Quantstamp Security Assessment Certificate

November 25th 2021 — Quantstamp Verified

NAOS-Formation Updates

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

QUANTSTAMP VERIFIED SECURITY CERTIFICATE

Executive Summary

Type	Ethereum
Reviewers	Ed Zulkoski, Senior Security Engineer Kacper Bąk, Senior Research Engineer
Timeline	2021-11-10 through 2021-11-24
EVM	London
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	None
Documentation Quality	Medium
Test Quality	Medium
Source Code	

Repository	Commit
NAOS-Formation	c1404ec

Total Issues	4 (4 Resolved)
High Risk Issues	0 (0 Resolved)
Medium Risk Issues	1 (1 Resolved)
Low Risk Issues	0 (0 Resolved)
Informational Risk Issues	3 (3 Resolved)
Undetermined Risk Issues	0 (0 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

This report pertains to the new changes to NAOS-Formation in the diff c125272...c1404ec since the previous audit report. The scope is limited to not include the following files: `MultiSigWallet.sol`, `MultiSigWalletWithTimelock.sol`, `NAOSToken.sol`, `NToken.sol`, `StakingPools.sol`, and `EllipsisVaultAdapter.sol`. One medium and three informational-level issues were noted in this diff.
Update: the team has addressed all the issues as of commit `a721545`.

ID	Description	Severity	Status
QSP-1	`amountOutMin` too low in `swapExactTokensForTokens` call	^ Medium	Fixed
QSP-2	Missing `ensureUserActionDelay` modifier	○ Informational	Fixed
QSP-3	Unchecked return values	○ Informational	Fixed
QSP-4	Unlocked Pragma	○ Informational	Fixed

Quantstamp Review 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 reviewing 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 security review.

Setup

Tool Setup:

- Slither v0.8.1

Steps taken to run the tools:

Installed the Slither tool: `pip install slither-analyzer` Run Slither from the project directory: `slither .`

Findings

QSP-1 `amountOutMin` too low in `swapExactTokensForTokens` call

Severity: Medium Risk

Status: Fixed

File(s) affected: `AlpacaVaultAdapter.sol`

Description: The call `uniV2Router.swapExactTokensForTokens(alpacaToken.balanceOf(address(this)), 0, path, address(this), block.timestamp + 800);` sets the `amountOutMin` value to 0. If the pool is out-of-balance or manipulated in an unfavorable way (possibly through a sandwich-attack), the swap may result in very few `bUSD` tokens being withdrawn.

Recommendation: Set `amountOutMin` to a higher value to mitigate such attacks, possible as some percentage of the `_amount` passed in, or as a separate user-defined parameter.

QSP-2 Missing `ensureUserActionDelay` modifier

Severity: Informational

Status: Fixed

File(s) affected: `TransmuterV2.sol`

Description: The related TransmuterB.sol contract utilizes an `ensureUserActionDelay` modifier, as implemented here. This guards functions `stake`, `transmute`, and `forceTransmute`, and may mitigate unforeseen attacks such as flash loan attacks.

Recommendation: Consider incorporating the `ensureUserActionDelay` modifier into `TransmuterV2`.

QSP-3 Unchecked return values

Severity: Informational

Status: Fixed

File(s) affected: `AlpacaVaultAdapter.sol`, `TransmuterV2.sol`, `YearnVaultAdapter.sol`, `Formation.sol`

Description: The following functions have return values that are not checked:

1. AlpacaVaultAdapter.withdraw(address,uint256) (contracts/adapters/AlpacaVaultAdapter.sol#124-133) ignores return value by busdToken.transfer(_recipient,busdToken.balanceOf(address(this))) (contracts/adapters/AlpacaVaultAdapter.sol#132)
2. AlpacaVaultAdapter.indirectWithdraw(address,uint256) (contracts/adapters/AlpacaVaultAdapter.sol#141-161) ignores return value by busdToken.transfer(_recipient,busdToken.balanceOf(address(this))) (contracts/adapters/AlpacaVaultAdapter.sol#160)
3. Formation._distributeToTransmuter(uint256) (contracts/Formation.sol#593-598) ignores return value by token.approve(transmuter,amount) (contracts/Formation.sol#594)
4. FormationV2._distributeToTransmuter(uint256) (contracts/FormationV2.sol#600-605) ignores return value by token.approve(transmuter,amount) (contracts/FormationV2.sol#601)
5. TransmuterV2._plantOrRecallExcessFunds() (contracts/TransmuterV2.sol#687-702) ignores return value by _activeVault.deposit(plantAmt) (contracts/TransmuterV2.sol#695)
6. TransmuterV2.migrateFunds(address) (contracts/TransmuterV2.sol#797-807) ignores return value by IERC20Burnable(Token).approve(migrateTo,migratableFunds) (contracts/TransmuterV2.sol#804)
7. AlpacaVaultAdapter.indirectWithdraw(address,uint256) (contracts/adapters/AlpacaVaultAdapter.sol#141-161) ignores return value by uniV2Router.swapExactTokensForTokens(alpacaToken.balanceOf(address(this)),0,path,address(this),block.timestamp + 800) (contracts/adapters/AlpacaVaultAdapter.sol#151-157)
8. YearnVaultAdapter.deposit(uint256) (contracts/adapters/YearnVaultAdapter.sol#63-65) ignores return value by vault.deposit(_amount) (contracts/adapters/YearnVaultAdapter.sol#64)
9. YearnVaultAdapter.withdraw(address,uint256) (contracts/adapters/YearnVaultAdapter.sol#73-75) ignores return value by vault.withdraw(_tokensToShares(_amount),_recipient) (contracts/adapters/YearnVaultAdapter.sol#74)

