

# Executive Summary

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

Type	Staking	Documentation quality	Medium	<div><div></div></div>
Timeline	2023-09-11 through 2023-09-13	Test quality	Low	<div><div></div></div>
Language	Solidity	Total Findings	6	<div><div></div></div> 6 Fixed: 5 Acknowledged: 1
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review	High severity findings ⓘ	0	
Specification	None	Medium severity findings ⓘ	3	<div><div></div></div> 3 Fixed: 2 Acknowledged: 1
Source Code	<ul style="list-style-type: none"><li><a href="#">stakewithus/eth-staking-contracts</a> ⓘ</li><li><a href="#">#d960b34</a> ⓘ</li></ul>	Low severity findings ⓘ	1	<div><div></div></div> 1 Fixed: 1
Auditors	<ul style="list-style-type: none"><li>Julio Aguiar Auditing Engineer</li><li>Michael Boyle Auditing Engineer</li><li>Mustafa Hasan Senior Auditing Engineer</li></ul>	Undetermined severity findings ⓘ	0	
		Informational findings ⓘ	2	<div><div></div></div> 2 Fixed: 2

# Summary of Findings

The project enables users to stake their ETH and earn rewards from block production, gas tips, and MEV boosts. This is carried out by allowing users to deposit ETH in multiples of 32 plus a fee for deploying validator nodes for the user. A fee recipient contract also gets deployed on the user's first deposit and acts as their reward store from deployed validators.

The code is generally well written and there is a fair amount if inline comments in the Solidity contract files. The provided test suite is comprehensive but lacked good code coverage for multiple contracts. The team spent sometime on improving that and coverage is now above 80% for most contracts over most metrics.

Our assessment did not yield any high severity issues, however a number of medium, low, and informative severity issues were identified and fixed by the team by implementing the recommendations provided by the Quantstamp team.

As for documentation, there is good technical documentation for the project, which provides a good amount of information on how the protocol operates when coupled with the inline code comments. However, user-facing documentation does not seem to exist and we recommend availing such information to end users.

ID	DESCRIPTION	SEVERITY	STATUS
STA-1	Missing Input Validation	• Medium ⓘ	Fixed
STA-2	Privileged Roles and Ownership	• Medium ⓘ	Acknowledged
STA-3	Re-Entrancy Attack Could Drain the <code>FeeRecipient</code> Contract	• Medium ⓘ	Fixed
STA-4	<code>_refund()</code> Does Not Check if <code>_validators</code> Is Zero	• Low ⓘ	Fixed
STA-5	Unlocked Pragma	• Informational ⓘ	Fixed
STA-6	State Changing Functions Do Not Emit Events	• Informational ⓘ	Fixed

# Assessment Breakdown

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



### Disclaimer

Only features that are contained within the repositories at the commit hashes specified on the front page of the report are within the scope of the audit and fix review. All features added in future revisions of the code are excluded from consideration in this report.

### 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

1. Code review that includes the following
  1. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
  2. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
  3. 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:
  1. 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.
  2. 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, clarity, 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.

# Scope

All contracts in the project repository were in-scope for this audit except the `SafeTransferLib.sol` contract.

### Files Included

`/src/*`

### Files Excluded

`lib/SafeTransferLib.sol`

# Findings

## STA-1 Missing Input Validation

• Medium ⓘ

Fixed



### Update

Marked as "Fixed" by the client. Addressed in: `0d3e7e515f4a6efb929344048a99b5ef7c849d43` . The client provided the following explanation:

`owned.nominateOwner()` no zero check is intentional, equivalent to revoking a nomination

Additionally, no maximum was defined for use in `Staking.setOneTimeFee()` .

File(s) affected: `Owned.sol` , `Staking.sol`

**Description:** It is important to validate inputs, even if they only come from trusted addresses, to avoid human error. Following is a list of functions and inputs can could use validation before usage:

1. `Staking.constructor()` should revert if `depositContract_` is the zero address.
2. `Staking.setOneTimeFee()` should revert if `oneTimeFee_` is the same as `oneTimeFee` and does not require the one-time fee to be less than a set maximum.
3. `Staking.setPerformanceFee()` should revert if `performanceFee_` is the same as `performanceFee` and does not require a maximum of less than one hundred percent.
4. `Staking.setTreasury()` should revert if `treasury_` is the zero address or `treasury_` is the same as `treasury` .
5. `Staking.deposit()` should revert if `user_` is the zero address.
6. `Staking.setRefundDelay()` should revert if `refundDelay_` is the same as `refundDelay` .
7. `Owned.nominateOwner()` should revert if `nominatedOwner_` is the zero address.
8. `Owned.constructor()` should revert if `owner` or `operator_` are the zero address.
9. `Owned.setOperator()` should revert if `operator_` is the zero address.

