Quantstamp Security Assessment Certificate

February 18th 2022 — Quantstamp Verified

Fuse Flywheel V2

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

Executive Summary

Type	Ethereum
Auditors	Ed Zulkoski, Senior Security Engineer Fayçal Lalidji, Security Auditor
Timeline	2022-02-14 through 2022-02-22
EVM	Muir Glacier
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	None
Documentation Quality	High
Test Quality	Medium

Source Code

Repository	Commit
flywheel-v2	1b7ec61

Total Issues	**6** (0 Resolved)
High Risk Issues	0 (0 Resolved)
Medium Risk Issues	**1** (0 Resolved)
Low Risk Issues	**5** (0 Resolved)
Informational Risk Issues	0 (0 Resolved)
Undetermined Risk Issues	0 (0 Resolved)

High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or is reasonably likely to lead to moderate financial impact.
Low Risk	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low-impact in view of the client's business circumstances.
Informational	The issue does not post an immediate risk, but is relevant to security best practices or Defence in Depth.
Undetermined	The impact of the issue is uncertain.
Unresolved	Acknowledged the existence of the risk, and decided to accept it without engaging in special efforts to control it.
Acknowledged	The issue remains in the code but is a result of an intentional business or design decision. As such, it is supposed to be addressed outside the programmatic means, such as: 1) comments, documentation, README, FAQ; 2) business processes; 3) analyses showing that the issue shall have no negative consequences in practice (e.g., gas analysis, deployment settings).
Resolved	Adjusted program implementation, requirements or constraints to eliminate the risk.
Mitigated	Implemented actions to minimize the impact or likelihood of the risk.

Summary of Findings

During the audit, a total of 6 issues were found ranging from Medium to Low severity. Of note, we recommend against allowing any address to claim on behalf of another, which could be detrimental to external contracts interacting with the system. Further, there is limited developer documentation describing how to execute the test suite, and several tests failed during our attempts. We suggest addressing all issues and adding developer testing documentation before using the code in production.

ID	Description	Severity	Status
QSP-1	Any address can claim tokens on behalf of another	^ Medium	Unresolved
QSP-2	Use of `transfer`	v Low	Unresolved
QSP-3	Missing Input Validation	v Low	Unresolved
QSP-4	Rewards may be less than expected if contract does not have enough funds	v Low	Unresolved
QSP-5	No coverage scripts	v Low	Unresolved
QSP-6	Failing tests and missing test suite documentation	v Low	Unresolved

Quantstamp Audit Breakdown

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

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

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

Methodology

The Quantstamp auditing process follows a routine series of steps:

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

Toolset

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

Setup

Tool Setup:

- Slither v0.8.3
- Mythril v0.2.7

Steps taken to run the tools:

1. Installed the Slither tool: `pip install slither-analyzer`
2. Run Slither from the project directory: `slither .`
3. Installed the Mythril tool from Pypi: `pip3 install mythril`
4. Ran the Mythril tool on each contract: `myth -x path/to/contract`

Findings

QSP-1 Any address can claim tokens on behalf of another

Severity: Medium Risk

Status: Unresolved

Description: `FlyWheelCore.claim()` allows any address to claim tokens on behalf of another user without pre-authorization. Multiple risks exist in this case, especially if the reward is claimed on behalf of a contract that has a specific implementation that won't allow it to withdraw its reward asset balance.

Recommendation: Confirm if this is intended behavior or restrict the function access following the application design.

QSP-2 Use of `transfer`

Severity: Low Risk

Status: Unresolved

File(s) affected: `FlywheelCore.sol`

Description: The function `claim` invokes `rewardToken.transfer(owner, accrued)`. First, the return value should be checked to be `true`, in order to ensure the function succeeded. Second, since some tokens may not strictly adhere to the ERC20 interface, particularly by not having a boolean return value, an approach similar to SafeERC20 should be used.

Recommendation: Perform safety checks on `transfer` calls.