Recommendation: Check the return values for success on each of the above function calls.

Update: The team has fixed all but the last two unchecked statements. The team explained that 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-4 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.

Automated Analyses

Slither

1. Slither notes several ignored return values as noted in an issue above.
2. Several instances of multiplication before division occur of the form `amount.div(USDT_CONST).mul(USDT_CONST)`, however this appears to be intended semantics to normalize the amount values.

Code Documentation

1. In `VaultV2.sol`, the comment on L101 "Directly withdraw deposited funds from the vault." is incorrect for the associated function `indirectWithdraw`, and appears to be copy+pasted from `directWithdraw` above. **Update:** fixed.
2. The function `FormationV2.liquidate` returns two values `_withdrawnAmount`, and `_decreasedValue`. Docstrings should be added confirming the units of these values (i.e., they are expected to have the same decimal amount as `token`, not `xtoken`). **Update:** fixed.

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
    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 peformance 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
  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 (51ms)
        ✓ governance can recall all of the funds (49ms)
        in an emergency
          ✓ anyone can recall funds (63ms)
          ✓ after some usage (119ms)
      from an inactive vault
        ✓ anyone can recall some of the funds to the contract
        ✓ anyone can recall all of the funds to the contract
        in an emergency
          ✓ anyone can recall funds
    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
      ✓ deposits token and then withdraws all (43ms)
      ✓ reverts when withdrawing too much
      ✓ reverts when cdp is undercollateralized
      ✓ deposits, mints, repays, and withdraws (64ms)
      ✓ deposits 5000 DAI, mints 1000 nUSD, and withdraws 3000 DAI (49ms)
      flushActivator
        ✓ deposit() flushes funds if amount >= flushActivator
        ✓ deposit() does not flush funds if amount < flushActivator
    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 (63ms)
      ✓ liquidates funds from vault if not enough in the buffer (109ms)
      ✓ liquidates the minimum necessary from the formation buffer (100ms)
      ✓ deposits, mints nUsd, repays, and has no outstanding debt (64ms)
      ✓ deposits, mints, repays, and has no outstanding debt (46ms)
      ✓ deposits, mints nUsd, repays with nUsd and DAI, and has no outstanding debt (71ms)
      ✓ deposits and liquidates DAI (69ms)
    mint
      ✓ reverts if the Formation is not whitelisted
      is whiltelisted
        ✓ reverts if the Formation is blacklisted
        ✓ reverts when trying to mint too much
        ✓ reverts if the ceiling was breached
        ✓ mints successfully to depositor (40ms)
        flushActivator
          ✓ mint() does not flush funds if amount < flushActivator (75ms)
    harvest
      ✓ harvests yield from the vault (47ms)
      ✓ sends the harvest fee to the rewards address (41ms)
      ✓ does not update any balances if there is nothing to harvest

