Quantstamp Security Assessment Certificate

January 6th 2021 — Quantstamp Verified

dForce

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

Executive Summary

Type	Defi protocol
Auditors	Kacper Bąk, Senior Research Engineer Fayçal Lalidji, Security Auditor Jan Gorzny, Blockchain Researcher
Timeline	2020-10-26 through 2020-12-15
EVM	Muir Glacier
Languages	Solidity, Javascript
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	USR Scheme USDx Design
Documentation Quality	Medium
Test Quality	Medium

Source Code

Repository	Commit
USDx 1.0	ed9e0ce
USR	48bf97e

Goals

- Is a DoS attack possible?
- Can any funds get locked up in the contract?

Total Issues	**12** (1 Resolved)
High Risk Issues	0 (0 Resolved)
Medium Risk Issues	**4** (1 Resolved)
Low Risk Issues	**6** (0 Resolved)
Informational Risk Issues	**2** (0 Resolved)
Undetermined Risk Issues	0 (0 Resolved)

ALL ISSUES ADDRESSED

Documentation Preparedness

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

Over the course of the assessment we have found a few medium and low-severity issues. Furthermore, we were unable to run tests in both repos despite following the steps indicated in the corresponding readme files. The project lacks proper specification and code documentation, and therefore it is impossible to assess the correctness of the provided code. We highly recommend addressing all these issues before proceeding with deployment of this project.
Update: the team acknowledged most of the issues and informed us that the code has already been deployed. We were able to build and run USR tests but not USDx tests.

ID	Description	Severity	Status
QSP-1	Tokens that add rewards or take fees may lead to DoS	Medium	Acknowledged
QSP-2	Uniswap Oracle Manipulation	Medium	Fixed
QSP-3	Fee Recipient Update	Medium	Acknowledged
QSP-4	Pool Migration	Medium	Acknowledged
QSP-5	Lack of input validation	Low	Acknowledged
QSP-6	Privileged Roles and Ownership	Low	Acknowledged
QSP-7	Too large value of `minimalBurnAmount` may prevent users from redeeming tokens	Low	Acknowledged
QSP-8	Malicious tokens may cause a DoS	Low	Acknowledged
QSP-9	Allowance Double-Spend Exploit	Low	Acknowledged
QSP-10	Gas Usage / `for` Loop Concerns	Low	Acknowledged
QSP-11	Unlocked Pragma	Informational	Acknowledged
QSP-12	Clone-and-Own	Informational	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 v0.6.12

Steps taken to run the tools:

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

Findings

QSP-1 Tokens that add rewards or take fees may lead to DoS

Severity: *Medium Risk*

Status: Acknowledged

File(s) affected: `DFCollateral.sol`

Description: Some functions make implicit assumptions that tokens remain constant in their supply during transactions. There exist tokens that add rewards or add fees, and hence their supply does not have to stay constant. We identified the following problematic pieces of code:

- In `DFCollateral.sol`: line 18,
- In `DFFunds.sol`: line 18,
- In `DFPool.sol`: lines 23, 35, 47, 59,
- In `DFPoolV2.sol`: lines 29, 41, 59, 77, 187, 203,
- In `DFEngineV2.sol`: functions `deposit()` and `withdraw()`. Also the function `checkUSDXTotalAndColTotal()` in line 329; consequently all other operations that rely on it would fail.

Recommendation: We recommend that any tokens used in the project are carefully vetted so that they do not cause any unexpected issues.

QSP-2 Uniswap Oracle Manipulation

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `UniswapOracle.sol`

Description: The implemented `Oracle` contract in `UniswapOracle.sol` calls the uniswap router to get the swap output value to be used as price estimation. However, depending on its usage, the router output can be manipulated by trading X amount back and forth on that specific Uniswap pair and the oracle call can be placed in between.

Recommendation: Uniswap implemented a special oracle that provide time weighted cumulative price that can be averaged later depending on the project needs, please refer to Uniswap documentation for more details.
Update: fixed as of commit `83066e2`.

QSP-3 Fee Recipient Update

Severity: *Medium Risk*

Status: Acknowledged

Description: When updating the interest provider in `USR.updateInterestProvider` the inherited fee recipient from `Chargeable` is not updated. (The address might be the same, however, nothing guarantees it.) Please note that in `USR.initialize` the fee recipient address it taken from `IInterestProvider(_interestProvider).funds()` which is not done inside `updateInterestProvider` when updating the provider.