**Recommendation:** We recommend adding the relevant checks.

## STA-2 Privileged Roles and Ownership

• Medium ⓘ Acknowledged

### Update

Marked as "Acknowledged" by the client. The client provided the following explanation:

user-facing documentation will be outside of github repo

**File(s) affected:** `Staking.sol` , `FeeRecipient.sol`

**Description:** The contracts in this project contain privileged functions that only permitted users can call. These functions control vital aspects of the protocol and abuse of privilege can lead to unexpected behavior and even loss of funds for users. For instance, the owner could set the performance fee to one hundred percent and claim all of the staking rewards for the treasury address that they set.

The following roles have escalated privileges and can perform the following operations:

1. `owner`
  1. Can set the performance fee from zero to one hundred percent.
  2. Can set the one-time fee without any maximum value.
  3. Can set the treasury address to any address.
  4. Can set the refund delay to any length of time greater than or equal to seven days.
2. `operator`
  1. Can call `stake()` to complete the staking process by sending Ether to the deposit contract.
  2. Can refund pending staking deposits earlier than the user can refund themselves.
  3. Can pause and unpause the ability to make deposits. Deposits must be paused and pending validators completed before changing the one-time fee.
3. `treasury`
  1. The treasury address will receive a percentage of staking rewards from all `FeeRecipient` instances as well as the one-time fee for staking.
4. `user`
  1. Can claim their portion of the rewards from their `FeeRecipient` instance.

**Recommendation:** Make users aware of this via documentation.

## STA-3 Re-Entrancy Attack Could Drain the `FeeRecipient` Contract

• Medium ⓘ Fixed

### Update

Marked as "Fixed" by the client. Addressed in: `a374662acdb17b26c6eb6dc62a14c7d911563958` .

**File(s) affected:** `FeeRecipient.sol`

**Description:** The `_treasureClaim()` function transfers `ETH` to the treasury and then updates the `_userRewards` which does not conform with the Check-Effects-Interactions pattern and allows for the possibility of calling `treasureClaim()` multiple times until the contract is drain. The possibility of this happening is very low since we assume the treasure of the project to be a trusted actor behind a multisig. However, in case that multisig gets compromised, this issue would be very severe.

**Recommendation:** We recommend to follow the CEI pattern by updating `_userRewards` before transferring `ETH` to the treasury.

✓

Update

Marked as "Fixed" by the client. Addressed in: `ca2080a53423be7ec17bc29be13ab61cc9243a53` .

**File(s) affected:** `Staking.sol` ,

**Description:** The `_refund()` function proceeds with attempting to transfer the value `validators_ * (_DEPOSIT_AMOUNT + oneTimeFee)` to the `user_` and emit an event even if `_validators` is zero. This leads to unnecessary gas consumption as there is nothing to refund.

**Recommendation:** Revert if `_validators` is zero.

STA-5 Unlocked Pragma

• Informational ⓘ

Fixed

✓

Update

Marked as "Fixed" by the client. Addressed in: `bb99765e7a5367c2919f93813ceb697a4731beeb` .

**File(s) affected:** `Staking.sol` , `StakingConstants.sol` , `FeeRecipient.sol` , `lib/*` , `interfaces/*`

**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.

STA-6 State Changing Functions Do Not Emit Events

• Informational ⓘ

Fixed

✓

Update

Marked as "Fixed" by the client. Addressed in: `01a2446e160a5b7565607ebc651a79586dee17a2` .

**File(s) affected:** `FeeReceipient.sol`

**Description:** Functions `claimRewards()` and `treasuryClaim()` change the contract state and do not emit events. This may be undesirable in case off-chain logic depends on emitted events.

**Recommendation:** Consider emitting events with the old and new states of the contract.

# Definitions

- **High severity** – High-severity issues usually put a large number of users' sensitive information at risk, or are reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
- **Medium severity** – Medium-severity issues tend to put a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or are reasonably likely to lead to moderate financial impact.
- **Low severity** – 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.
- **Fixed** – Adjusted program implementation, requirements or constraints to eliminate the risk.
- **Mitigated** – Implemented actions to minimize the impact or likelihood of the risk.
- **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).

