



May 11th 2021 – Quantstamp Verified

MakerDAO Liquidations 2.0

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

Executive Summary

Type	DeFi				
Auditors	Jake Goh Si Yuan, Senior Security Researcher Ed Zulkoski, Senior Security Engineer Sebastian Banescu, Senior Research Engineer				
Timeline	2021-02-10 through 2021-03-10				
EVM	Muir Glacier				
Languages	Solidity				
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review				
Specification	Liquidation Redesign Specifications				
Documentation Quality	Medium				
Test Quality	Undetermined				
Source Code	<table border="1"> <thead> <tr> <th>Repository</th> <th>Commit</th> </tr> </thead> <tbody> <tr> <td>dss(only clip.sol, dog.sol and abaci.sol)</td> <td>8aae83</td> </tr> </tbody> </table>	Repository	Commit	dss(only clip.sol, dog.sol and abaci.sol)	8aae83
Repository	Commit				
dss(only clip.sol, dog.sol and abaci.sol)	8aae83				



Total Issues	12 (3 Resolved)
High Risk Issues	0 (0 Resolved)
Medium Risk Issues	2 (1 Resolved)
Low Risk Issues	4 (2 Resolved)
Informational Risk Issues	5 (0 Resolved)
Undetermined Risk Issues	1 (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

Through this audit, we have uncovered 12 total issues ranging from **Medium** to **Informational** severity levels, and 1 of **Undetermined**. Overall, we found the code to be well reasoned, and mostly well validated with the exception of several authorized setter type functions (QSP-4, QSP-6) and some external call (QSP-5). That being said, we found that most of the issues arise from mis-set contract state variables. Though the risk is generally lower as they are modified through authorized functions, the potential damage would not be trivial. Therefore it would be prudent to employ cheap and simple validation to easily minimize the risk further.

We note that the specification delivered before the audit was comprehensive and should be held as a diamond standard to how protocols should document. We also note that given the unique nomenclature of the codebase, more effort should be put into easing the minds of unfamiliar readers, through more inline documentation wherever appropriate, such as specifying units of return and input.

Reaudit Update: The Maker team and the Quantstamp auditors had a meeting prior to the submission of the reaudit results by the former, where all the findings were discussed verbally. The Maker team made acknowledgements and gave verbal justifications for some of the findings. However, as the Maker team did not include the acknowledgement justifications in writing, we were unable to include it in the report.

ID	Description	Severity	Status
QSP-1	<code>Dog</code> and <code>Clipper</code> could potentially have mismatched components	^ Medium	Acknowledged
QSP-2	Misaligned incentives may encourage aberrant behavior	^ Medium	Mitigated
QSP-3	Division by zero	∨ Low	Fixed
QSP-4	Auction may halt if peek is zero	∨ Low	Fixed
QSP-5	Missing input validation	∨ Low	Acknowledged
QSP-6	Auction parameters may change mid flight	∨ Low	Acknowledged
QSP-7	Unlocked Pragma	○ Informational	Acknowledged
QSP-8	Privileged Roles and Ownership	○ Informational	Acknowledged
QSP-9	Contracts may have no authorized ward	○ Informational	Acknowledged
QSP-10	Tips may not be enough to cover gas fees when network is congested	○ Informational	Acknowledged
QSP-11	Oracle delay may lead to <code>redo</code> -ing and bad debt	○ Informational	Acknowledged
QSP-12	Incorrect initialization behavior	? Undetermined	Acknowledged

Quantstamp Audit Breakdown

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

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

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

Methodology

The Quantstamp auditing process follows a routine series of steps:

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

Toolset

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

Setup

Tool Setup:

- [Slither](#) v0.7.0

Steps taken to run the tools:

1. Installed the Slither tool: `pip install slither-analyzer`
2. Run Slither on the specific contract file: `slither ./ $PATH`

Findings

QSP-1 `Dog` and `Clipper` could potentially have mismatched components

Severity: *Medium Risk*

Status: Acknowledged

File(s) affected: `dog.sol`, `clip.sol`

Description: The `Dog` and `Clipper` contracts are designed to be components in a modular system that works together consistently to achieve certain functions. These contracts also interact with components other than `Dog` and `Clipper`, and it is expected that these components are the same from both `Dog` and `Clipper` perspectives. However, given that `vow` can be set quite trivially via `file` in both contracts, without any validation, there is potential that `vow` may diverge and therefore lead to aberrant unexpected behaviors. At the same time, it was also mentioned in the code walk-through call that the `Clipper.dog` state variable which is currently `immutable` may lose that status in the future, and therefore even components like `vat` could potentially be mismatched and lead to larger issues.