FormationV2
  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
    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 peformance 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
  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 (43ms)
      from an inactive vault
          ✓ anyone can recall some of the funds to the contract
    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
        ✓ deposits token and then withdraws all
        ✓ reverts when cdp is undercollateralized
        ✓ deposits, mints, repays, and withdraws (68ms)
        ✓ revert with withdraw too much tokens after repay with nUSD (76ms)
        ✓ deposits, mints, repays with USDT, and withdraws (70ms)
        ✓ revert with repay too much USDT (64ms)
        ✓ revert with withdraw too much tokens after repay with USDT (75ms)
        ✓ deposits 5000 USDT, mints 1000 nUSD, and withdraws 3000 USDT (54ms)
      flushActivator
          ✓ deposit() flushes funds if amount >= flushActivator
          ✓ deposit() does not flush funds if amount < flushActivator
    repay and liquidate tokens
        ✓ repay with USDT reverts when nothing is minted and transmuter has no nUsd deposits
        ✓ liquidate max amount possible if trying to liquidate too much (60ms)
        ✓ liquidates funds from vault if not enough in the buffer (137ms)
        ✓ liquidates the minimum necessary from the formation buffer (113ms)
        ✓ deposits, mints nUsd, repays, and has no outstanding debt (65ms)
        ✓ deposits, mints, repays, and has no outstanding debt (51ms)
        ✓ deposits, mints nUsd, repays with nUsd and USDT, and has no outstanding debt (68ms)
        ✓ deposits and liquidates USDT (61ms)
    mint
        ✓ reverts if the Formation is not whitelisted
      is whiltelisted
          ✓ reverts if the Formation is blacklisted
          ✓ reverts when trying to mint too much
          ✓ reverts if the ceiling was breached
          ✓ mints successfully to depositor (43ms)
    harvest
        ✓ harvests yield from the vault (52ms)
        ✓ sends the harvest fee to the rewards address (51ms)
        ✓ does not update any balances if there is nothing to harvest
  Alpaca vault
    test deposit/withdraw of ibBusd
        ✓ deposit/withdraw (52ms)
    test swapExactTokensForTokens of uniswapV2
        ✓ swapExactTokensForTokens 1:1 (48ms)
    test deposit/withdraw/harvest of alpacaStakingPool
        ✓ deposit/withdraw/harvest (111ms)
    from the active vault
        ✓ should work (129ms)
  Ellipsis vault
    test add_liquidity/remove_liquidity of 3es
        ✓ add_liquidity/remove_liquidity (54ms)
    test deposit/withdraw/harvest of ellipsisStakingPool
        ✓ deposit/withdraw/harvest (95ms)
    from the active vault
        ✓ should work (142ms)

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
  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 (68ms)
        ✓ 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
    ✓ stakes 1000 nUsd two times and reads the correct amount
  unstake()
    ✓ reverts on depositing and then unstaking balance greater than deposit
    ✓ deposits and unstakes 1000 nUSD
    ✓ deposits 1000 nUSD and unstaked 500 nUSD
  distributes correct amount
    ✓ deposits 100000 nUSD, distributes 1000 DAI, and the correct amount of tokens are distributed to depositor
    ✓ two people deposit equal amounts and recieve equal amounts in distribution (49ms)
    ✓ deposits of 500, 250, and 250 from three people and distribution is correct (65ms)
  transmute() claim() transmuteAndClaim()
    ✓ transmutes the correct amount (43ms)
    ✓ burns the supply of nUSD on transmute() (45ms)
    ✓ moves DAI from pendingdivs to inbucket upon staking more (45ms)
    ✓ transmutes and claims using transmute() and then claim() (59ms)
    ✓ transmutes and claims using transmuteAndClaim() (53ms)
    ✓ transmutes the full buffer if a complete phase has passed (63ms)
    ✓ transmutes the staked amount and distributes overflow if a bucket overflows (271ms)
  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 (51ms)
    ✓ User 'depositor' has nUSD overfilled, user 'minter' force transmutes user 'depositor' and user 'minter' overflow added inbucket (54ms)
    ✓ you can force transmute yourself (56ms)
    ✓ you can force transmute yourself even when you are the only one in the transmuter (44ms)
    ✓ reverts when you are not overfilled
  Multiple Users displays all overfilled users
    ✓ returns userInfo (46ms)
  distribute()
    ✓ must be whitelisted to call distribute
    ✓ increases buffer size, but does not immediately increase allocations
    userInfo()
      ✓ distribute increases allocations if the buffer is already > 0 (41ms)
      ✓ increases buffer size, and userInfo() shows the correct state without an extra nudge (40ms)

