Quantstamp Security Assessment Certificate

June 29th 2022 — Quantstamp Verified

Origami (Governance & Membership tokens)

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

Executive Summary

Type	DAO/Governance (ERC20) and membership (ERC721) tokens
Auditors	Roman Rohleder, Research Engineer Bohan Zhang, Auditing Engineer Fatemeh Heidari, Security Auditor
Timeline	2022-06-17 through 2022-06-20
EVM	Arrow Glacier
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	None
Documentation Quality	Medium
Test Quality	High

Source Code

Repository	Commit
crane	33dd352
crane	7a837c2

Total Issues	**5** (4 Resolved)
High Risk Issues	0 (0 Resolved)
Medium Risk Issues	0 (0 Resolved)
Low Risk Issues	**2** (1 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).
Fixed	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

Quantstamp audited the Origami governance (ERC20) and membership (ERC721) tokens. Only informational and low severity issues were found, besides some best practices. Nonetheless, we recommend addressing all points before deploying in production. The test suite reached almost ideal 100% for statement and branch coverage, only missing one function to be tested. We recommend adding corresponding tests.
Update: For the re-audit all issues have been addressed, as well as all best practice-related findings. Further, the test suite was improved, now reaching 100% statement and branch coverages and thorough NatSpec documentation inside the code was added.

ID	Description	Severity	Status
QSP-1	Missing Input Validation	⌄ Low	Fixed
QSP-2	Privileged Roles and Ownership	⌄ Low	Acknowledged
QSP-3	Unlocked Pragma	○ Informational	Fixed
QSP-4	Application Monitoring Can Be Improved by Emitting More Events	○ Informational	Fixed
QSP-5	Block Timestamp Manipulation	○ Informational	Fixed

Quantstamp Audit Breakdown

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

DISCLAIMER:
Files in the sub-directory `./contracts/versions/*` were out-of-scope for this audit.

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

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

Methodology

The Quantstamp auditing process follows a routine series of steps:

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

Toolset

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

Setup

Tool Setup:

- Slither v0.8.2

Steps taken to run the tools:

1. Install the Slither tool: `pip3 install slither-analyzer`
2. Run Slither from the project directory: `slither .`

Findings

QSP-1 Missing Input Validation

Severity: *Low Risk*

Status: Fixed

File(s) affected: `contracts/OrigamiGovernanceToken.sol`, `contracts/OrigamiMembershipToken.sol`

Related Issue(s): SWC-123

Description: It is important to validate inputs, even if they only come from trusted addresses, to avoid human error. Specifically, in the following functions arguments of type `address` may be initialized with value `0x0`:

1. `OrigamiGovernanceToken.initialize()`
2. `OrigamiMembershipToken.initialize()`

Recommendation: We recommend adding the relevant checks.

Update: Fixed in commit https://github.com/JoinOrigami/crane/tree/130cb13af8b2b72f19b8a8ec0392cdc7c003ce7e, by adding corresponding `address(0)` checks, as suggested.

QSP-2 Privileged Roles and Ownership

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `contracts/OrigamiGovernanceToken.sol`, `contracts/OrigamiGovernanceTokenFactory.sol`, `contracts/OrigamiMembershipToken.sol`, `contracts/OrigamiMembershipTokenFactory.sol`

Description: Smart contracts will often have `owner` variables to designate the person with special privileges to make modifications to the smart contract.
The `OrigamiGovernanceToken.sol` contract contains the following privileged roles:

- `DEFAULT_ADMIN_ROLE`, as initialized during the `initialize()` function to the `_admin` address parameter:
  . **Renounce his role and thereby disable all followingly listed actions, by calling** `renounceRole()`.
  . Add/Remove arbitrary addresses from the used roles (`DEFAULT_ADMIN_ROLE`, `PAUSER_ROLE` and `MINTER_ROLE`), by calling `grantRole()`/`revokeRole()`.
  . Enable/Disable the burning of tokens, by calling `enableBurn()`/`disableBurn()`.
  . Enable/Disable the transfer of tokens, by calling `enableTransfer()`/`disableTransfer()`.

- `PAUSER_ROLE`, as initialized during the `initialize()` function to the `_admin` address parameter:
  . Renounce his role and thereby disable all followingly listed actions, by calling `renounceRole()`.
  . Transition the contract in and out of the paused state (impacting the ability to transfer or burn tokens), by calling `pause()`/`unpause()`.