Recommendation: Ensure that `Clipper.vow` and `Clipper.dog.vow()` are the same by deriving one from the other, and adding validation to see if these state variables are in sync when a change is requested.

Update from the Maker team: "Not doing, authed function".

QSP-2 Misaligned incentives may encourage aberrant behavior

Severity: *Medium Risk*

Status: Mitigated

File(s) affected: `clip.sol`

Description: There are incentives `tip` and `chip` that are designed to encourage keepers to perform liquidations. However, given that there is a static component, `tip`, it might be possible in certain configurations that it is profitable for users to create unsafe vaults and liquidate it to receive the incentive, encouraging a highly aberrant behavior. This issue was also noted by the Maker team in [MIP45](#) under [MIP45c19 Incentive Farming](#).

Recommendation: Investigate and research into safe ratios between `ilk.dust`, `Clipper.tip` and `Clipper.chip` such that this aberrant behaviour will not be encouraged, and document it such that any potential parameter changes will be better informed.

Update from the Maker team: "Acknowledged, aware of incentive farming risk. Governance will need to manage this carefully. Risk team will likely simulate."

QSP-3 Division by zero

Severity: *Low Risk*

Status: Fixed

File(s) affected: `dog.sol`

Description: There is a primitive division operation inside the `bark` function which could lead to a "division by zero" error, because the divisor is not checked to be greater than zero, i.e. the 2nd division on L174: `dart = min(art, mul(room, WAD) / rate / milk.chop)`; where `milk.chop` could be zero if set by the `Dog.file` function.

Recommendation: Check that the value of `milk.chop` is greater than zero. Or prevent `chop` from being set to zero in the `Dog.file` function.

QSP-4 Auction may halt if peek is zero

Severity: *Low Risk*

Status: Fixed

File(s) affected: `clip.sol`

Description: There is no check that enforces the price returned by `pip.peek()` be greater than zero inside `Clipper.kick` and `Clipper.redo`. This would lead to setting the `top` value of an auction to zero, which is probably not desired. Having `top == 0` would cause a revert in the `rdiv(price, top)` inside the `Clipper.status` function, which would block access to calling `Clipper.redo` and `Clipper.take` again for `tail` amount of seconds.

Recommendation: Check that the `val` returned by `pip.peek()` is greater than zero inside `Clipper.kick` and `Clipper.redo`.

QSP-5 Missing input validation

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `dog.sol`, `clip.sol`, `abaci.sol`

Description: The majority of auth-ed functions do not have any input parameter validation in place, which could lead to unexpected values being set by authenticated accounts. The following list contains a few instances of such functions, however, the list is not exhaustive as it would be too long:

1. `Clipper.setBreaker` does not check the value of `level` and therefore the circuit breaker could be set to any unsigned integer value. In the current code, this is not a problem as the `isStopped` modifier uses the less-than (<) sign. However, it could be problematic if some service listens for `SetBreaker` events and it doesn't expect any values other than 0, 1 and 2.

2. `Clipper.file` does not check if the value of the `data` parameter corresponding to each value of what is in the right range. For example, `buf` seems to always need to be higher than 100%, while `chip` and `cusp` should always be less than 100%. `tail` should always be greater than zero.
3. `Dog.file` does not check if the value of the `data` parameter corresponding to each value of what is in the right range. For example, `chop` should probably never be 0; `Hole` and `hole` should also never be zero.
4. `LinearDecrease.file` does not check if the value of the `data` parameter corresponding to each value of what is in the right range. For example, `tau` should always be greater than 0, otherwise the `price` function will throw due to a division-by-zero error.
5. `StairstepExponentialDecrease.file` does not check if the value of the `data` parameter corresponding to each value of what is in the right range. For example, `step` should always be greater than 0, otherwise the `price` function will throw due to a division-by-zero error. Also `cut` should always be greater than 0, otherwise the `price` function will return 0.