QSP-3 Missing Input Validation

Severity: Low Risk

Status: Unresolved

File(s) affected: `FlywheelCore.sol`, `FlywheelDynamicRewards.sol`

Description: In order to mitigate issues such as faulty deployments, the following sanity checks should be added:

1. `FlywheelCore.constructor` should check that all address arguments are non-zero.
2. `FlywheelDynamicRewards.construtor` should check that all address arguments are non-zero.
3. `FlywheelStaticRewards.construtor` should check that all address arguments are non-zero.

Recommendation: Add the aforementioned checks as `require` statements.

QSP-4 Rewards may be less than expected if contract does not have enough funds

Severity: Low Risk

Status: Unresolved

File(s) affected: `FlywheelStaticRewards.sol`

Description: The function `getAccruedRewards` has the following lines:

```
uint256 balance = rewardToken.balanceOf(address(this));

if (balance < amount) {
    amount = balance;
}
```

If the rewards supplier does not add sufficient tokens to the contract, the market will silently receive less funds than expected.

Recommendation: Clarify that this is expected behavior. Ensure that systems are in-place to detect insufficient balances in reward contracts.

QSP-5 No coverage scripts

Severity: Low Risk

Status: Unresolved

Description: The project uses `forge` for project management. It is not immediately clear how well code coverage is supported, but no scripts are currently available.

Recommendation: Add scripts to run coverage.

QSP-6 Failing tests and missing test suite documentation

Severity: Low Risk

Status: Unresolved

Description: As described in the "Test Results" section below, several tests appear to be failing. Further, there is limited documentation in the test suite describing how to run tests. No coverage scripts were provided.

Recommendation: Add developer documentation for running the test suite and ensure all tests pass.

Automated Analyses

Slither

We could not run slither due to it, nor `truffle-flattener`, not supporting `forge`.

Code Documentation

1. Consider using full natspec docstrings for all function definitions, particularly in interfaces.

Adherence to Best Practices

1. It may make sense to define an interface for `Flywheel` itself, including functions such as `flywheelPreSupplierAction` and so on.
2. In `FlywheelDynamicRewards.getAccruedRewards`, the nested assignment in `rewardToken.safeTransferFrom(address(market), flywheel, amount = rewardToken.balanceOf(address(market)));` should be moved to its own line.

