Quantstamp Security Assessment Certificate

QUANTSTAMP VERIFIED SECURITY CERTIFICATE

December 17th 2021 — Quantstamp Verified

Rari Vaults

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	2021-12-13 through 2022-01-21
EVM	Arrow Glacier
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	None
Documentation Quality	Medium
Test Quality	Undetermined
Source Code	

Repository	Commit
vaults	0d57d65
vaults	021fb61

Total Issues	**8** (2 Resolved)
High Risk Issues	0 (0 Resolved)
Medium Risk Issues	**1** (0 Resolved)
Low Risk Issues	**6** (2 Resolved)
Informational Risk Issues	**1** (0 Resolved)
Undetermined Risk Issues	0 (0 Resolved)

0 Unresolved
6 Acknowledged
2 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 several issues of varying severity were noted. Our main concern relates to the upgrade logic that utilizes `selfdestruct`, which may not properly clean up funds in associated strategies. We were unable to compile or test the contracts with the provided documentation as noted in the Test Results section.
Update: all issues have been addressed or acknowledged as of commit 021fb61.

ID	Description	Severity	Status
QSP-1	`destroy` does not clean up strategies	^ Medium	Acknowledged
QSP-2	Unchecked and undocumented bounds on configuration parameters	⌄ Low	Mitigated
QSP-3	Missing input validation	⌄ Low	Acknowledged
QSP-4	Missing return value check	⌄ Low	Acknowledged
QSP-5	Unrestricted functions	⌄ Low	Mitigated
QSP-6	Contract Receive Should Be Restricted	⌄ Low	Acknowledged
QSP-7	Unnecessary function implementation	⌄ Low	Acknowledged
QSP-8	Privileged Roles and Ownership	○ 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 v0.8.1

Steps taken to run the tools:

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

Findings

QSP-1 `destroy` does not clean up strategies

Severity: *Medium Risk*

Status: Acknowledged

File(s) affected: `Vault.sol`

Description: The function `destroy` invokes `selfdestruct`, sending any floating ETH to the caller. However, it does not perform any cleanup on funds deposited via strategies. It is unclear how the re-initialization process will recover these funds. Using self destruct as an upgrade solution is not recommended since the storage is kept in the same contract meaning that all users liquidity token balances will not be saved.

Recommendation: Ensure that all underlying funds have been recovered before upgrading. Avoid usage of `selfdestruct` as an upgrade solution especially when it is not handled properly.

Update: From the Rari team -- This is desired behavior. Administrators calling destroy() are expected to have flushed the Vault prior to calling. This behavior is difficult to codify, especially since this function is only meant for use in unforeseen emergencies, which is why it is left up to the caller.

QSP-2 Unchecked and undocumented bounds on configuration parameters

Severity: *Low Risk*

Status: Mitigated

File(s) affected: `VaultConfigurationModule.sol`

Description: There are several setter functions (e.g., `setDefaultFeePercent`) that allow for arbitrarily large inputs (i.e., values that are beyond 100%). Further, it is unclear from the inline documentation what the precision is for each variable, i.e., what is the `uint256` value that would correspond to 100%?

Recommendation: Add inline documentation describing the intended precision of each variable. Add `require` statements to prevent erroneous values for each state variable.

Update: From the Rari team -- This logic and documentation is present in Vault.sol, it is not the configuration module's responsibility to manage bounds. Using a value out of bounds will only result in Vaults refusing to accept the change. However, for the purpose of clarity we have added comments for each variable and function instructing readers to seek out explanations in Vault.sol when appropriate in a PR.

QSP-3 Missing input validation

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `VaultInitializationModule.sol, VaultFactory.sol, Vault.sol`

Description: The following functions should perform additional argument validation:

1. `VaultInitializationModule.constructor` should check that all address arguments are non-zero.
2. `VaultInitializationModule.setConfigModule` should check that `newConfigModule` is non-zero.
3. `VaultFactory.constructor` should check that all address arguments are non-zero.
4. `Vault.constructor` should check that `_UNDERLYING` is non-zero.

