Quantstamp Security Assessment Certificate

August 23rd 2021 — Quantstamp Verified

defiyield.tech

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

Executive Summary

Type	Automated yield engine
Reviewers	Leonardo Passos, Senior Research Engineer
Timeline	2021-08-17 through 2021-08-23
EVM	Berlin
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	None
Documentation Quality	Undetermined
Test Quality	Undetermined

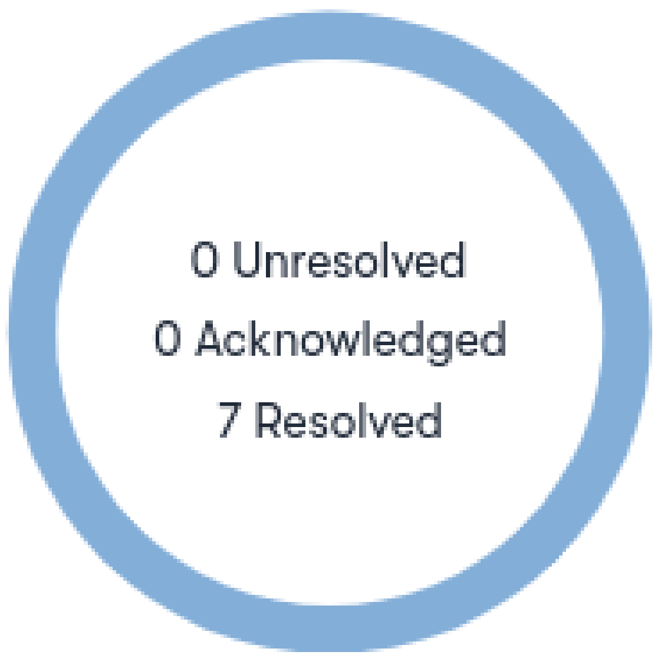
Source Code

Repository	Commit
None	None

Goals

- Can deposited funds be lost?
- Is the deposit/withdraw logic correct?

Total Issues	**7** (7 Resolved)
High Risk Issues	**2** (2 Resolved)
Medium Risk Issues	**1** (1 Resolved)
Low Risk Issues	**2** (2 Resolved)
Informational Risk Issues	**2** (2 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

Quantstamp did a security review of two contracts: `DefiAeth` and `DefiYusdc`. We did not inspect any other source code or deployed contracts that the latter two interface with. Overall, the inspected code is fairly well written, but it is unclear whether they are thoroughly tested or correctly deployed, as these are outside of the scope of this review. Altogether, we found seven issues, of which two are of high severity and deserves prompt attention from the development team.

Update: After re-auditing, all reported issues were fixed.

ID	Description	Severity	Status
QSP-1	Maximum Approval Put Funds at Risk in the Face of Hacks	High	Fixed
QSP-2	Ownership Can be Renounced	High	Fixed
QSP-3	`IERC20.approve` Return Value is Ignored	Medium	Fixed
QSP-4	Constructor Input Address Parameters Could be 0x0	Low	Fixed
QSP-5	Input Contract Addresses Could Refer to EOAs	Low	Fixed
QSP-6	Unlocked Pragma	Informational	Fixed
QSP-7	Clone-and-Own	Informational	Fixed

Quantstamp Review Breakdown

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

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

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

Methodology

The Quantstamp reviewing process follows a routine series of steps:

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

Findings

QSP-1 Maximum Approval Put Funds at Risk in the Face of Hacks

Severity: *High Risk*

Status: Fixed

File(s) affected: `DefiAeth.sol`, `DefiYusdc.sol`

Description: The `DefiAeth` constructor approves the `_aTokenPool` token contract the maximum amount possible (`type(uint256).max`). While this is certainly convenient, in the event of a hack, `_aTokenPool` could drain all the tokens owned by `DefiAeth`. This could occur if `_aTokenPool` is not trusted, if it has a bug, if it relies on privileged operations requiring specific private keys that get stolen, etc. A similar issue occurs in `DefiYusdc`.

Recommendation: At the very least introduce the ability to pause `DefiAeth` & `DefiYusdc` in case any of the contracts they interface with gets hacked. Additionally, provide external facing documentation s.t. users of your platform are aware of the risks involved.

Update: As suggested, developers made the contracts pausable. As a follow-up, we suggest approving a value of zero when a pause is made, an setting it back to `type(uint256).max` upon un-pausing. This could either be scripted, or put as part of the overriden `pause` & `unpause` functions.