Recommendation: Use `updateFeeRecipient` inherited from `Chargeable` contract to update the recipient. Prior to calling the function, the new recipient must be checked to be different than the initial address, otherwise `updateFeeRecipient` will throw.

QSP-4 Pool Migration

Severity: *Medium Risk*

Status: Acknowledged

File(s) affected: `DFPoolV2.sol`

Description: The check in lines 203-206 implemented in `migrateOldPool()` assumes the new contract does not hold any token balance, when tokens can be sent to the contract. The usage of such strict equality can deny access to that specific function.

Recommendation: The initial balance can be checked to be equal to zero before making the token transfer, to avoid token balance inequality.

QSP-5 Lack of input validation

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `DFPool.sol`, `DFPoolV2.sol`, `Collaterals_t.sol`, `DSWrappedToken.sol`, `DFSetting.sol`, `DFProtocolView.sol`, `DFEngine.sol`, `InterestProvider.sol`, `Chargeable.sol`, `USR.sol`

Description: There are no non-zero checks for arguments of type address in the following:

- `DFPool.constructor()`,
- `DFPoolV2.constructor()`, `DFPoolV2.initialize()`, `DFPoolV2.migrateOldPool()`,

- `Collaterals_t.constructor()`,
- `DSWrappedToken.constructor()`,
- `DFSetting.constructor()`,
- `DFProtocolView.constructor()`,
- `DFEngine.constructor()`,
- `InterestProvider.initialize()`,
- `Chargeable.initialize()`,
- `USR.initialize()`.

Furthermore, `DFStore._setSection()` does not check that `_wrappedTokens` is a list of unique tokens.

Recommendation: We recommend adding relevant checks.

QSP-6 Privileged Roles and Ownership

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `DFStore.sol`, `ERC20Pausable.sol`

Description: Smart contracts will often have `owner` variables to designate the person with special privileges to make modifications to the smart contract.

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-7 Too large value of `minimalBurnAmount` may prevent users from redeeming tokens

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `DFStore.sol`

Description: If the value of `minimalBurnAmount` is too large, it will prevent users from redeeming tokens.

Recommendation: We recommend informing users about this potential risk. We also recommend adding a limit on what the maximum value of `minimalBurnAmount` can be. Finally, we recommend double checking the value before setting it.

QSP-8 Malicious tokens may cause a DoS

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `DFStore.sol`, `InterestProvider.sol`

Description: The function `_setSection()` adds tokens. Malicious token may lead to a DoS attack. Similar observation applies to the function `withdrawInterest()`.

Recommendation: We recommend vetting the tokens carefully to avoid a DoS.

QSP-9 Allowance Double-Spend Exploit

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `DSToken.sol`

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

- `approve()`/`transferFrom()` in `Collaterals_t.sol`, and
- `approve()`/`transferFrom()` in `DSTokenBase.sol`.

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-10 Gas Usage / `for` Loop Concerns

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `DFEngineV2.sol`

Description: Gas usage is a main concern for smart contract developers and users, since high gas costs may prevent users from wanting to use the smart contract. Even worse, some gas usage issues may prevent the contract from providing services entirely. For example, if a `for` loop requires too much gas to exit, then it may prevent the contract from functioning correctly entirely. It is best to break such loops into individual functions as possible.

Recommendation: We recommend performing a gas analysis to find out the iteration limits in the loops.

QSP-11 Unlocked Pragma

Severity: *Informational*

Status: Acknowledged

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.

QSP-12 Clone-and-Own

Severity: *Informational*

Status: Acknowledged

File(s) affected: `ReentrancyGuard.sol`, `DSAuth.sol`, `DSMath.sol`, `DSNote.sol`, `DSThing.sol`, `DSValue.sol`, `DSGuard.sol`, `DFProxy.sol`, `Collaterals_t.sol`, `USRProxy.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.