- `MINTER_ROLE`, as initialized during the `initialize()` function to the `_admin` address parameter:
  . Renounce his role and thereby disable all followingly listed actions, by calling `renounceRole()`.
  . Mint an arbitrary amount of tokens to any address, by calling `mint()` (**Note: As only the holder can burn ones tokens, a malicious minter could mint an arbitrary amount of tokens to himself and retain them**).

The `OrigamiGovernanceTokenFactory.sol` contract contains the following privileged roles:

- `DEFAULT_ADMIN_ROLE`, as initialized during the `initialize()` function to the `msg.sender`:
  . Renounce his role and thereby disable all followingly listed actions, by calling `renounceRole()`.
  . Add/Remove arbitrary addresses from the `DEFAULT_ADMIN_ROLE` role, by calling `grantRole()`/`revokeRole()`.
  . Create new Origami governance token clones, by calling `createOrigamiGovernanceToken()`.
  . Retrieve deployed contract addresses by index on-chain (Note: The corresponding array `proxiedContracts[]` is readable off-chain by anyone), by calling `getProxyContractAddress()`.

- The implicit role of the `TransparentUpgradeableProxy` admin, as set as the message sender, when creating the proxy:
  . **Renounce his role and thereby disable all followingly listed actions, by calling** `changeAdmin()` **with an uncontrolled address**.
  . Upgrade the proxy, by calling `upgradeTo()`/`upgradeToAndCall()`.
  . Change the current admin role to an arbitrary other address, by calling `changeAdmin()` (**Note: Even when the initial admin gets stripped of his** `DEFAULT_ADMIN_ROLE` **role, he may independently still remain admin of the proxy**).

The `OrigamiMembershipToken.sol` contract contains the following privileged roles:

- `DEFAULT_ADMIN_ROLE`, as initialized during the `initialize()` function to the `_admin` address parameter:
  . **Renounce his role and thereby disable all followingly listed actions, by calling** `renounceRole()`.
  . Add/Remove arbitrary addresses from the used roles (`DEFAULT_ADMIN_ROLE`, `PAUSER_ROLE`, `MINTER_ROLE` and `REVOKER_ROLE`), by calling `grantRole()`/`revokeRole()`.
  . Change the tokens base URI, by calling `setBaseURI()`.
  . Enable/Disable the transfer of tokens, by calling `enableTransfer()`/`disableTransfer()`.

- `PAUSER_ROLE`, as initialized during the `initialize()` function to the `_admin` address parameter:
  . Renounce his role and thereby disable all followingly listed actions, by calling `renounceRole()`.
  . Transition the contract in and out of the paused state (impacting the ability to transfer tokens), by calling `pause()`/`unpause()`.

- `MINTER_ROLE`, as initialized during the `initialize()` function to the `_admin` address parameter:
  . Renounce his role and thereby disable all followingly listed actions, by calling `renounceRole()`.

-
  . Mint tokens to any address, by calling `safeMint()`.

- `REVOKER_ROLE`, as initialized during the `initialize()` function to the `_admin` address parameter:
  . Renounce his role and thereby disable all followingly listed actions, by calling `renounceRole()`.
  . Burn/Revoke the token of any address, by calling `revoke()`.

The `OrigamiMembershipTokenFactory.sol` contract contains the following privileged roles:

- `DEFAULT_ADMIN_ROLE`, as initialized during the `initialize()` function to the `msg.sender`:
  . Renounce his role and thereby disable all followingly listed actions, by calling `renounceRole()`.
  . Add/Remove arbitrary addresses from the `DEFAULT_ADMIN_ROLE` role, by calling `grantRole()`/`revokeRole()`.
  . Create new Origami governance token clones, by calling `createOrigamiMembershipToken()`.
  . Retrieve deployed contract addresses by index on-chain (Note: The corresponding array `proxiedContracts[]` is readable off-chain by anyone), by calling `getProxyContractAddress()`.
- The implicit role of the `TransparentUpgradeableProxy` admin, as set as the message sender, when creating the proxy:
  . **Renounce his role and thereby disable all followingly listed actions, by calling `changeAdmin()` with an uncontrolled address**.
  . Upgrade the proxy, by calling `upgradeTo()`/`upgradeToAndCall()`.
  . Change the current admin role to an arbitrary other address, by calling `changeAdmin()` (**Note: Even when the initial admin gets stripped of his `DEFAULT_ADMIN_ROLE` role, he may independently still remain admin of the proxy**).

