	
IYearnController.sol	100	100	100	100	
IYearnVault.sol	100	100	100	100	
IyVaultV2.sol	100	100	100	100	
<b>contracts/libraries/</b>	91.3	64.29	80	91.3	
FixedPointMath.sol	91.3	64.29	80	91.3	29,39
<b>contracts/libraries/formation/</b>	75	50	90	75	
CDP.sol	53.13	43.75	85.71	53.13	... 3,84,87,104
Vault.sol	96.88	100	92.31	96.88	98
<b>contracts/libraries/pools/</b>	89.29	100	80	89.29	
Pool.sol	85	100	75	85	112,121,122
Stake.sol	100	100	100	100	
<b>contracts/mocks/</b>	60.47	50	61.9	60.47	
ERC20Mock.sol	66.67	100	66.67	66.67	20
VaultAdapterMock.sol	100	100	100	100	
YearnControllerMock.sol	25	100	25	25	22,32,36
YearnVaultMock.sol	59.38	50	55.56	59.38	... 73,74,88,89
<b>All files</b>	<b>72.57</b>	<b>56.34</b>	<b>69.57</b>	<b>71.68</b>	

# 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

010d87f5b749174a18d32d577c37949b72ad1220ac1dc78205686418f7f231a5 ./contracts/mocks/YearnControllerMock.sol

d3689277b208ab3673fecc79d813ee3dc8b1d870943c136c6ec79b336311e622 ./contracts/mocks/VaultAdapterMock.sol

c64f648cfe9d272b9326f51e8a98a4f420315da2779c4e1085026a34c6cdfc0e ./contracts/mocks/ERC20Mock.sol

70ea3e5f36c0eee2a91541e552f69d7b9747a0c4b505a21fd42c2b8fef35368e ./contracts/mocks/YearnVaultMock.sol

3b57da08608868578027de8528ec77473f12cf27b28369505d69fc42a0450a8f ./contracts/Transmuter.sol

883511ce5a876498ae9d4efb92b001810bfb3b6d66abbf74c6ec0ff2b3b1f2ac ./contracts/NToken.sol

2ddead147290d78d8de785104378fa75e821715faac2cda7ffdfcb568c68892 ./contracts/StakingPools.sol

87103d310f77815f0889436bd18d748fee8094df16e47bcad540188e3d268fbe ./contracts/MultiSigWallet.sol

e42835e481637c651a99965c4950964283a2646d9e9dc8ccaf815c3912f3e50f ./contracts/libraries/pools/Pool.sol

7ceab35ebf7ce6ac1f19010d6208e6c23a3da703b0dee2a95956cec5e0d64b11 ./contracts/libraries/pools/Stake.sol

74bb52436690fa3ce30af2396d5714e2518ec5cb42aa8f6fba45bd15446c8cd5 ./contracts/libraries/formation/CDP.sol

a223043d557bb6a8d4097913cad5dbd9aded2666fcd9dbfb4a6844e02ccbea90 ./contracts/libraries/formation/Vault.sol

7fc3ca4d813aaf6211b850cfb3816eba4b4b40e7df1fb9c3b84610cc19d208e1 ./contracts/libraries/FixedPointMath.sol

774d3393fdfc226b005ce248e02aba440d2db0d51cfa9611f7b3f5e533d34651 ./contracts/MultiSigWalletWithTimelock.sol

c7e04d7da6b87930d12c9d456908132903d0bcc7b06f4977a593f8fe496fb50e ./contracts/adapters/YearnVaultAdapter.sol

c52158c891cb4c9b7a9cb9b1c24b611bb89be62b0dfeef7cb07d41cd09576aa ./contracts/NAOSToken.sol

6951d3153bf7563580eab2c6187b2c4fc94d6e5091f059de6e82a294e051199c ./contracts/Formation.sol

6cd1f951a3d8c2fb50f9a80ca8751f327173c3e89f322a40f6f56291ccc3608f ./contracts/interfaces/ICurveMetaFactory.sol

1d52f58d8414170b5079ea564c9ae5b4c8ef6b9b007b8881da47f419582ad63e ./contracts/interfaces/IERC20Burnable.sol

7f2ebe1fa159bdd88f979a6da1524a7d19fcaaa7b1caf2c3227da7891d1fd7c7 ./contracts/interfaces/IYearnVault.sol

b1afb5498f28599b468df335be07b2616a9bdde000b16f3d6a54f09ad93b0d22 ./contracts/interfaces/IYearnController.sol

8a785a54386f44d774fad87791e7bb0506ba4b9554346c59f7cc424aae5b3c41 ./contracts/interfaces/IyVaultV2.sol

e1a5a084aca37d6da955fc81743790f3766848ba34a615fbb3c3837d495b6cf2 ./contracts/interfaces/IMintableERC20.sol

d3f7a322fd28948786c95358182262e507fd3adcb418e298cb459f9d6e6bb10 ./contracts/interfaces/IChainlink.sol

851a30759b81adaa9138bdd8f65a255e87532bf8e8c13878c213b5d35f2efef2 ./contracts/interfaces/IDetailedERC20.sol

23dfc93a8801d4f8ae158292634d3e614a39748575d08ea2a08accb75342af3e ./contracts/interfaces/ITransmuter.sol

110da5ee996d6abc7f66a66a1495715adc7395fbeaa1718521f252b522a0b40d ./contracts/interfaces/IVaultAdapter.sol

### Tests

ba24f168908a0045631e50e357fc89e51d5fe4a019b0e93a24184c6d4a0d81b5 ./test/contracts/Formation.spec.ts

7917f298bccd1312efafb1883e1a2f58172d9827b45cd895f74c502c59989e9e ./test/contracts/StakingPools.spec.ts

cf4e99125c1fdcb206a257e01d0dfadcbc5f239dd5c322059f776345b1317675 ./test/contracts/NAOSToken.spec.ts

427ee05cfa97c00ef3ed97a85a57f72815c7d0aad1cd20a0b6ef762da0852613 ./test/contracts/Transmuter.spec.ts

0f683d627b09bf1fea3a59870a060156067519de76ffa85e1442b91aabe9e446 ./test/utills/helpers.ts

a51aec117e9dc151f05b2faab9dbe998642b957f798f044b1611c0b5283aae72 ./test/utills/ethereum.ts

# Changelog

- 2021-05-28 - Initial report

# About Quantstamp

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

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

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

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

## 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.