Transmuter
  when NToken is the zero address
    ✓ reverts
  when token is the zero address
    ✓ reverts
  stake()
    ✓ stakes 1000 nUsd and reads the correct amount
    ✓ stakes 1000 nUsd two times and reads the correct amount
    ✓ stakes 1000.123456789 nUsd two times and reads the correct amount
  unstake()
    ✓ reverts on depositing and then unstaking balance greater than deposit
    ✓ deposits and unstakes 1000 nUSD
    ✓ deposits 1000 nUSD and unstaked 500 nUSD
  distributes correct amount
    ✓ deposits 100000 nUSD, distributes 1000 USDT, and the correct amount of tokens are distributed to depositor (334ms)
    ✓ two people deposit equal amounts and recieve equal amounts in distribution (52ms)
    ✓ deposits of 500, 250, and 250 from three people and distribution is correct (70ms)
  transmute() claim() transmuteAndClaim()
    ✓ transmutes the correct amount (50ms)
    ✓ burns the supply of nUSD on transmute() (49ms)
    ✓ moves USDT from pendingdivs to inbucket upon staking more (49ms)
    ✓ transmutes and claims using transmute() and then claim() (50ms)
    ✓ transmutes and claims using transmuteAndClaim() (50ms)
    ✓ transmutes the full buffer if a complete phase has passed (67ms)
    ✓ transmutes the staked amount and distributes overflow if a bucket overflows (265ms)
  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 USDT to owner using transmuteClaimAndWithdraw()
  exit()
    ✓ transmutes and then withdraws nUSD from staking
    ✓ transmutes and claimable USDT 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 USDT sent to his address (62ms)
    ✓ User 'depositor' has nUSD overfilled, user 'minter' force transmutes user 'depositor' and user 'minter' overflow added inbucket (61ms)
    ✓ you can force transmute yourself (70ms)
    ✓ you can force transmute yourself even when you are the only one in the transmuter (40ms)
    ✓ reverts when you are not overfilled
  Multiple Users displays all overfilled users
    ✓ returns userInfo
  distribute()
    ✓ must be whitelisted to call distribute
    ✓ increases buffer size, but does not immediately increase allocations
    userInfo()
      ✓ distribute increases allocations if the buffer is already > 0 (41ms)
      ✓ increases buffer size, and userInfo() shows the correct state without an extra nudge

TransmuterUSDV2
  stake()
    ✓ stakes 1000 nUSD and reads the correct amount
    ✓ stakes 1000 nUsd two times and reads the correct amount
    ✓ stakes 1000.123456789 nUsd two times and reads the correct amount
  unstake()
    ✓ reverts on depositing and then unstaking balance greater than deposit
    ✓ deposits 1000.123456789 and unstakes 1000.123456 nUSD
    ✓ deposits 1000 nUSD and unstaked 500 nUSd
  distributes correct amount
    ✓ deposits 100000 nUSD, distributes 1000 USDT, and the correct amount of tokens are distributed to depositor
    ✓ two people deposit equal amounts and recieve equal amounts in distribution (58ms)
    ✓ deposits of 500, 250, and 250 from three people and distribution is correct (84ms)
  transmute() claim() transmuteAndClaim()
    ✓ transmutes the correct amount (53ms)
    ✓ burns the supply of nUSD on transmute() (60ms)
    ✓ moves USDT from pendingdivs to inbucket upon staking more (58ms)
    ✓ transmutes and claims using transmute() and then claim() (67ms)
    ✓ transmutes and claims using transmuteAndClaim() (57ms)
    ✓ transmutes the full buffer if a complete phase has passed (73ms)
    ✓ transmutes the staked amount and distributes overflow if a bucket overflows (270ms)
    ensureSufficientFundsExistLocally()
      transmuterPreClaimBal < claimAmount
        ✓ recalls enough funds to handle the claim request
        ✓ recalls enough funds to reach plantableThreshold
        ✓ recalls all funds from the vault if the vault contains less than plantableThreshold (66ms)
      transmuterPreClaimBal >= claimAmount
        ✓ does not recall funds from the vault if resulting balance is under plantableThreshold
  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() (63ms)
    ✓ successfully sends USDT to owner using transmuteClaimAndWithdraw()
  exit()
    ✓ transmutes and then withdraws nUSD from staking
    ✓ transmutes and claimable USDT 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 USDT sent to his address (75ms)
    ✓ User 'depositor' has nUSD overfilled, user 'minter' force transmutes user 'depositor' and user 'minter' overflow added inbucket (65ms)
    ✓ you can force transmute yourself (62ms)
    ✓ you can force transmute yourself even when you are the only one in the transmuter (49ms)
    ✓ reverts when you are not overfilled
  Multiple Users displays all overfilled users
    ✓ returns userInfo (67ms)
  distribute()
    ✓ must be whitelisted to call distribute
    ✓ increases buffer size, but does not immediately increase allocations (42ms)
    userInfo()
      ✓ distribute increases allocations if the buffer is already > 0 (42ms)
      ✓ increases buffer size, and userInfo() shows the correct state without an extra nudge (44ms)
    _plantOrRecallExcessFunds
      transmuterPostDistributeBal < plantableThreshold
        ✓ does not send funds to the active vault
        vault has funds before distribute()
          ✓ recalls funds from the active vault if they are available
          ✓ recalls the exact amount of funds needed to reach plantableThreshold
          ✓ does not recall funds if below by less than plantableMargin
      transmuterPostDistributeBal > plantableThreshold
        ✓ sends excess funds to the active vault
        ✓ sends the exact amount of funds in excess to reach plantableThreshold
        ✓ does not send funds if above by less than plantableMargin
      transmuterPostDistributeBal == plantableThreshold
        ✓ does nothing
  recall
    recallAllFundsFromVault()
      ✓ reverts when not paused
      ✓ reverts when not governance or sentinel
      ✓ recalls funds from active vault
      ✓ recalls funds from any non-active vault (55ms)
    recallFundsFromVault
      ✓ reverts when not paused
      ✓ reverts when not governance or sentinel
      ✓ recalls funds from active vault
      ✓ recalls funds from any non-active vault (56ms)
  harvest()
    ✓ harvests yield from the vault
  migrateFunds()
    ✓ reverts if anyone but governance tries to migrate
    ✓ reverts when trying to migrate to 0x0
    ✓ reverts if not in emergency mode
    ✓ reverts if there are not enough funds to service all open transmuter stakes
    ✓ sends all available funds to the new transmuter (64ms)