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: Acknowledged with: `This is an intentional decision by Origami Inc to enable us to manage contracts on behalf of our customers. We disclose to them during onboarding the particulars of our access control patterns and work with them to decide the best strategy for management for their org. We have increased our documentation (as of 7f5dead1db59c2965f9592ea955ffe3ada478c44) around what roles are capable of and endeavor to assign these roles to their DAO and renounce them from the admin address which we control wherever possible. It is entirely possible for Origami to cede all access to the DAO.`

QSP-3 Unlocked Pragma

Severity: *Informational*

Status: Fixed

File(s) affected: `contracts/*`,

Related Issue(s): SWC-103

Description: Every Solidity file specifies in the header a version number of the format `pragma solidity (^)0.8.*`. 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, we recommend to remove the caret to lock the file onto a specific Solidity version.

Update: Fixed in commit https://github.com/JoinOrigami/crane/tree/45df6c39f2a1c76fa4905e8eeafe9c6b17ac27b4, by binding the solidity pragma version to `0.8.9`, as suggested.

QSP-4 Application Monitoring Can Be Improved by Emitting More Events

Severity: *Informational*

Status: Fixed

File(s) affected: `contracts/OrigamiGovernanceToken.sol`, `contracts/OrigamiGovernanceTokenFactory.sol`, `contracts/OrigamiMembershipToken.sol`, `contracts/OrigamiMembershipTokenFactory.sol`

Description: In order to validate the proper deployment and initialization of the contracts, it is a good practice to emit events. Also, any important state transition can be logged, which is beneficial for monitoring the contract, and also tracking eventual bugs, or hacks. Below we present a non-exhaustive list of events that could be emitted to improve the application management:

1. `OrigamiGovernanceToken.enableBurn()` does not emit events reflecting changes made to state variable `_burnEnabled`.
2. `OrigamiGovernanceToken.disableBurn()` does not emit events reflecting changes made to state variable `_burnEnabled`.
3. `OrigamiGovernanceToken.enableTransfer()` does not emit events reflecting changes made to state variable `_transferEnabled`.
4. `OrigamiGovernanceToken.disableTransfer()` does not emit events reflecting changes made to state variable `_transferEnabled`.
5. `OrigamiGovernanceTokenFactory.createOrigamiGovernanceToken()` does not emit events reflecting changes made to state variable `proxiedContracts`.
6. `OrigamiMembershipToken.setBaseURI()` does not emit events reflecting changes made to state variable `_metadataBaseURI`.
7. `OrigamiMembershipToken.enableTransfer()` does not emit events reflecting changes made to state variable `_transferEnabled`.
8. `OrigamiMembershipToken.disableTransfer()` does not emit events reflecting changes made to state variable `_transferEnabled`.
9. `OrigamiMembershipTokenFactory.createOrigamiMembershipToken()` does not emit events reflecting changes made to state variable `proxiedContracts`.

Recommendation: Consider emitting the events.

Update: All fixed in commit https://github.com/JoinOrigami/crane/tree/32c73c2b6b3701714086f676f8fcedc280b43bf1, by adding and emitting corresponding events, as suggested.

QSP-5 Block Timestamp Manipulation

Severity: *Informational*

Status: Fixed

File(s) affected: `contracts/OrigamiMembershipToken.sol`

Related Issue(s): SWC-116

Description: Projects may rely on block timestamps for various purposes. However, it's important to realize that miners individually set the timestamp of a block, and attackers may be able to manipulate timestamps for their own purposes to up to 900 seconds. If a smart contract relies on a timestamp, it must take this into account.

Recommendation: Clarify to users that a time difference up to 900 seconds for the `tokenIdToBlockTimestamp` logging does not impact the intended operations.

Update: Fixed in commit https://github.com/JoinOrigami/crane/tree/3af1881a53c8555274 1e5a031fae149d24758d6d, by commenting said potential block timestamp variability in code.

Automated Analyses

Slither

Slither reported 199 findings. Some of which were integrated into the report, while others have been dismissed as false positives.