Update (2): Developers implemented the `pause` & `unpause` functions as suggested previously.

QSP-2 Ownership Can be Renounced

Severity: *High Risk*

Status: Fixed

File(s) affected: `DefiAeth.sol`, `DefiYusdc.sol`

Description: All files that inherit from the `Ownable` contract could be left with no owner if the latter renounces his ownership. Consequently, one loses the ability of pausing/unpausing the target contracts and protect funds in the face of hacks.

Recommendation: Override the `renounceOwnership` such that it always reverts.

QSP-3 `IERC20.approve` Return Value is Ignored

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `DefiAeth.sol`, `DefiYusdc.sol`

Description: Calls to `IERC20(...).approve(...)` are missing a check whether the approval succeeds; missing such checks can lead to system misbehavior.

Recommendation: Both `DefiAeth` and `DefiYusdc` set `using SafeERC20 for IERC20`; hence, change calls to `approve` to `safeApprove`. The latter handles returns properly, throwing an exception in case of a failure. Also, prior to deployment, make sure that the `approve` implementation of the contracts that `DefiAeth` and `DefiYusdc` will interface with DO return a value.

QSP-4 Constructor Input Address Parameters Could be 0x0

Severity: *Low Risk*

Status: Fixed

File(s) affected: `DefiAeth.sol`, `DefiYusdc.sol`

Description: The constructor in `DefiAeth` & `DefiYusdc` does not check if the input address parameters are different from `address(0)` (a.k.a `0x0`). If 0x0 is used to set internal state, the resulting system will not behave as expected.

Recommendation: Either one of:

1. In your deployment scripts, ensure that input addresses passed on to the constructor are not 0x0, or;
2. Add verification logic to the smart contracts requiring input addresses to be different from `address(0)`.

QSP-5 Input Contract Addresses Could Refer to EOAs

Severity: *Low Risk*

Status: Fixed

File(s) affected: `DefiAeth.sol`, `DefiYusdc.sol`

Description: The constructor in `DefiAeth` & `DefiYusdc` does not check if the input addresses are indeed a contract. Hence, it is possible to pass in an Externally Owned Account (EOA) when a contract is expected. In the latter case, if EOA contract addresses are used to set internal state, the system will not behave as expected.

Recommendation: Either one of:

1. In your deployment scripts, ensure that the input addresses are indeed the expected contract addresses, or;
2. Add verification logic to the smart contracts requiring that the input addresses pass the `Address.isContract` check (mitigation).

QSP-6 Unlocked Pragma

Severity: *Informational*

Status: Fixed

File(s) affected: `DefiAeth.sol`, `DefiYusdc.sol`

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-7 Clone-and-Own

Severity: *Informational*

Status: Fixed

File(s) affected: `DefiAeth.sol`, `DefiYusdc.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. 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.

Recommendation: Contracts `Context`, `ReentrancyGuard`, and `Ownable` are clones from OpenZeppelin. Same for the `SafeERC20` and `Address` libraries. Instead of cloning and owning those, we suggest adding an explicit dependency by means of using a dependency package manager. Not only this allows to keep track of fixes, it is also easier to just pull them.

Adherence to Specification

Quantstamp has not provided with any specification and there we cannot confirm whether the code is truly aligned with the project's requirements.

Adherence to Best Practices

- **[Fixed]** In `DefiAeth` & `DefiYusdc`, having an owner is irrelevant, as no function is marked as `onlyOwner`- except for `renounceOwnership`, which is required for renouncing the ownership itself. As the owner has no role, it is safe to remove it from both contracts.

Test Results

Test Suite Results

Quantstamp has not been given access to the project's test suite and therefore cannot make comments on its quality nor coverage.

Code Coverage

Quantstamp has not been given access to the project's test suite and therefore cannot make comments on its quality nor coverage.

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

```
b878baaf19ec0e71a3e42903536a59864713ef5c26cdbe7cbc597bd28ad489e7  ./Downloads/DefiYusdc.sol
```

```
a0f6cc5ba6baf42592573cf69bf2fc35a0018f9e1fd2fb3d8b3e6f051eaedebf  ./Downloads/DefiAeth.sol
```

Changelog

- 2021-08-17 - Initial report
- 2021-08-19 - Re-audit report (1)
- 2021-08-23 - Re-audit report (2)

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.