TransmuterV2
  stake()
    ✓ stakes 1000 nUSD and reads the correct amount
    ✓ stakes 1000 nUsd two times and reads the correct amount
  unstake()
    ✓ reverts on depositing and then unstaking balance greater than deposit
    ✓ deposits and unstakes 1000 nUSD
    ✓ deposits 1000 nUSD and unstaked 500 nUSd
  distributes correct amount
    ✓ deposits 100000 nUSD, distributes 1000 DAI, and the correct amount of tokens are distributed to depositor (39ms)
    ✓ two people deposit equal amounts and recieve equal amounts in distribution (58ms)
    ✓ deposits of 500, 250, and 250 from three people and distribution is correct (82ms)
  transmute() claim() transmuteAndClaim()
    ✓ transmutes the correct amount (55ms)
    ✓ burns the supply of nUSD on transmute() (54ms)
    ✓ moves DAI from pendingdivs to inbucket upon staking more (55ms)
    ✓ transmutes and claims using transmute() and then claim() (72ms)
    ✓ transmutes and claims using transmuteAndClaim() (63ms)
    ✓ transmutes the full buffer if a complete phase has passed (73ms)
    ✓ transmutes the staked amount and distributes overflow if a bucket overflows (257ms)
    ensureSufficientFundsExistLocally()
      transmuterPreClaimBal < claimAmount
        ✓ recalls enough funds to handle the claim request
        ✓ recalls enough funds to reach plantableThreshold
        ✓ recalls all funds from the vault if the vault contains less than plantableThreshold (67ms)
      transmuterPreClaimBal >= claimAmount
        ✓ does not recall funds from the vault if resulting balance is under plantableThreshold
  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 (61ms)
    ✓ User 'depositor' has nUSD overfilled, user 'minter' force transmutes user 'depositor' and user 'minter' overflow added inbucket (59ms)
    ✓ you can force transmute yourself (60ms)
    ✓ you can force transmute yourself even when you are the only one in the transmuter (46ms)
    ✓ reverts when you are not overfilled
  Multiple Users displays all overfilled users
    ✓ returns userInfo (48ms)
  distribute()
    ✓ must be whitelisted to call distribute
    ✓ increases buffer size, but does not immediately increase allocations
    userInfo()
      ✓ distribute increases allocations if the buffer is already > 0 (42ms)
      ✓ increases buffer size, and userInfo() shows the correct state without an extra nudge (42ms)
    _plantOrRecallExcessFunds
      transmuterPostDistributeBal < plantableThreshold
        ✓ does not send funds to the active vault
        vault has funds before distribute()
          ✓ recalls funds from the active vault if they are available
          ✓ recalls the exact amount of funds needed to reach plantableThreshold
          ✓ does not recall funds if below by less than plantableMargin
      transmuterPostDistributeBal > plantableThreshold
        ✓ sends excess funds to the active vault (284ms)
        ✓ sends the exact amount of funds in excess to reach plantableThreshold
        ✓ does not send funds if above by less than plantableMargin
      transmuterPostDistributeBal == plantableThreshold
        ✓ does nothing
  recall
    recallAllFundsFromVault()
      ✓ reverts when not paused
      ✓ reverts when not governance or sentinel
      ✓ recalls funds from active vault
      ✓ recalls funds from any non-active vault (57ms)
    recallFundsFromVault
      ✓ reverts when not paused
      ✓ reverts when not governance or sentinel
      ✓ recalls funds from active vault
      ✓ recalls funds from any non-active vault (55ms)
  harvest()
    ✓ harvests yield from the vault
  migrateFunds()
    ✓ reverts if anyone but governance tries to migrate
    ✓ reverts when trying to migrate to 0x0
    ✓ reverts if not in emergency mode
    ✓ reverts if there are not enough funds to service all open transmuter stakes
    ✓ sends all available funds to the new transmuter (53ms)


