Quantstamp Security Assessment Certificate

October 29th 2020 — Quantstamp Verified

Aavegotchi GHST Staking

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

Executive Summary

Type	Diamond pattern showcase
Auditors	Kacper Bąk, Senior Research Engineer Jan Gorzny, Blockchain Researcher Fayçal Lalidji, Security Auditor
Timeline	2020-10-09 through 2020-10-19
EVM	Muir Glacier
Languages	Solidity, Javascript
Methods	Unit Testing, Manual Review
Specification	EIP-2535 Diamond Standard
Documentation Quality	Medium
Test Quality	Low

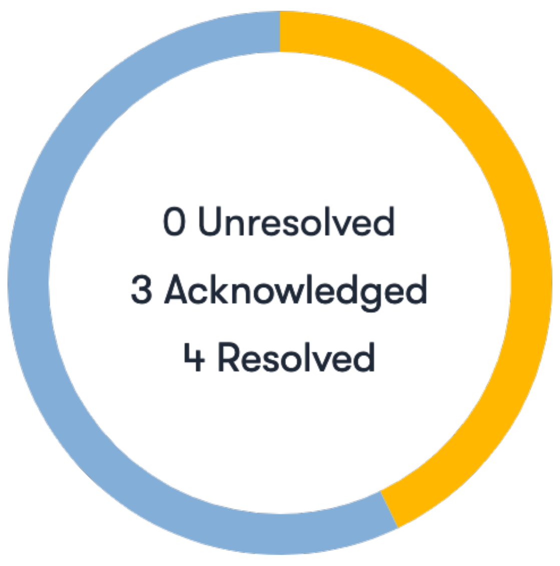
Source Code

Repository	Commit
ghst-staking	af267c0

Goals

- Can staking funds get locked up in the contract?
- Are there any security issues with the diamond pattern?

Total Issues	7 (4 Resolved)
High Risk Issues	1 (1 Resolved)
Medium Risk Issues	0 (0 Resolved)
Low Risk Issues	3 (2 Resolved)
Informational Risk Issues	3 (1 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

Overall the project is overengineered since it uses the diamond pattern although it is unnecessary. The team, however, decided to use the diamond pattern intentionally to show how it could be used. Although we have not found any significant issues with the code itself, we very highly recommend improving the test suite both for: 1) the user-facing functionality of the project, and 2) the diamond pattern itself. We also provide a few recommendations for improving the code.
Update: the report has been revised based on commit `5978a3d`. Notably, the team improved the test suite.

ID	Description	Severity	Status
QSP-1	Limited test suite	High	Mitigated
QSP-2	The interface `IERC20` not fully compatible with ERC20	Low	Fixed
QSP-3	Lack of non-zero check for address	Low	Fixed
QSP-4	Diamond facet upgrade	Low	Acknowledged
QSP-5	Unlocked Pragma	Informational	Fixed
QSP-6	Clone-and-Own	Informational	Acknowledged
QSP-7	Storage Data Packing	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 v0.6.6
- 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 .s`
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 Limited test suite

Severity: *High Risk*

Status: Mitigated

Description: The project appears to have a very limited test suite. Tests may express requirements and are necessary to validate software's intent.
Update: the team informed us that this project uses the diamond-2 implementation. The repository has 13 tests for that diamond implementation. Furthermore, the team has improved the test suite as of commit `5978a3d`.

Recommendation: We highly recommend improving the test suite, especially since this is one of the first projects to rely on the recently proposed diamond pattern.

QSP-2 The interface `IERC20` not fully compatible with ERC20

Severity: *Low Risk*

Status: Fixed

File(s) affected: `IERC20.sol`

Description: The interface `IERC20` is not fully compatible with ERC20 as the name would suggest. For example, the interface is missing `approve()`.

Recommendation: We recommend one of the following: 1) making the interface fully compatible with the ERC20 standard; 2) renaming the interface to reflect that it is only partially compatible with ERC20; or 3) documenting this fact in the code.

QSP-3 Lack of non-zero check for address

Severity: *Low Risk*

Status: Fixed

File(s) affected: `GHSTStakingDiamond.sol`

Description: The function `constructor()` does not check if arguments of type `address` are non-zero which may lead to invalid initialization.