Test Results

Test Suite Results

Several tests are failing. It is not clear if there are missing configuration steps in the documentation.

```
$ forge test
compiling...
Compiling 18 files with 0.8.10
Compilation finished successfully
success.
Running 5 tests for FlywheelIntegrationTest.json:FlywheelIntegrationTest
[FAIL. Reason: Setup failed: Execution reverted: Revert(Reverted), (gas: 2282153)] testIntegration() (gas: 0)
[FAIL. Reason: Setup failed: Execution reverted: Revert(Reverted), (gas: 2282153)] testPreSupplier() (gas: 0)
[FAIL. Reason: Setup failed: Execution reverted: Revert(Reverted), (gas: 2282153)] testPreSupplierOld() (gas: 0)
[FAIL. Reason: Setup failed: Execution reverted: Revert(Reverted), (gas: 2282153)] testPreTransfer() (gas: 0)
[FAIL. Reason: Setup failed: Execution reverted: Revert(Reverted), (gas: 2282153)] testPreTransferOld() (gas: 0)

Running 4 tests for FlywheelStaticRewardsTest.json:FlywheelStaticRewardsTest
[PASS] testGetAccruedRewards() (gas: 108599)
[PASS] testGetAccruedRewardsAfterEnd() (gas: 89050)
[PASS] testGetAccruedRewardsCappedAfterEnd() (gas: 89071)
[PASS] testSetRewardsInfo() (gas: 29379)

Running 12 tests for FlywheelTest.json:FlywheelTest
[PASS] testAccrue() (gas: 239179)
[PASS] testAccrueBeforeAddMarket() (gas: 124535)
[PASS] testAccrueSecondUserLater() (gas: 313981)
[PASS] testAccrueTwoUsers() (gas: 287079)
[PASS] testAccrueTwoUsersBeforeAddMarket() (gas: 150083)
[PASS] testAccrueTwoUsersSeparately() (gas: 292820)
[PASS] testAddMarket() (gas: 27294)
[PASS] testBoost() (gas: 1594855)
[PASS] testClaim() (gas: 251534)
[PASS] testFailAddMarket() (gas: 4503)
[PASS] testFailSetFlywheelRewards() (gas: 4463)
[PASS] testSetFlywheelRewards() (gas: 4191)

Failed tests:
[FAIL. Reason: Setup failed: Execution reverted: Revert(Reverted), (gas: 2282153)] testIntegration() (gas: 0)
[FAIL. Reason: Setup failed: Execution reverted: Revert(Reverted), (gas: 2282153)] testPreSupplier() (gas: 0)
[FAIL. Reason: Setup failed: Execution reverted: Revert(Reverted), (gas: 2282153)] testPreSupplierOld() (gas: 0)
[FAIL. Reason: Setup failed: Execution reverted: Revert(Reverted), (gas: 2282153)] testPreTransfer() (gas: 0)
[FAIL. Reason: Setup failed: Execution reverted: Revert(Reverted), (gas: 2282153)] testPreTransferOld() (gas: 0)

Encountered a total of 5 failing tests, 16 tests succeeded


$ FORK_BLOCK=14193630 npm run test:integration

> dapptools-template@1.0.0 test:integration
> forge clean && forge test --fork-url https://eth-mainnet.alchemyapi.io/v2/$MAINNET_ALCHEMY_API_KEY --fork-block-number $FORK_BLOCK

compiling...
Compiling 18 files with 0.8.10
Compilation finished successfully
success.
Error:
   0: Deserialization Error: missing field `id` at line 1 column 295. Response: {"message":"Not Found","logref":null,"path":null,"_links":{"self":
{"href":"/v2/","templated":false,"profile":null,"deprecation":null,"title":null,"hreflang":null,"type":null,"name":null}},"_embedded":{"errors":[{"message":"Page Not Found","logref":null,"path":null,"_links":{},"_embedded":
{}}]}}
```

Code Coverage

No coverage scripts were provided.

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

```
0438316a33553278ef993209a540f271c004fd68ec484b386d1cd50612e22bcf ./src/FlywheelCore.sol
6bc2f6eb224856d0e559f1aec6b98d036914728df7f49dc7e8a5b04a7ecfc381 ./src/FuseFlywheelCore.sol
ac92d899a9aa46911031ad9d3a98b03e3725cc3f0fada389fcbe9573f76b730d ./src/rewards/FlywheelDynamicRewards.sol
b95430eaea64b0766518f89b02ced919e83071248c4dbe00d4d70e7b33f73d0c ./src/rewards/FlywheelStaticRewards.sol
8ac0cbc3c0ae8dd9c14192bac71d3072c347ab26427c13d1c1db68a89e221660 ./src/interfaces/IFlywheelBooster.sol
1ddfba6b245764ea485d76aeb8c9fbcdde64cfa5311dee364ebf6994ec8094cf ./src/interfaces/IFlywheelRewards.sol
```

Tests

```
e677ef89bfe70561d2c8ebd993f2b386776dc6134594b10b3fe114059debf75e ./test/FlywheelStaticRewardsTest.t.sol
b15b17b3893a5b1562d239954d81dc0beccfa3c85a4fcde12906258b431d8d3a ./test/FlywheelTest.sol
a46896c73d70d3e70e8dd2ca3e5b37fbc8bc8be003328322757d19b5fee694f5 ./test/Integration.t.sol
c5b41dad817f9766e98bd522a0c979d25cc0426680ee0ec46e19648e64bb251c ./test/mocks/MockBooster.sol
8d46572de786f60de7dc43f4eac32f37e69910e7411396819f4ef3f9bec992fd ./test/mocks/MockMarket.sol
```

Changelog

- 2022-02-18 - Initial report

About Quantstamp

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

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

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

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