355 passing (2m)
```

Code Coverage

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	78.17	59.95	71.88	78.44	
Formation.sol	86.31	71.43	82.05	86.63	… 606,607,608
FormationV2.sol	80.57	64.89	78.95	81.01	… 614,615,616
MultiSigWallet.sol	0	0	0	0	… 313,315,316
MultiSigWalletWithTimelock.sol	0	0	0	0	… 87,88,94,95
NAOSToken.sol	100	100	100	100	
NToken.sol	88.46	75	84.62	89.66	91,104,105
StakingPools.sol	87.64	83.33	83.33	86.81	… 247,276,277
Transmuter.sol	**93.98**	**68.75**	**91.67**	**94.24**	**… 444,446,448**
TransmuterV2.sol	93.27	67.78	87.76	93.48	… 642,651,652

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/adapters/	62.59	31.25	60.53	62.24	
AlpacaVaultAdapter.sol	47.95	25	37.5	48	… 273,283,292
EllipsisVaultAdapter.sol	72	38.89	54.55	70.59	… 164,173,193
YearnVaultAdapter.sol	100	50	100	100	
YearnVaultAdapterV2.sol	100	100	100	100	
contracts/interfaces/	100	100	100	100	
I3ESPool.sol	100	100	100	100	
IAlpacaPool.sol	100	100	100	100	
IAlpacaVaultConfig.sol	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	
IEllipsisPool.sol	100	100	100	100	
IMintableERC20.sol	100	100	100	100	
ITransmuter.sol	100	100	100	100	
IUniswapV2Router.sol	100	100	100	100	
IVaultAdapter.sol	100	100	100	100	
IVaultAdapterV2.sol	100	100	100	100	
IYearnController.sol	100	100	100	100	
IYearnVault.sol	100	100	100	100	
IbBUSDToken.sol	100	100	100	100	
IyVaultV2.sol	100	100	100	100	
contracts/libraries/	91.3	64.29	80	91.3	
FixedPointMath.sol	91.3	64.29	80	91.3	29,39
contracts/libraries/formation/	84.76	55	94.12	84.76	
CDP.sol	53.13	43.75	85.71	53.13	… 6,87,90,107
Vault.sol	96.88	100	92.31	96.88	105
VaultV2.sol	100	100	100	100	
contracts/libraries/pools/	89.29	100	80	89.29	
Pool.sol	85	100	75	85	103,112,113
Stake.sol	100	100	100	100	
contracts/mocks/	80	47.67	78.46	81.15	
AlpacaStakingPoolMock.sol	86.79	45.83	100	90	… 113,117,118
AlpacaVaultConfigMock.sol	0	100	0	0	6,10
ERC20Mock.sol	66.67	100	66.67	66.67	23
ERC20MockUSD.sol	100	100	100	100	
EllipsisPoolMock.sol	87.27	45.83	100	90.38	… 115,119,120
I3ESPoolMock.sol	96	58.33	100	96	60
IbBUSDMock.sol	83.33	37.5	100	83.33	48,49,50
UniswapV2Mock.sol	100	50	100	100	
VaultAdapterMock.sol	100	100	100	100	
VaultAdapterV2Mock.sol	100	100	100	100	
YearnControllerMock.sol	25	100	25	25	20,30,34
YearnVaultMock.sol	61.76	50	55.56	61.76	… 72,73,88,89
YearnVaultMockUSD.sol	62.86	50	55.56	62.86	… 73,74,89,90
All files	**77.9**	**55.56**	**74.28**	**78.18**	

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