Recommendation: Add input validation for all functions even if they are protected by the `auth` modifier, in order to prevent human-error.

Update: The `setBreaker` method in 1 was removed in the reaudit due to a design decision by the Maker team.

Update from the Maker team: "Acknowledged. Not Doing."

QSP-6 Auction parameters may change mid flight

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `clip.sol`

Description: There are several parameters in an auction that are deeply involved in tuning it, such as `cusp`, `tail`, `chip` or `tip`. In standard auctions, these parameters (or what they represent in terms of tuning) are usually set before the auction begins and held immutable until the end so that behaviour can be more predictable and fair to users and operators. However, it appears that in `Clipper`, all of these parameters can be mutated via `Clipper.file` at any time. This would mean that potentially an auction could be in flight and a subsequent parameter change in `file` abruptly stops the auction due to a `cusp` or `tail` decrement.

Recommendation: Consider moving parameters to a per-auction state, and disallow changes mid flight.

Update from the Maker team: "Acknowledged. Not Doing."

QSP-7 Unlocked Pragma

Severity: *Informational*

Status: Acknowledged

File(s) affected: `All contracts`

Description: Every Solidity file scoped specifies in the header a version number of the format `pragma solidity >=0.6.11`. The greater-or-equal (`>=`) 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 from the Maker team: "Acknowledged. Not Doing."

QSP-8 Privileged Roles and Ownership

Severity: *Informational*

Status: Acknowledged

File(s) affected: `dog.sol`, `clip.sol`

Description: Smart contracts will often have `owner` variables to designate the person with special privileges to make modifications to the smart contract. In these contracts we have a homemade authentication system that guards key functions such as `file`, `cage`, `yank` or `setBreaker`. Whilst it is understood these contracts are designed with a multilateral governance system in mind for the privileged role, it is important to note the potential impact.

Recommendation: This centralization of power needs to be made clear to the users, especially depending on the level of privilege the contract allows to the owner.

Update from the Maker team: "Acknowledged. Not Doing."

QSP-9 Contracts may have no authorized ward

Severity: *Informational*

Status: Acknowledged

File(s) affected: `All contracts`

Description: There is nothing preventing the single authorized address/user from calling `deny()` on themselves. This would cause the user to lock themselves out since they will no longer be able to call any function protected by the `auth` modifier in the smart contract.

Recommendation: Keep track of the number of authorized users and revert calls to deny if there is a single authorized user.

Update from the Maker team: "Acknowledged. Intended behavior."

QSP-10 Tips may not be enough to cover gas fees when network is congested

Severity: *Informational*

Status: Acknowledged

File(s) affected: `clip.sol`

Description: The purpose of the `tip` state variable is to incentivize keepers to call `bark` such that they would be reimbursed for the gas fee. However, since this is a flat fee, it may not cover the

gas costs when the network is highly congested and the gas prices are high.

Recommendation: Compute the `tip` using the current gas price when the liquidation call is made. This could be done with the `GASPRICE` opcode proposed in EIP-1559. Until EIP-1559 is implemented, it is not straightforward to compute the current gas price without an external oracle such as ETH Gas Station. However, such oracles could be DDoSed as we have seen on Feb 23rd, 2021.

Update from the Maker team: "Acknowledged. Governance will manage."

QSP-11 Oracle delay may lead to `redo`-ing and bad debt

Severity: *Informational*

Status: Acknowledged

File(s) affected: `clip.sol`

Description: In `Clipper`, the auction sets the `top` price as a function of the OSM result. It is stated in the MIP45 specification that:

Note that the current OSM price is between one and two hours delayed relative to the actual market price.

This means that there is a possibility that the actual market price could be sufficiently under the OSM price that the Keepers have no incentive `take`, therefore leading to incomplete auctions that `redo`.