Adherence to Specification

Specification for the logic of `DFStore.sol` is missing. Therefore, it is impossible to verify its correctness. In line 167 it sets `mintingTokens[_wrappedTokens[i]]` to `true` without checking if it was `false` before. Should it?
It is unclear why in `DFEngineV2.sol#193` `_amount` needs to be a multiple of `dfStore.getMinBurnAmount()`. The latter name seems confusing. Similarly, in line 259, it is unclear why `_amount` has to be a multiple of `_sumMintCW`. **Update:** the team informed us that in lines 193 and 256, the mint and burn amount is required to meet certain criteria, such as the minimum unit, to support different underlying constituent tokens with different decimals.
The modifier `auth` is used in `DSToken.burn()`. The function calls `_burn()`. Even if the `msg.sender` has to be authenticated it: 1) can either burn own tokens, or 2) burn only an approved amount of tokens on behalf of the other user. It is unclear if this is intentional design or not. **Update:** the team informed this design is intentional.
The specification for formula in `Chargeable.calcAdditionalFee()` is missing. **Update:** the team informed us that `Chargeable.calcAdditionalFee()` is called by functions like `redeemUnderlying(amount)`, the `amount` should be the net amount caller get, so `fee = amount / (1 - feeRate) - amount` to be consistent with `redeem(amount)`.
In `Collaterals_t` contract `_totalSupply` state variable is declared with a value of `10**58`. The comment in the same line indicates a different value since the decimals are declared with a value of `18`. That would indicate that `_totalSupply` value should be `10**28`. **Update:** the file is only used in tests.

Code Documentation

In general the code is poorly documented and lacks inline comments.

Adherence to Best Practices

1. In `DSNote.sol` and `DSValue.sol`, the naming of variables is cryptic. We recommend making the names more explicit. **Update:** acknowledged.
2. In `DFStore.sol#139`, typo "data not allow" -> "data not allowed". **Update:** acknowledged.
3. In `DFPoolV2.sol#220`, typo "dToekn" -> "dToken". **Update:** acknowledged.
4. `IERC20.sol` does not fully conform to standard; e.g. `transfer()` does not return a boolean result. **Update:** acknowledged. It is intentional to support `USDT`.
5. Both `decimals` and `_totalSupply` state variables are declared with an initial value, but they are both reset in the constructor. These state variables can declared without any initialization. **Update:** `Collaterals_t` is only used in tests.
6. In `DSToken.transferFrom()`, although it is not required, an approval event could be emitted each time the allowance is reduced to allow for better tracking of the allowances (see the OZ implementation). **Update:** acknowledged.

Test Results

Test Suite Results

The team provided us with the following instructions to run tests.

USR

Clone the repo with git sub modules:

```
git clone --recursive https://github.com/dforce-network/USR
git checkout audit
git submodule update
```

in an existing USR repo.
Install buidler and plugins

```
npm install
```

Run the following commands to compile all contracts:

```
npx buidler compile
```

Compile the USDx contracts for integration test:

```
npx buidler compile --config buidler.config.usdx.js
```