Recommendation: Add corresponding `require` statements to each function.

Update: From the Rari team -- All of these arguments are validated indirectly (like addresses being called in the constructor) or invalid values do not pose a risk to the system as they can be reset, etc.

QSP-4 Missing return value check

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `Vault.sol`

Description: In `depositIntoStrategy`, we have the two lines:

```
// Deposit into the strategy and assume it will revert on error.
ETHStrategy(address(strategy)).mint{value: underlyingAmount}();
```

It is not clear why the return-value is not checked unlike in the `else` branch.

Recommendation: Wrap the statement in a `require`.

Update: From the Rari team -- ETH strategies (CEther tokens) do not return error codes. This is a choice made in the Compound system, not ours.

QSP-5 Unrestricted functions

Severity: *Low Risk*

Status: Mitigated

File(s) affected: `VaultConfigurationModule.sol`

Description: Anyone can call `VaultConfigurationModule.syncFeePercent()`, `VaultConfigurationModule.syncHarvestDelay()`, `VaultConfigurationModule.syncHarvestWindow()` and `VaultConfigurationModule.syncTargetFloatPercent()` which can change the Vault contract state. Similarly `VaultInitalizationModule.VaultInitializationModule()` is not restricted.

Recommendation: It should be clearly documented if this action is part of the design and the logic behind it.

Update: From the Rari team -- They are intentionally public, so administrators do not have to call the function themselves for every Vault after a configuration value is updated. However, at the time of the audit anyone could call these functions as many times as they wanted even if the configuration values did not change. To mitigate this spam vector we have disabled calling the function if the new configuration value would not change the state of the Vault in a PR.

QSP-6 Contract Receive Should Be Restricted

Severity: Low Risk

Status: Acknowledged

File(s) affected: `VaultRouterModule.sol`

Description: `VaultRouterModule.receive()` should be allowed only to the WETH contract address to avoid receiving ETH funds by mistake.

Recommendation: Restrict access to the function.

Update: From the Rari team -- This change would not be worth the additional bytecode size and complexity it introduces.

QSP-7 Unnecessary function implementation

Severity: Low Risk

Status: Acknowledged

File(s) affected: `Vault.sol`

Description: The implemented `Vault.underlyingIsWETH()` seems unnecessary and should be removed since the function allows authorized user to change an important property that is only related with the underlying token itself.

Recommendation: Remove the function.

Update: From the Rari team -- It is present to allow supporting multiple WETH implementations on multiple different networks.

QSP-8 Privileged Roles and Ownership

Severity: Informational

Status: Acknowledged

File(s) affected: `Vault.sol`

Description: Smart contracts will often have `owner` variables to designate the person with special privileges to make modifications to the smart contract. The authenticated addresses for `Vault.sol` can arbitrarily change or seize strategies, and `selfdestruct` the 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.

Update: From the Rari team -- These contracts will be managed by on-chain governance and we will ensure we inform users of all the relevant details.

Automated Analyses

Slither

We were unable to run tools due to issues related to Dapp Tools.

Adherence to Best Practices

1. It is not clear why the `withdrawalQueue` is called a queue since it is used primarily as a stack. **Update:** fixed.
2. Remove all unnecessary `uncheck{}` blocks even if you think that arithmetic operation will never over or underflow. **Update:** from the Rari team -- We consider the gas and bytecode size benefit worth the additional risk.