Every `redo` incentivizes callers with `tip` and `chip` which is funded from `vat`, thereby draining it slowly. Auctions which are incomplete for longer amounts of time run a higher risk of accumulating bad debt. Combined, they may cause a persistent overall value loss. The potential loss is scaled by the length of the OSM delay, since this plausible event is bounded by it. This issue was foreshadowed by [MIP45c21](#) and [MIP45c23](#) of the specification, and can be thought of as similar but not the same due to the focus on the oracle delay.

Recommendation: It might be useful to further document and elaborate on how delay lengths are chosen, and how they relate to `cusp` and `tail` settings, as these information would be extremely valuable for users, especially in times of chaotic market movements.

Update from the Maker team: "Acknowledged. Not doing."

QSP-12 Incorrect initialization behavior

Severity: *Undetermined*

Status: Acknowledged

File(s) affected: `clip.sol`, `abaci.sol`

Description:

- [ACKNOWLEDGED] The specification indicates that the `Clipper.dog` should be authorized to call `kick`. However, the `Clipper.constructor` does not set `wards[dog] = 1`.
- [MITIGATED] The `cut` in the `StairstepExponentialDecrease.constructor` is set to zero. This is not good, because it will lead to the `price` function returning zero. It is also not inline with the specification, which gives an example of `cut = 0.99 * RAY`.

Recommendation:

- Set `wards[dog] = 1` in the `Clipper.constructor`.
- Initialize the value of `cut` to something other than 0.

Update from the Maker team:

- "Acknowledged. Not doing."
- "Added comment to constructor".

Automated Analyses

Slither

There were 19 results uncovered via Slither on the three contracts, and we checked through all of them and found them to be false positives.

Code Documentation

- [MITIGATED] Every function should at least have a short description of its purpose, input parameters and output value. This is not the case with the majority of function in the code base.
- [UNRESOLVED] Each function that returns `uint256` values or which has input parameters of type `uint256` should indicate the prevision its return value and input parameters are expecting, i.e. `WAD`, `RAY` or `RAD`. This would greatly facilitate code maintainability and auditability.
- [FIXED] In `dog.sol::bark` it appears that there are scenarios where any combination of `Hole < Dirt`, `milk.hole < milk.dirt` and `room < dust` then liquidation throws and is therefore not possible. More user-facing documentation should be provided on this to clarify when these scenarios could occur.
- [UNRESOLVED] It is possible to encounter situations where the price of the collateral in a CDP would drop faster than the price decrease function used in the Dutch-auction. To counter this attack it appears that the `tail` and `cusp` state variables are used. However, it is not clear who is in charge of tuning these values and how they will be set in a time-critical situation. We strongly believe that more clarity should be brought to this in the form of expanded documentation.

Adherence to Best Practices

1. [FIXED] In `dog.sol::L54 mapping (address => uint) public wards;` should be `uint256` instead of `uint` to be consistent with the rest of the codebase.
2. [UNRESOLVED] In `Clipper.take` the primitive subtraction operator is used instead of `Clipper.sub` as expected.
3. [UNRESOLVED] Code clones should be avoided. In this repo, the code for authorization and safe arithmetic is cloned and duplicated in every file, which decreases maintainability of the code.