To run the tests:

```
npx buidler test
```

USDx

Install packages😉:

```
npm install
```

Test:

```
npm run test
```

```
USR
   Initializable
      ✓ Should be able to initialize (649ms)
      ✓ Should not be able to initialize again
   ERC20Pausable
      ✓ Should be able to pause (76ms)
      ✓ Should be able to unpause  (135ms)
   Chargeable
      ✓ Should be able to update fee recipient
      ✓ Should be able to charge some fee when mint (82ms)
      ✓ Should be able to charge some fee when redeem (97ms)
      ✓ Should be able to charge some fee when redeemUnderlying (118ms)
      ✓ Should be able to update fee to zero
   ERC20Exchangeable
      ✓ Initial exchange rate should be 1.0
      ✓ Should be able to update exchange rate (53ms)
      ✓ Should be able to get underlying balance (103ms)
   Mint/Redeem/RedeemUnderlying
      ✓ Should not be able to mint < 0 when totalSupply is 0
      ✓ Should be able to mint with mock profit provider (68ms)
      ✓ Should be able to redeem with mock profit provider (93ms)
      ✓ Should be able to redeemUnderlying with mock profit provider (82ms)

USDx_1.0


xDAI address : 0x80e1d5a1f7BDAc82dAe838416C3dc47cd2e5F77e
xPAX address : 0xd819760c09Dae98B9C861bf6C71B0eea3a680c87
xTUSD address : 0x15a26542AE1B9b98aA03f8686fBD5E38EC35d019
xUSDC address : 0x4ae5daA20a0310a7DB1EF3d687E8D0F37058a441
----------------------------------


1 ####### xDAI.setAuthority

-####### 0x61ee55016e7e8aa3c60d5538521650e878575bcfb0ed26c9171af6ba77d714d3

2 ####### xPAX.setAuthority

-####### 0x206c7f8e7e1fadbd222409555d922591182822 71f69005c931f5cd5f5c3220af

3 ####### xTUSD.setAuthority

-####### 0xbee0b6ce7d1fe00e893ef03e14784491a7016f787a3ac45516151665392bbeba

4 ####### xUSDC.setAuthority

-####### 0xd8bf8548bbf7b0e48d08a713199c03aad267a274ecf3a26039bc780ae2d6d661

5 ####### contractPool.approve xDAI

-####### 0xccf3405d94cb565bf36b2d6ccaa0c5094a08a4abf7489d61d561fcafc537faf8

6 ####### contractPool.approve xPAX

-####### 0x1949db9aaece1230588585577d63a334559d4700e29f5e63eefb8a2cfd240385

7 ####### contractPool.approve xTUSD

-####### 0xc590c020c11a50f135bd8de24b3b9429091470474254e69f7330149a26e660e6

8 ####### contractPool.approve xUSDC

-####### 0x36169be9debaae899078aac58d843c36305391aab261ef8878b4306e6dcbb7a0

9 ####### contractCollateral.approve xDAI

-####### 0x906477b411f3ce0d47bfdf7f79eff66e6ccc5c219417d18e246fb5d98033d733

10 ####### contractCollateral.approve xPAX

-####### 0x36c148b5df14af0f3483ec71b586580e3f1a97b8e423573f2944d1fdb1aa58dc

11 ####### contractCollateral.approve xTUSD

-####### 0x645f7f5456447280a4fee3ea1de0d7c44fa5f20d3374a36844038ff56a5e2105

12 ####### contractCollateral.approve xUSDC

-####### 0x2a17dd8fe9253372f3ea6484a686bf87e06a08217a0436b8fb8f579208527fd9
```

```
1 ####### contractUSDx.setAuthority

-####### 0x533fda3e3c34dedad7edb7e68ff8fbbf3fc2131ce40c065e03abf1fc3c5717ea

2 ####### contractStore.setAuthority

-####### 0xa084563318fb078f1f6cb69da5ee89707c1926adb65ec5a09446fdf004f48b9b

3 ####### contractPool.setAuthority

-####### 0x11648b1a50d0ce4e1f2bc713ca08afa4d1b226eca0556c53ef7a6867ce69a660

4 ####### contractCollateral.setAuthority

-####### 0x2d2b1633328fec2642cf60cf35976773dc42b9367f411f00b8c563750f06f3ed

5 ####### contractFunds.setAuthority

-####### 0xd48ab14930a1d749fb3ec372720e87e1173ca4874efa1b1bbda6bff972d503d6

6 ####### contractEngine.setAuthority

-####### 0xac803f7ddafa9325d60794f1726deb3af77d7f0e55543da7933734e92dc58317

7 ####### contractSetting.setAuthority

-####### 0x2e480d62152f1e21251cfcd4e8356b45a9f66e466dfe5843dfd9ab8b080b0868

8 ####### contractGuard.permitx Store Engine