```
0ac069e79fafd6f34755f946bb18cb7116b80539674a0aeebb917448640e837d ./contracts/Transmuter.sol
5853a9c02b98281fa4de50617476800ce2de4891977974def714a62fede5c1ed ./contracts/FormationV2.sol
0028abc5ce66db7383bd95ab91d419c63febf01091631a4cec6e2d4a63dcacff ./contracts/NToken.sol
04df2adba97228f808a6a5e5fdf137c50c0f109e417e867082d541c016db5837 ./contracts/StakingPools.sol
9dc0785eb8dcd01cb4b2323d1bf607ac561fb5ce4d348b790f2c85f5f348c38b ./contracts/MultiSigWallet.sol
a8e7017949faf82008718d4bfcbf9d7da92589a4d37bb6ae58dbec764e03c610 ./contracts/MultiSigWalletWithTimelock.sol
9643ce62e3ae6327c9955cd818bfb04d95d658b3f7d332e5654c1ad52f995c07 ./contracts/NAOSToken.sol
f0a8befe1aa10645b2d7e62d50b1a339048a72710c0b1b47a11dcb572d967a84 ./contracts/Formation.sol
a38fa5e74f41d5b231af3c0077394e77336ed3d900cfdbb9314fa2283f3972bf ./contracts/TransmuterV2.sol
32805b2f9e43f6464a1417aa599d2b3c013ca42ffe46368688c8ef6dd6cb590a ./contracts/interfaces/I3ESPool.sol
4e60a44a0115f6315790db9a5e00ffc0b2431fc64f84e082382949d58ee66442 ./contracts/interfaces/IbBUSDToken.sol
085df950efad6da874d57cf716e3d13d518b166eafe56477609c60b1e8136a86 ./contracts/interfaces/IVaultAdapterV2.sol
0fc3af25f10021274a0bdf342391414f6b5dc83a0fbffd20798707d9fd2f0bcc ./contracts/interfaces/ICurveMetaFactory.sol
0d86c02d5f51e518033e854d0aad73408a47ab1eccbfa1d947a82e1e3e86e9ef ./contracts/interfaces/IUniswapV2Router.sol
8a9de3b97978b2947c5422a140e3965020eea0b6f4870e1529b710fe15f65667 ./contracts/interfaces/IERC20Burnable.sol
1f2a8bfde6c51afe997eff851591bf44c425fa901c314fa444d0fb9f2499bb92 ./contracts/interfaces/IYearnVault.sol
1f70eb25eb5c37e55c260fbb2ffbdab46de8da53c5566c1f99edb0fa30ae3d8a ./contracts/interfaces/IYearnController.sol
768695e9856722a223c2f34203491229cf29423650beea296470da2afe74295c ./contracts/interfaces/IAlpacaVaultConfig.sol
c3e332eade464787d6bf9d89514854a86e2d89e8bb6ab84e1b64f5431b22eeaa ./contracts/interfaces/IyVaultV2.sol
e4a04f9f4a7d24fb651097c230aa4966eab42543696cd4d3556ba28289590235 ./contracts/interfaces/IMintableERC20.sol
7f11a2c63e3893e89d63838666ecbb385428f421d9c2d84d08be733dd7ed343f ./contracts/interfaces/IChainlink.sol
2f643bb90b225045d75c003858cbfa8c9fc44c370dfb3e9eb60a0af51bd4ed9e ./contracts/interfaces/IDetailedERC20.sol
550af86706540ab6047c68d47453d71d3ae1e981a13ad7700ab1ccaf1072751e ./contracts/interfaces/IAlpacaPool.sol
f28f5bb67326a17c1e8d25311711a84f48807e33b4379138d8f4ef4cf7b48821 ./contracts/interfaces/ITransmuter.sol
77660b9405fabb13e3e0c1fb0d19956063774f607a4b28ab5d85a423503b2db1 ./contracts/interfaces/IVaultAdapter.sol
f118c4536fc6959c35d3ef3115ff92164acc34559878ceb0798f920f4a934a09 ./contracts/interfaces/IEllipsisPool.sol
879f32e821719e921465eaf9606c5ebcdf2735fd9f411a14e89f8f270564909b ./contracts/adapters/YearnVaultAdapter.sol
d41d9b53b244a7c69a7c2195d858fd6c8693100c59683e3e0b4be433a16cc7e8 ./contracts/adapters/EllipsisVaultAdapter.sol
a000273946d6d68e33b1ada650dbd68d4cd14ebc89b7d872b62220e963c86db4 ./contracts/adapters/YearnVaultAdapterV2.sol
5c76280aa712563fbb4ab293164004d10068b4dd4b9949b86a4924a226293640 ./contracts/adapters/AlpacaVaultAdapter.sol
f031f1f809c6676129dc03aa89042642c359bee9bbe851801ab20585c6634d00 ./contracts/libraries/FixedPointMath.sol
02ce9677b701b5484019ca3cf6c62c6ff014ac0a151f1c2182f4a9abf7f6bbcf ./contracts/libraries/formation/CDP.sol
c0898a6876962cdffca28cae162ceee48b0a846528353ddb20d4a6594b309d70 ./contracts/libraries/formation/Vault.sol
13c4bfd28e3cea83ae9e5dbba40cff6e36c61fe62f963ccb0b86fc40692348f0 ./contracts/libraries/formation/VaultV2.sol
63d7ac6ba2ba42f9985d02fe547bef7f6231af3cc95e8d19d16f780e3db9ab12 ./contracts/libraries/pools/Pool.sol
d3cc9a239e9913738ecae2a5cbc30b64d53ccbdc8138e9e1dd20c560d80bbdfb ./contracts/libraries/pools/Stake.sol
991821120883d0ceb42eab8213e77bcc0253f4e6fd6fc1f7c1a9f81957874478 ./contracts/mocks/YearnVaultMockUSD.sol
367b39dfcea11af4178391b24b444d379002126e8b743e877e7aa0a1cd373ef5 ./contracts/mocks/UniswapV2Mock.sol
d7d577afb0f3e69c545eddae65bf487b83c22ef18acc32c7c6a61047d4551e0e ./contracts/mocks/YearnControllerMock.sol
90ead4562ce825244fe534dce5878d45d35fc56fa3441c63d7d714cbe1fdb258 ./contracts/mocks/IbBUSDMock.sol
1b7a02a98a4c77f0f9e8ae85815e4ddc66a79e2c1458992b8723109c0fa59c1e ./contracts/mocks/EllipsisPoolMock.sol
0b2aaa74369aa8bcf325cdbe020cb561077a2e206f346a091d089fc7fedb394c ./contracts/mocks/VaultAdapterMock.sol
ff329a7fb34d04cf81d50f2f1fd04659bb0272d78b7aa30806bdc3bbe812dc7a ./contracts/mocks/AlpacaStakingPoolMock.sol
11e6932d42a9dc38d688d1aecea75d6f263ea79042edc6dd5be100fc5e240ed8 ./contracts/mocks/I3ESPoolMock.sol
7db7d9e75a1cf38fadaa4ee469c8278563937e7dd9361e47afc0330e172b63c2 ./contracts/mocks/ERC20Mock.sol
b1190d357109fbbcb9f03ed1ef31192b5f72c88aa51b1d026e6f37b6253daf96 ./contracts/mocks/ERC20MockUSD.sol
311ba796ca31d17bcbfa6ee03d0615ec4cb5adf3d018829613c2e1bb91e71def ./contracts/mocks/VaultAdapterV2Mock.sol
5291fd4d6f45d12dc22226b826d94e75fef0f65db78d084fc3b435641871d27e ./contracts/mocks/AlpacaVaultConfigMock.sol
8e457ec59a4c94dae80e7c4e32249f7e6de7fe37ddd571a8d9c17e59c3eead2c ./contracts/mocks/YearnVaultMock.sol
```

Tests