Test Results

Test Suite Results

We could not run tests due to the following errors during build.

```
$ make
nix-env -f https://github.com/dapphub/dapptools/archive/master.tar.gz -iA solc-static-versions.solc_0_8_10
replacing old 'solc-static-0.8.10'
installing 'solc-static-0.8.10'
npm install
audited 15 packages in 0.4s
found 1 moderate severity vulnerability
  run `npm audit fix` to fix them, or `npm audit` for details
dapp update
make: dapp: No such file or directory
make: *** [update] Error 1
```

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

```
08edd8d76a7463b3ab7d38c305d67e9c74b60bf8fe8bf3e867ed893a634bb173 ./src/Vault.sol

93d1f310d1f35e545683f70153a720401ee3cc2b9106695492a32263ac7e178b ./src/VaultFactory.sol

9e36428b3e4597e9cbf9b94307092688ada41567537d8aef5a4fb6b79d592ffb ./src/test/Integration.t.sol

8a636fa79f5b692d0ef859a45916c7b1b8c5650febe7b820223ae17c96f158bf ./src/test/Vault.t.sol

8544af957c269150443683626bb753abf8be711a98d500c0a018a35ec821dc1a ./src/test/VaultAuthorityModule.t.sol

55c2304505c9886da27d717ef8c780dbd092a5550bbefac8e0dc242caac2c166 ./src/test/VaultConfigurationModule.t.sol

16d360c00552970eb4dab9242573e74aca40dffad99db7577e1bb2412c717335 ./src/test/VaultFactory.t.sol

4a4a66860ebb0fc8db4371eb0df8b4857ceabf0c5aafb2d308ce2a909d82bd8c ./src/test/VaultInitializationModule.t.sol

d060cc006957a417dff549c627ea81fe4560e45c49da39db8b8a5c5d2fc642e2 ./src/test/VaultRouterModule.t.sol

606d189426a447da60c53b74987e54a08c31ff99bfb0513fa67c2275a9c647cb ./src/test/mocks/MockERC20Strategy.sol

500764b3d57e1e41c3644cc969fa0538cc907a642780b4fd1c9c902c16301b53 ./src/test/mocks/MockETHStrategy.sol

853dc73f15211f7849f708c3d3761c16152090b95c64a3dd24726b83eb8a56d0 ./src/modules/VaultAuthorityModule.sol

934886762b60adeaa10ace3f6bd508a6aa9fa950e30baa32731d9b2a8fcc551e ./src/modules/VaultConfigurationModule.sol

9b43b9b20faecfc634d90bd061c422c5be09967d3a542ec154776487d69f10af ./src/modules/VaultInitializationModule.sol

e1fba234951c1a487b5a2ebdd4857dbb4af38a50d2b67ca0ca80f98181b3466b ./src/modules/VaultRouterModule.sol

49947b1403b57327942cff97bdef82288fb83f2763f0ee31f8271a60a512236c ./src/interfaces/AllowedPermit.sol

af94754c36f97df4c84535ceb9ddf545a76cebb71d1a3d16676923a656929e63 ./src/interfaces/Strategy.sol
```

Tests

```
9e36428b3e4597e9cbf9b94307092688ada41567537d8aef5a4fb6b79d592ffb ./test/Integration.t.sol

8a636fa79f5b692d0ef859a45916c7b1b8c5650febe7b820223ae17c96f158bf ./test/Vault.t.sol

8544af957c269150443683626bb753abf8be711a98d500c0a018a35ec821dc1a ./test/VaultAuthorityModule.t.sol

55c2304505c9886da27d717ef8c780dbd092a5550bbefac8e0dc242caac2c166 ./test/VaultConfigurationModule.t.sol

16d360c00552970eb4dab9242573e74aca40dffad99db7577e1bb2412c717335 ./test/VaultFactory.t.sol

4a4a66860ebb0fc8db4371eb0df8b4857ceabf0c5aafb2d308ce2a909d82bd8c ./test/VaultInitializationModule.t.sol

d060cc006957a417dff549c627ea81fe4560e45c49da39db8b8a5c5d2fc642e2 ./test/VaultRouterModule.t.sol

606d189426a447da60c53b74987e54a08c31ff99bfb0513fa67c2275a9c647cb ./test/mocks/MockERC20Strategy.sol

500764b3d57e1e41c3644cc969fa0538cc907a642780b4fd1c9c902c16301b53 ./test/mocks/MockETHStrategy.sol
```

Changelog

- 2021-12-21 - Initial report
- 2022-01-20 - Revised report based on commit 021fb61

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.