-####### 0x58dafe3df15f3cca5953e95848fa5ec1391741f376227a0c2e0875a2c36916fd

9 ####### contractGuard.permitx Store Setting

-####### 0x7ad9b7472d79722a3f92b94da9ef70cc123f746e2b32df33edd3a119df6a9b34

10 ####### contractGuard.permitx Pool

-####### 0x5cfbe61385771c897cc431f7367f88a473ea45d47bcb7fb92312760433535cf7

11 ####### contractGuard.permitx Collateral

-####### 0xfa72273e8be13c223ca8660681b16ca58cb721e806f7827580b1922039914297

12 ####### contractGuard.permitx Funds

-####### 0xd95552c70932771c2db9b305c8e72d5d78b3dbf4f16be1887a7cec8493f21a19

13 ####### contractGuard.permitx Engine

-####### 0x920dea659825fc26c95100fe14eeef15c8ba85ba3521be619b0076bac0a7cf8e

14 ####### contractProtocol.requestImplChange

-####### 0xa4d984f3f38c16089942f168743b0cf4bbb6d0e1f26ca2021ef3de26117aec40

15 ####### contractProtocol.confirmImplChange

-####### 0x3f8fd627a65962b5e44ec5d2540216e37b73df8c59e3814620ab7dde9d3e2f14

16 ####### contractSetting.setCommissionRate

-####### 0xd3b56050b6f3c8749a7a1e06b792a1a5b9010b7d2e94eec3c2e5d131fb9da6e2

17 ####### contractSetting.setCommissionRate

-####### 0x341c64730b822f25874c80d32ad9f658684ca943378e80d4e9f70ef7aa94827d

18 ####### contractSetting.setCommissionToken

-####### 0xf9b7b0a2fbf443ed0ed9b66ca37d01f38d0cad047f675740b4b1c0a172a6de22

19 ####### contractSetting.setDestroyThreshold

-####### 0x7fd68f68b91ae015ef0a0fe3e44b465177b604c5ec11070368f1ef337e9ea5f5

20 ####### contractSetting.setCommissionMedian

-####### 0x22e25eb9de4f1ad418dc56c673e92fe5cb903b3a107eda33696fe81a7cde3a60

21 ####### MedianizerSetFeed.post

-####### 0xfbdc724eda5679b0281d368e3f3ea147b85f426939913d2947a69ca6cb3e68aa

22 ####### contractPriceFeed.post

-####### 0xaafa219bd52d9bb3e4b68fc4c93db7918a0314e634008f3ebb6f4dece0fe3213


  DFEngine deposit claim destroy
    ✓ Config 0 (8588ms)

  DFEngine deposit claim
    ✓ Config 0 (35633ms)

  DFEngine deposit destroy
    ✓ Config 0 (167012ms)

  DFEngine deposit withdraw claim destroy
    ✓ Config 0 (105480ms)

  DFEngine deposit withdraw claim
    ✓ Config 0 (128316ms)

  DFEngine deposit withdraw destroy
    ✓ Config 0 (131296ms)

  DFEngine deposit withdraw
    ✓ Config 0 (166502ms)

  DFEngine random
    ✓ Config 0 (149828ms)

  DFEngine claim amount
    ✓ Config 0 (75710ms)
    ✓ Config 1 (78829ms)

  DFUpdateEngine
    ✓ Config 0 (32896ms)

  test0.1
    ✓ Config 0 (22952ms)
    ✓ Config 1 (36749ms)
    ✓ Config 2 (31672ms)
    ✓ Config 3 (22715ms)
    ✓ Config 4 (30366ms)
    ✓ Config 5 (75188ms)
    ✓ V2 migration and verify (11340ms)

  test0.2
    ✓ Config 0 (24057ms)
    ✓ Config 1 (69645ms)
    ✓ Config 2 (67560ms)
    ✓ V2 migration and verify (12342ms)

  test0.3
    ✓ Config 0 (94079ms)
    ✓ Config 1 (103045ms)
    ✓ V2 migration and verify (10922ms)
```

```
test0.4
   ✓ Config 0 (92792ms)
   ✓ Config 1 (162652ms)

USDx with Pool & Engine V2
   ✓ Deployment (9238ms)
  User Operations
     ✓ should be able to deposit and withdraw (2281ms)
     ✓ should be able to destroy (1196ms)
     ✓ should be able to one click minting (1076ms)
  Admin Operations
    Stop
       ✓ should be able to stop by owner (1063ms)
       ✓ should not be able to stop by non-owner (50ms)
       ✓ should not be able to start by non-owner (50ms)
       ✓ should be able to start by owner (1111ms)
    Transfer Assets
       ✓ should be able to transfer assets by owner (363ms)
       ✓ should not be able to transfer assets by non-auth (373ms)
  Query Interface for USR
     ✓ should be able to call getUnderlying() (187ms)
     ✓ should be able to call getInterestByXToken() (887ms)


39 passing (33m)
```