Test Results

Test Suite Results

The test suite ran successfully without errors, and the tests correlated with their titles.

```
Running 2 tests for src/test/abaci.t.sol:ClipperTest
[PASS] test_linear_decrease() (gas: 510772)
[PASS] test_stairstep_exp_decrease() (gas: 41589578)

Running 40 tests for src/test/clip.t.sol:ClipperTest
[PASS] test_bark_only_leaving_dust_over_hole_rate() (gas: 456805)
[PASS] test_take_bid_fails_no_partial_allowed() (gas: 1275752)
[PASS] test_redo_zero_usr() (gas: 27959)
[PASS] test_auction_reset_tail() (gas: 1030851)
[PASS] test_kick_zero_usr() (gas: 5577)
[PASS] test_kick_zero_tab() (gas: 5538)
[PASS] test_stopped_take() (gas: 1178319)
[PASS] test_partial_liquidation_hole_limit() (gas: 410556)
[PASS] test_take_over_tab() (gas: 1176367)
[PASS] test_Clipper_yank() (gas: 1090425)
[PASS] test_kick_zero_lot() (gas: 5561)
[PASS] testFail_stopped_auction_reset_tail() (gas: 736640)
[PASS] test_partial_liquidation_hole_limit() (gas: 411082)
[PASS] test_bark_not_leaving_dust() (gas: 403897)
[PASS] test_take_zero_usr() (gas: 1049321)
[PASS] test_Hole_hole() (gas: 4861258)
[PASS] test_take_at_tab() (gas: 1176317)
[PASS] test_auction_reset_cusp_twice() (gas: 978331)
[PASS] testFail_take_impersonation() (gas: 1412241)
[PASS] test_flashsale() (gas: 1417725)
[PASS] testFail_stopped_take() (gas: 1066079)
[PASS] testFail_stopped_kick() (gas: 223054)
[PASS] testFail_take_bid_too_low() (gas: 1051111)
[PASS] test_setBreaker() (gas: 28545)
[PASS] test_bark_not_leaving_dust_over_hole() (gas: 403962)
[PASS] test_take_under_tab() (gas: 1158628)
[PASS] test_kick() (gas: 940392)
[PASS] test_take_multiple_bids_different_prices() (gas: 1321090)
[PASS] testFail_reentrancy_redo() (gas: 1090069)
[PASS] test_redo_incentive() (gas: 1700977)
[PASS] test_take_bid_recalculates_due_dust() (gas: 1164519)
[PASS] test_stopped_auction_reset_tail() (gas: 1029502)
[PASS] test_auction_reset_cusp() (gas: 1032837)
[PASS] test_kick_basic() (gas: 171079)
[PASS] test_bark_not_leaving_dust_rate() (gas: 456883)
[PASS] testFail_not_enough_dai() (gas: 1087974)
[PASS] testFail_reentrancy_take() (gas: 1090268)
[PASS] test_get_chop() (gas: 10328)
[PASS] test_take_bid_avoids_recalculate_due_no_more_lot() (gas: 1175249)
[PASS] test_auction_reset_tail_twice() (gas: 977336)

Running 7 tests for src/test/dog.t.sol:DogTest
[PASS] test_bark_equals_ilk_hole_plus_dust() (gas: 398981)
[PASS] test_bark_equals_Hole_plus_dust() (gas: 398477)
[PASS] test_bark_basic() (gas: 360444)
[PASS] test_bark_unliquidatable_vault() (gas: 384049)
[PASS] test_bark_over_ilk_hole_under_ilk_hole_plus_dust() (gas: 398953)
[PASS] testFail_bark_not_unsafe() (gas: 225388)
[PASS] test_bark_over_Hole_under_Hole_plus_dust() (gas: 398493)
```

Code Coverage

Unfortunately, there is currently no easy way to retrieve code coverage results from a dapptools project right now. We have created a [feature request](#), and until then there will be no way of getting code coverage data without spending an excessive amount of time.

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

`b464cd5324680514adcdca23752cd6b474626108bb8d9c8fa282ed3b78598008` `./src/abaci.sol`

`3ff088efda5c580f1f7cbe32effabe89de822efcbe54eb9a75000a2248e3846` `./src/dog.sol`

`349b80e72735d55c9f22821b6f973687a119080bea176d15bf70e3950e4460eb` `./src/clip.sol`

Tests

`2c2049ae95cb164ec2c3aea5a6b2048f8b869f9c6c08e199f7e0c93f9a9ca60a` `./test/dog.t.sol`

`745d5594d49fa28b11b765c60f1f56abda22d4fd576ad8b9d29e16e4262e7220` `./test/abaci.t.sol`

`cc55f454abe4e3a516e36cec4c27c2076d42f9eebf8cb7a19b91ced6c1420cd3` `./test/clip.t.sol`

Changelog

- 2021-02-25 - Initial report
- 2021-03-08 - Reaudit report, switching from commit `c8a1344` to commit `a4759e`
- 2021-03-10 - Update commit from previous to `8aae83` and updating issues with response from Maker team.

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.