Recommendation: We recommend adding a relevant check or documenting the fact that a 0x0 address is a correct value.

QSP-4 Diamond facet upgrade

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `GHSTStakingDiamond.sol`

Description: If during an upgrade `diamondCut()` calls are executed in multiple Ethereum transactions, users may be exposed to contracts that are upgraded only partially, i.e., some of the functions are upgraded while others are not. This may result in unexpected inconsistencies.

Recommendation: We recommend upgrading the contracts in a single transaction, or making the fallback function pausable for the duration of an upgrade.

QSP-5 Unlocked Pragma

Severity: *Informational*

Status: Fixed

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-6 Clone-and-Own

Severity: *Informational*

Status: Acknowledged

File(s) affected: `String.sol`, `SafeMath.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.

Recommendation: 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.

QSP-7 Storage Data Packing

Severity: *Informational*

Status: Acknowledged

File(s) affected: `DiamondCutFacet.sol`, `DiamondLoupeFacet.sol`, `LibDiamond.sol`

Description: All data packing for `LibDiamond.DiamondStorage` is done manually whereas structures could be used instead. The compiler will tightly pack any ordered state variables by groups of 32 bytes and use only one storage slot (also applicable for struct members). Please note that all the necessary operations for reading and writing variables will be created and optimized automatically by the compiler.

Recommendation: When there is tradeoff between lower gas consumption and code simplicity, we recommend picking the latter unless there are good reasons not to. Higher complexity tends to introduce more risks and less clarity.
The diamond pattern functions in `DiamondCutFacet`, `DiamondLoupeFacet` and `LibDiamond` could be simplified. For example, `mapping(bytes4 => bytes32) facets` maps to `address facet` and `uint96 selectorPosition`, it can be redefined as follows:

```
struct Facet {
    address facetAddr;
    uint96 selectorPosition;
}
mapping(bytes4 => Facet) facets;
```

The same logic applies to all the other state variables including `DiamondStorage.selectorSlots` where a `bytes4` array can be used to fetch the selector following its `selectorPosition`.
Update: the team informed us that they used a gas-optimized implementation on purpose; it is less readable than it could be and that it is also more gas efficient. The diamond-1 implementation is implemented the same way as diamond-2 but uses a more readable style that costs a little more gas.

Adherence to Best Practices

1. In `GHSTStakingDiamond.sol#101`, `either` -> `ether`. **Update:** fixed.
2. `LibERC20.sol` may be used with addresses other than tokens. There may exist some corner cases where `handleReturn()` reverts for other reasons than `transfer()` and `transferFrom()`. We recommend adjusting the error messages accordingly. **Update:** fixed.
3. The function `StakingFacet.claimTickets()` takes an array of IDs as an argument. The length of the array is proportional to the number of IDs. It could be simplified by setting the length to 6. Each index would represent an ID. A number placed at a given index in the array would determine the number of tickets with given ID. **Update:** fixed.
4. In `StakingFacet.sol` double check that the upper limits on the number of tokens won't cause overflow in L23 and downcasts from `uint256` are correct, e.g., in lines 30, 36, 42, 55. **Update:** fixed.

Test Results

Test Suite Results

We ran the test suite following the steps:

1. `npm install`
2. `touch .secret`
3. `npx buidler test`

```
GHSTStakingDiamond
  ✓ Check that all functions and facets exist in the diamond (442ms)
  ✓ Check that diamond upgrades are not possible
  ✓ Should stake all GHST (96ms)
  ✓ Should stake GHST-ETH pool tokens (53ms)
  ✓ Should accumulate frens from staked GHST and staked GHST-ETH pool tokens
  ✓ Should be able to claim ticket (58ms)
  ✓ Cannot claim tickets above 5
  ✓ Should not be able to purchase 3 million Godlike tickets
  ✓ Total supply of all tickets should be 27
  ✓ Balance of second item should be 6
  ✓ Balance of third item should be 5
  ✓ Can transfer own ticket
  ✓ Cannot transfer someone else's ticket
  ✓ Can batch transfer own tickets
  ✓ Cannot batch transfer someone else's tickets
  ✓ Can get balance of batch
  ✓ Cannot transfer more tickets than one owns
  ✓ Can approve transfers
  ✓ Can transfer approved transfers (84ms)
  ✓ Can withdraw staked GHST (80ms)
  ✓ Can withdraw staked GHST-ETH pool tokens (81ms)
  ✓ Cannot withdraw more than staked
  ✓ Can set setBaseURI (91ms)
  ✓ Set contract owner (51ms)


24 passing (3s)
```