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

```
0ecce745ab3804706e10bdb5b88ef0bf34c353c7eaa644be745b6c3a40f327ab ./Chargeable.sol
5b2a866917892358a44c217278db01ec7a6bf4d0b891046835f596e4be38039f ./Funds.sol
91575a780daa458017fc7829bbb543b227f2c4512ccb42dbed6e683322aefae9 ./USR.sol
451a2f41f7d1777acacf9fba56432d74c1272787fdc5462475fc2537418a40da ./Migrations.sol
453f84ce217159b09910d5ca66e50a7a29753a5fd25451ab628e927ad61974ff ./InterestProvider.sol
a9d01f3ca2dea6ba43aba867563cf8f4240ebb0e83ae5fe1693ecb0c1f65f4e9 ./USRProxy.sol
1eeafc207caa85071431a1feb87dd123f7638df00263e5169b8fd94b6509a09c ./DSAuth.sol
a9a0033c36c10f8c424e3044432ec24ef90cd14ae507f82d8d58de1734fbd5ac ./Pausable.sol
c9f7414432e82700d017d15ef2c7e5a69287cfcd6f186f985c281b5ceaa4ee3a ./SafeRatioMath.sol
faa945b8a12e618e467910d81a813d2c6965db6263692aa97942d1f02996f41b ./ERC20Pausable.sol
c66471d2e1ccaee1639ed2da842c5dfb850b4108976a17659f4902375c728798 ./DSGuard.sol
de25ab249c86d6c7aa1ad09429195eff1962935d7acb3011ae02fc3600bd704b ./InterestProvider.sol
3d1d012e937c18c09d445e4f19309bed36a0e7886623164e460a26c5b13c5b82 ./USDx.sol
2140e945a70affce558607528972c57b486b34a2af778b26bfeb9f399e23b742 ./IInterestProvider.sol
ae7a88204dee99d02bc8cec029e2110aa3e4113ed2d06daccdcd024a6139ed59 ./contracts/utility/ReentrancyGuard.sol
459e17849150154e826b699a1ccb1d2aaf581c33026a1ec2d7a71b8edb0a8bcb ./contracts/utility/DSAuth.sol
0f86ccf0cff2c6fa4ccd5a5715140112b5a98a7867edaa283225a1c9b9f627bf ./contracts/utility/DSNote.sol
bf8607fa7009853edd8024fd6ac9bc481f0677cfef6001c77a63938833a86c51 ./contracts/utility/DFProxy.sol
b35e5b61e1b99d307b5f2290968efeb5daad7f1f1b929d65344af082fee33bdc ./contracts/utility/DSValue.sol
6e9ecc0cde606e016672f419ec9966ef5636f426be3a735e41d2a7b6bd8ede00 ./contracts/utility/DSGuard.sol
2961a36433028add089e76b7d8040272c6763bd88d439c2d7ce821a2b96f5ce1 ./contracts/utility/DSMath.sol
98ebba0a696037e88fc6f6bb4abaaaac91ea8b7e49fe791f43805b19d5e057da ./contracts/utility/DSThing.sol
07dcb9e46a8b118f1e25e41e3b31185e127b8fad556e90d4de4cd45dd64c3f28 ./contracts/utility/Utils.sol
4853bc08ae89cc40d347ee5cdb5ce112236203dea139a2515e6624fea9bd3930 ./contracts/helpers/Migrations.sol
4809890405ef3bc86f1a9ae82123661bd83978580cace5ebfee46b6a77cbd9d4 ./contracts/storage/DFCollateral.sol
eb3440c09e583384a0d1a8b9debb213ae5cd437eb1aefc69bb1ab96fd0ed5632 ./contracts/storage/DFPool.sol
d5558e99ee7f1faa21ebb077b3eb05b1684120391d2592a35e1bdeb622588ea4 ./contracts/storage/DFPoolV2.sol
d37012b08649a93ac39994ac650c442a34c438e5109983ea28117940823bc75e ./contracts/storage/DFFunds.sol
29428484857ced89c60ff847ac7c1a9378180d0ad8f65d0cd87c6be6fc6b9c55 ./contracts/storage/DFStore.sol
149ef57bd23db7562dc0aa8f22772cf750fccdd5fcd7942f2b10883639d1683a ./contracts/storage/interfaces/IDTokenController.sol
4eac55ed0020e384d0dcd16b25788a80d38bfa15b012315a325a7b1c5ecaea1f ./contracts/storage/interfaces/IDFCollateral.sol
9e5b4388ad7c987b572b0fc75e07ced22ae4cab342815dd8430e94894b80d9e9 ./contracts/storage/interfaces/IDFPoolV2.sol
84c3cf18cdfd859a3c6ecc4ca2f710c77ffa1ab2654483e2d2439eb02cb4b02b ./contracts/storage/interfaces/IDToken.sol
45dd22d984a5d056f33a9a6f4812f365ac9c7aa4601bb23bf87cee2e5cf41021 ./contracts/storage/interfaces/IDFPool.sol
ae4ba04e95e24ea88c26402a08aacbc5382662da8a32ae8c2e7d3ce94c74b1ce ./contracts/storage/interfaces/IDFFunds.sol
3f529608af5dc7a4d714eed76e1d7721f97c05fc0b8b9e6c36dfb464131e3ad7 ./contracts/storage/interfaces/IDFStore.sol
f6f627efb26e8e03bd0bcdc9b1867fd27d21756f50936a420a8a313c3dcb165d ./contracts/oracle/UniswapOracle.sol
36b88731c141000f90622362b88fad82cd16b9a8056652d6a5e6904ff6e7080d ./contracts/oracle/Medianizer.sol
b4302dd4108d3db20f512b1c647ceea85ea5cc07ef58548e3399255fdedf202c ./contracts/oracle/PriceFeed.sol
be2491863f0557813265a437f084b93445ec9c342b26e630778493e0f2317f2f ./contracts/oracle/interfaces/IMedianizer.sol
934dc54da924aa1ec606e5a01e2321e5849e34aad2bebfb4d4d86f7aa4ffe6a0 ./contracts/mock/DToken.sol
5ecaff1748ae114132ce0c44fbb5a3518370f717da148aea4949790ef184285e ./contracts/mock/DTokenController.sol
```