```
0f683d627b09bf1fea3a59870a060156067519de76ffa85e1442b91aabe9e446 ./test/utils/helpers.ts
a51aec117e9dc151f05b2faab9dbe998642b957f798f044b1611c0b5283aae72 ./test/utils/ethereum.ts
1f03ef18f721eff5dedb15bc1857cb414823ecc217b4d7b0162da80ec1ff0958 ./test/contracts/TransmuterUSD.spec.ts
ba24a79f1b0826336e8ed1274b12fbdd0382b51008fefbbc3fd3ee4577b5fd09 ./test/contracts/Formation.spec.ts
7917f298bccd1312efafb1883e1a2f58172d9827b45cd895f74c502c59989e9e ./test/contracts/StakingPools.spec.ts
eace4c7dc4fb0e8d76b5657fb66b2d9e095f30e98598761947339fbbc3f6ff65 ./test/contracts/TransmuterUSDV2.spec.ts
cf4e99125c1fdcb206a257e01d0dfadcbc5f239dd5c322059f776345b1317675 ./test/contracts/NAOSToken.spec.ts
```

```
427ee05cfa97c00ef3ed97a85a57f72815c7d0aad1cd20a0b6ef762da0852613 ./test/contracts/Transmuter.spec.ts
2337a70f477e16ca4a4c83c0bc1001837642097e1573c6748266556dc54c28d0 ./test/contracts/FormationV2.spec.ts
574517681af121d1fd9facc29a0d1f2d1db870f3722444ebfa6284198a2e7305 ./test/contracts/TransmuterV2.spec.ts
```

Changelog

- 2021-11-17 - Initial report
- 2021-11-24 - Revised report based on commit `a721545`.

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.