Adherence to Best Practices

1. In accordance with best practices, code should follow a common style guide. In this regard, the following instance should be adjusted accordingly:
    1. `OrigamiGovernanceToken.whenNontransferrable()`: Not in `mixedCase`. (**Update**: Acknowledged)
    2. `OrigamiMembershipToken.whenNontransferrable()`: Not in `mixedCase`. (**Update**: Acknowledged)

2. Adding purpose documentation for each function and variable clarification would help readers better understand the purpose and functionality. It will also help developers better maintain the code. (**Update**: Fixed)
3. As variable `proxiedContracts` is publicly readable anyway and non-security sensitive, it is advised to remove the modifer `onlyRole(DEFAULT_ADMIN_ROLE)` for function `getProxyContractAddress()` in contracts `OrigamiGovernanceTokenFactory.sol` and `OrigamiMembershipTokenFactory.sol`. (**Update**: Fixed)
4. In the function `initialize()`, the contract first grants the message sender the admin role, and later revokes it. Deleting them would not affect anything, as the internal function `_grantRole()` does not check the callers role. Therefore we suggest deleting L53 and L63 in `OrigamiMembershipToken.sol`, and L43 and L51 in `OrigamiGovernanceToken.sol`. (**Update**: Fixed)

Test Results

Test Suite Results

All of the 66 provided tests were successfully executed and passed.

Update: For the re-audit 17 additional tests have been added. Of the now 83 tests all passed.

```
GovernanceToken
  initializing
     ✓ reverts when initialized with the zero address as the admin (330ms)
  upgrading
     ✓ reverts when upgrading to the zero address (1132ms)
  minting
     ✓ reverts when minting with the zero address (40ms)
     ✓ emits an event when minting occurs (38ms)
  Limited Supply
     ✓ should revert mint when caller does not have proper role (133ms)
     ✓ should mint when caller has proper role (54ms)
     ✓ should revert when we attempt to mint more than the total supply's worth (68ms)
  Burning Tokens
     ✓ emits a BurnEnabled event when burning is enabled
     ✓ emits a BurnEnabled event when burning is disabled
     ✓ reverts if non-admin tries to set enableBurn (53ms)
     ✓ reverts if non-admin tries to set disableBurn (38ms)
     ✓ allows admin to set enableBurn
     ✓ allows admin to set disableBurn
     ✓ prevents burning when not enabled
     ✓ prevents calling enableBurn when already enabled
     ✓ allows burning when enabled (41ms)
     ✓ allows burning from a wallet we have an allowance for when burning is enabled (47ms)
     ✓ prevents burning more than allowed from a wallet we have an allowance for when burning is enabled (46ms)
  Pausing
     ✓ only allows PAUSER to set pause (44ms)
     ✓ only allows PAUSER to set unpause (46ms)
     ✓ prevents minting when paused
     ✓ prevents burning when paused
     ✓ prevents transfers when paused (41ms)
     ✓ prevents TRANSFERRER_ROLE transfers when paused (40ms)
     ✓ allows burning when unpaused and burnable
     ✓ allows minting when unpaused
     ✓ allows transfers when unpaused (43ms)
  Transferrability
     ✓ emits a TransferEnabled event when transfer is enabled
     ✓ emits a TransferEnabled event when transfer is disabled
     ✓ only allows admin to set transferrable (51ms)
     ✓ only allows admin to set nontransferrable (46ms)
     ✓ prevents calling enableTransfer when already enabled
     ✓ allows mint when transfer is disabled
     ✓ prevents burn when transfer is disabled
     ✓ prevents (non-MINTER) transfers when transfer is disabled
     ✓ allows transfers when transfer is enabled (74ms)
     ✓ allows TRANSFERRER_ROLE to transfer when transfer is disabled (38ms)
     ✓ allows TRANSFERRER_ROLE to transfer when transfer is enabled (49ms)

OrigamiGovernanceTokenFactory
  Deploying
     ✓ Created the KIDA token (40ms)
     ✓ Created the OKC token
  AccessControl for deployed instances
     ✓ allows admin to grant roles
  Upgrading the implementation for clones
     ✓ reverts when you try to access a proxy that does not exist
     ✓ allows non-admin to retrieve proxy addresses
     ✓ has access to the old functions
     ✓ reflects changes in the upgraded contract
     ✓ new factory instances of the proxy have to be upgraded independently

MembershipToken
  initializing
     ✓ reverts when initialized with the zero address as the admin (56ms)
  upgrading
     ✓ reverts when upgrading to the zero address (310ms)
  minting
     ✓ mints membership NFTs
     ✓ limits minting to one NFT per address
     ✓ emits a Mint event
  silly coverage tests
     ✓ does nothing but delegate the implementation of supportsInterface
  metadata
     ✓ reverts when attempting to fetch tokenId 0
     ✓ reverts when attempting to fetch an unminted tokenId
     ✓ emits an event when the base URI is changed
     ✓ allows the admin to set a base URI
```

```
      ✓ reverts when a non-admin tries to setBaseURI
    pausing
      ✓ allows the admin to pause
      ✓ reverts when a non-admin tries to pause
      ✓ allows the admin to unpause
      ✓ reverts when a non-admin tries to unpause
      ✓ prevents minting when paused
    transferrability
      ✓ emits an event when transfer is enabled
      ✓ emits an event when transfer is enabled
      ✓ allows minter to transfer (aka mint) when nontransferrable (45ms)
      ✓ prevents token transfers when disabled (42ms)
      ✓ prevents a single address from being transferred more than one token (47ms)
      ✓ allows token transfers when enabled (94ms)
      ✓ only allows the owner to transfer when transferrable
      ✓ prevents enabling transfers when they're already enabled
      ✓ allows disabling transfers after they've been enabled
    revoking
      ✓ allows admin to revoke (69ms)
      ✓ allows REVOKER to revoke
      ✓ reverts when non-admin tries to revoke
      ✓ reverts when from address does not own a token

  OrigamiMembershipTokenFactory
    Deploying
      ✓ Created the ADM token (43ms)
      ✓ Created the BBQ token (42ms)
    AccessControl for deployed instances
      ✓ allows admin to grant roles
    Upgrading
      ✓ reverts when you try to access a proxy that does not exist
      ✓ allows a non-admin to retrieve proxy addresses
      ✓ has access to the old functions
      ✓ reflects changes in the upgraded contract
      ✓ new factory instances of the proxy have to be upgraded independently
```