dfef6d8a56ee3346c03dededfaa55f40acd1c850783cf52cb6c71d8b58aca537 ./contracts/converter/DFEngineV2.sol

82055aa1a82ac68313822e9f7671f9fc81a8bab151c7d7d0c42b4fea893396ab ./contracts/converter/DFProtocol.sol

707776a56744fe2f92134c711509a1a2d846efebf484b257b72cc90ad3a6db2d ./contracts/converter/DFSetting.sol

d02ce29c89627dd14167c9f56235664cac68342aa4bdbb7b1684b4df2f82e41b ./contracts/converter/DFProtocolView.sol

3b110f3b9d12e9fcc4528dfdce062559b8135c1a662804795c3322f03b215bfa ./contracts/converter/DFEngine.sol

f1f116f2bbf2799082ad5e968b59fd9b3d947f83014993b1036c68479d35d0ba ./contracts/converter/interfaces/IDFEngine.sol

a20ff99a0d662e9513b53e625d40534a0c1d0a324dca54009f3b17d88ecd09a2 ./contracts/converter/interfaces/IDFProtocol.sol

786e190f061c2513d989d07e78b943c6f015456bfffc278e4a30aeb9c8d05418 ./contracts/update/DFUpgrader.sol

26186305f92cd519981ae1b8dc0e4bc2b3747f0ce776fa799c74f57dab1597b9 ./contracts/token/Collaterals_t.sol

779b50781ebe08c52edb060695fa0e58e5f400ada1e7eab59bbcaa7119b203d6 ./contracts/token/ERC20SafeTransfer.sol

8597ea4792330560e729ac36c6a30c196f2797745ef2165b05cb34fb6fe7aad6 ./contracts/token/DSWrappedToken.sol

3f917bbac30d82507524731db1ab4d499aa46e1293568631c784fa334b0323d9 ./contracts/token/DSToken.sol

68a95d03f972f76c02227103e44cee99474f7c2a69688570aaeaefc0c0a1914f ./contracts/token/interfaces/IERC20.sol

97bbb637e892ebda3c018e19acbfae455354ec8eac04fb599d19921932266e92 ./contracts/token/interfaces/IERC20Token.sol