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

```
fa52b2470f817c244759cdf4f5dfba0319c1ee9c5e9e937a70480c46ca8204f4 ./contracts/GHSTStakingDiamond.sol
8286ad4b1dc5f1307b35fa571eff831969cfd26ea5be81cca560ae00a100524d ./contracts/interfaces/IERC20.sol
a280edee4411b1afe680857c2949d73b4d4db999d516b605c01f200ae4b6c2fb ./contracts/interfaces/IERC1155.sol
7feaf8d0322e0b38f901248d35eab2db866854badb281fcf5524135a96f01f43 ./contracts/interfaces/IDiamondCut.sol
82564b3a1e10a572c4776a734bbec7a045052828d4452196f4202a8903f1c472 ./contracts/interfaces/IERC1155TokenReceiver.sol
78057a2a0a0079c9d08faf027237ba67f118ebd2cd2344bad2d251e30983652d ./contracts/interfaces/IDiamondLoupe.sol
0051a3c6865b53c2baeb0eea386bd00b222b8a714c71b09a3d8c24aea4295604 ./contracts/interfaces/IERC1155Metadata_URI.sol
33eb126062d9350a2edc416e24b788db6694fb9cc9c56f25bc425eb3ab1cd396 ./contracts/interfaces/IUniswapV2Pair.sol
c50094b0805bda6c004634e1fbaf09b5bd53ccef0cd8140bfcfde81eec1c7396 ./contracts/interfaces/IERC165.sol
394e0b4dbe80b81c6ccce95f39f04766c84097f436f117da8ae6ec1f89b70dc3 ./contracts/interfaces/IGHSTStakingDiamond.sol
18d9f30afaeffa11393654cb5dcf9e5265ee18042d8a76e3f634c9e74393cf8a ./contracts/interfaces/IERC173.sol
6cb5c302023724bf4351b021d18d57343943c496504bc91b15ad0774d1c2c032 ./contracts/libraries/LibStrings.sol
8500cb67e4761722e3930fd51cd5dc400b814b5ca701e5b589661dea8e06918b ./contracts/libraries/LibERC20.sol
b37d1a7c0aee02c93f9e406ae9d85abc20487ff964f7045366d749022d2d5e48 ./contracts/libraries/LibDiamond.sol
d143cffcfb7404bc7668999bd86d2cfd9202b27c67a364fa170e42bc4544dd69 ./contracts/libraries/AppStorage.sol
ef42d1b6d5212fd8afbdddb64abccddc5bd5c931436018563b86c1c3e65a000f ./contracts/test/GHST/ERC20.sol
f6401fb608bd5d2bad2f05c551786750c88e90f0beed0e96a0e60c2e42eb985f ./contracts/test/GHST/AppStorage.sol
04f7d812bc8fa128f593eebe7db454a2c526e396446c23ad2b483e867fe0bb78 ./contracts/facets/DiamondCutFacet.sol
2ccca5af6d396873575ad01754083e973475093aa941db4a8ff490da998247c8 ./contracts/facets/TicketsFacet.sol
e28a83775c8d1a6535ef46a45024efba180d1821eaf29b355748c037fabd3231 ./contracts/facets/OwnershipFacet.sol
478a834a235f8a72c22fe0c3fa20d58a2ef8fb7308abd8118cb989217f7f4251 ./contracts/facets/StakingFacet.sol
4f8dd8c65c44e9daa588854cc9d6973d61681de42172443d2bbfa5a8709cfa1b ./contracts/facets/DiamondLoupeFacet.sol
```

Tests

```
d8a9c3060ad6e7711fd99802f0585d8c12a3362a38778e196bb28f41ddd81b27 ./test/test.js
```

Changelog

- 2020-10-15 - Initial report
- 2020-10-23 - Updated report based on commit `5978a3d`

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.