Code Coverage

Of the audited files, the statement and branch coverages were almost at an ideal 100%. The only untested/uncovered function was `OrigamiMembershipToken.supportsInterface()`. We recommend adding tests that cover this function as well.

Update: After the re-audit the statement and branch coverages have been improved to be 100%, covering all present code.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	100	100	100	100	
OrigamiGovernanceToken.sol	100	100	100	100	
OrigamiGovernanceTokenFactory.sol	100	100	100	100	
OrigamiMembershipToken.sol	100	100	100	100	
OrigamiMembershipTokenFactory.sol	100	100	100	100	
All files	**100**	**100**	**100**	**100**	

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

```
ce6b7fbd42cf3acaa2c1a7cac28e91fa01f79632ee9043ad8fc525dfd7b181cc ./contracts/OrigamiMembershipToken.sol
d5095c2f754209e4c9d089e62f547c237b41c5aea747461cb76fcc83683d8bbc ./contracts/OrigamiGovernanceToken.sol
2c6091290bafe0fac8e2eb359c831d12af58a9ddf290e1bd9da598a5a3ab17f6 ./contracts/OrigamiMembershipTokenFactory.sol
ed0b90dcd906e034587223fca85ee90bea0722f4c5faeed7546f58ccaf7a561c ./contracts/OrigamiGovernanceTokenFactory.sol
```

Tests

```
ae9a2889ed83bf6791e3ad3c9509331d80e6cf2a3a29be55ef8ff3fc86a65327 ./test/OrigamiGovernanceTokenFactory.test.ts
7dfa64f31bab74de9e759d11c3ce960501af082d54e28786a42617d489d2ca70 ./test/OrigamiMembershipTokenFactory.test.ts
8cc2786aa3be5b56f71271f4a859bca2f77b3a38a7c991bbb059b6e78d25eee9 ./test/OrigamiMembershipToken.test.ts
1d501329e8690f8464b37b3a886cc169ab3744fba5ebe50d0dfcfecc4bc677c4 ./test/OrigamiGovernanceToken.test.ts
```

Changelog

- 2022-06-20 - Initial report
- 2022-06-29 - Final report

About Quantstamp

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

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

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

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