8c8f0e207fb0cb40251e9c98ca8c540ac2a45505e799a3c27ef7f7809bc0de91 ./contracts/token/interfaces/IDSToken.sol

35290746767029cdf1c903e384f94ef79182697f5a7c272280e7d0ab93fef711 ./contracts/token/interfaces/IDSWrappedToken.sol

Tests

5d4024573d1fa8b4745f0d7d03cf8c0917128ddc8eae71d2de628dc6904edf45 ./testUSR.js

7c7701e6a55b1435ba2e894d5a829ccf0b983e445034aa990b40340f10ea7b98 ./testInterestProvider.js

4aee16b8d9d2de924807971e553d5351ec172bed8e887345d2f4026321e96a2e ./test/DFEngine_deposit_claim.js

999af2167d33a96894c4f4bb0d25e86b8483cdd7f7ea9ab7e33b86cbb8d3baeb ./test/test0.1.js

eb4fb7ed61670a1c2ad0129365b1f2e9f14f2a76a5e6cf5c920fb798aae406f0 ./test/DFUpdateEngine.js

30a3ce623061a8d6d3fd7b7927086a6239b16f988c66919d00c5f40a81878fc7 ./test/testUSDxV2.js

4098ce7e3237759f7ed8cf31ac68a6059f94fe5628e5ea29d3fe90405a90689a ./test/DFEngine_random.js

e4090fab3224f1d007d703133e51f7860f566c6157503c8e42f51494cfdf3720 ./test/test0.3.js

637fd36cbc062e72c186bf164f542b25ba2e53aa171bc44de4a6229984f34790 ./test/test0.4.js

86b928258f711117bccf2f6bc89c613e8a6262ea92e0ba527b01745fa8a2c1c6 ./test/DFEngine_deposit_withdraw.js

c731dea5cee964f8baaaa94c181d3a5d0f273a1a3752e707f5b1ec4389e57094 ./test/DFEngine_deposit_withdraw_claim_destroy.js

2569ef91946bc675487fb2b25a1b822c164036bb14fe99f85bc47b6a44237a9b ./test/caseSourceFile.js

7499457b86996fe8e988bb0cc4a3d3a1da8b2f2269846b2444f1b187780feac5 ./test/DFEngine_deposit_claim_destroy.js

61a3d486406aa9d96a4f3c3fa509d87d39ca87b1dc62c71dace3fdc08817aaf5 ./test/DFEngine_deposit_withdraw_claim.js

a565d92797a91e9975a0e9c7cce648c6edc7cc9690f39054ea62c6a526ee98eb ./test/DFEngine_deposit_withdraw_destroy.js

46adbb323a7333ed2f978233be9f45f8b081d9127dbe6905ad6921a777e1cb5f ./test/DFEngineClaimAmount.js

f24769844c0cf5523512a8045d76bd03a7d30adcabfc2c7f554f423ef2f061dc ./test/DFEngine_deposit_destroy.js

6856e3316de8976f7fcbf22abacf5ccb798f8e233df29dbaa9ede320b14ca553 ./test/test0.2.js

0af14b211c91e312e6cb7a6f948e7e2c044ee4234fce3f0da99bb2fb88726dbf ./test/helpers/MathTool.js

632101e3b7e5886ea909a55924283f0a55fd23b1710d20f4b884c2bb5902fcae ./test/helpers/Utils.js

e288d530e9ea193ed4434afa5e7fd85fca95216d3d97fadd085b127390e6e77e ./test/helpers/migrate.js

8213fc0511c6c4c30072081b18cdfd7a2e2016cef29dc721f9c1bbe06f3d28f3 ./test/helpers/USDxV2deploy.js

707c250d8792fba10c4748322e2934aab1ece1022b1e16c9a96eb951d9028f31 ./test/helpers/DFEngine.js

b1fd8a5f68288252e1d020aca349a1893262621314 46df0508a3c17d82cd9474 ./test/helpers/DataMethod.js

ec77f33488ba6e5b6f8a9afcfe88adf6300a1ed9ec3f74ec00d7d068bf9b52b1 ./test/helpers/supportDToken.js

Changelog

- 2020-10-30 - Initial report
- 2020-11-17 - Revised report based on commit `83066e2`
- 2020-12-15 - Updating USDx_1.0 tests based on commit `b122a37`

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.