# Code Documentation

1. Typo in NatSpec comment for `receive()` : change "if" to "is".

## 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

- `ad9...3a6 ./src/Staking.sol`
- `f00...ab8 ./src/FeeRecipient.sol`
- `3d8...ed4 ./src/StakingConstants.sol`
- `6e8...791 ./src/interfaces/IDepositContract.sol`
- `201...4e9 ./src/interfaces/ISTaking.sol`
- `c10...5ab ./src/lib/Pausable.sol`
- `7ff...6cd ./src/lib/Owned.sol`
- `47b...7e3 ./src/lib/SafeTransferLib.sol`

### Tests

- `6ba...fdd ./test/unit/Staking.t.sol`
- `38d...5ba ./test/unit/FeeRecipient.t.sol`
- `b36...e17 ./test/integration/Staking.integration.t.sol`

## Toolset

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

### Setup

Tool Setup:

- [Slither](#)  v0.8.3

Steps taken to run the tools:

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

## Automated Analysis

### Slither

Slither returned several items, but most were false positives. All the relevant ones were included in this report.

## Test Suite Results

```
Running 2 tests for test/unit/Pausable.t.sol:PausableTest
[PASS] test_whenNotPaused() (gas: 35661)
[PASS] test_whenPaused() (gas: 25408)
Test result: ok. 2 passed; 0 failed; 0 skipped; finished in 321.03µs

Running 3 tests for test/unit/Owned.t.sol:OwnedTest
[PASS] test_nominateOwner() (gas: 38404)
[PASS] test_onlyOperator() (gas: 20096)
[PASS] test_setOperator() (gas: 33821)
Test result: ok. 3 passed; 0 failed; 0 skipped; finished in 609.24µs
```



```
Running 12 tests for test/unit/Staking.t.sol:StakingTest
[PASS] test_deposit(uint8) (runs: 256, μ: 519871, ~: 519871)
[PASS] test_deposit_reverts_if_zero_address() (gas: 22493)
[PASS] test_deposit_reverts_on_invalid_amount(uint256) (runs: 256, μ: 30131, ~: 30131)
[PASS] test_deposit_reverts_when_paused() (gas: 27221)
[PASS] test_receive(uint8) (runs: 256, μ: 471470, ~: 471470)
[PASS] test_refund(uint8,uint256) (runs: 256, μ: 449842, ~: 450105)
[PASS] test_refund_reverts_before_refund_delay() (gas: 469941)
[PASS] test_refund_reverts_if_no_pending_validators() (gas: 479692)
[PASS] test_setOneTimeFee_reverts_if_there_are_pending_validators() (gas: 453289)
[PASS] test_setPerformanceFee_cannot_exceed_maximum(uint256) (runs: 256, μ: 13399, ~: 13399)
[PASS] test_setRefundDelay_cannot_exceed_maximum(uint256) (runs: 256, μ: 13464, ~: 13464)
[PASS] test_stake_reverts_if_invalid_length() (gas: 476195)
Test result: ok. 12 passed; 0 failed; 0 skipped; finished in 37.23ms

Running 3 tests for test/unit/FeeRecipient.t.sol:FeeRecipientTest
[PASS] test_claimRewards(uint256) (runs: 256, μ: 93026, ~: 99779)
[PASS] test_claimRewards_reverts_if_nothing_to_claim() (gas: 20416)
[PASS] test_treasuryClaim(uint256) (runs: 256, μ: 99140, ~: 105944)
Test result: ok. 3 passed; 0 failed; 0 skipped; finished in 37.24ms

Running 1 test for test/integration/Staking.integration.t.sol:StakingIntegrationTest
[PASS] test_stake() (gas: 83630)
Test result: ok. 1 passed; 0 failed; 0 skipped; finished in 1.24s

Ran 5 test suites: 21 tests passed, 0 failed, 0 skipped (21 total tests)
```

# Code Coverage

Coverage for `FeeRecipient.sol` is 100% and for `Staking.sol` is above 80% for lines, statements, and functions, and above 70% for branches. Coverage has been improved for `Owned.sol` and `Pausable.sol` and hits 100% on all criterea.

File	% Lines	% Statements	% Branches	% Funcs
<b>script</b> /Staking.s.sol	0.00% <b>(0/6)</b>	0.00% <b>(0/8)</b>	0.00% <b>(0/4)</b>	0.00% <b>(0/2)</b>
<b>src</b> /FeeRecipient.sol	100.00% <b>(15/15)</b>	100.00% <b>(25/25)</b>	100.00% <b>(6/6)</b>	100.00% <b>(5/5)</b>
<b>src</b> /Staking.sol	85.71% <b>(48/56)</b>	83.33% <b>(70/84)</b>	71.88% <b>(23/32)</b>	81.25% <b>(13/16)</b>
<b>src/lib</b> /Owned.sol	100.00% <b>(8/8)</b>	100.00% <b>(10/10)</b>	100.00% <b>(2/2)</b>	100.00% <b>(3/3)</b>
<b>src/lib</b> /Pausable.sol	100.00% <b>(4/4)</b>	100.00% <b>(4/4)</b>	100.00% <b>(0/0)</b>	100.00% <b>(2/2)</b>
<b>src/lib</b> /SafeTransferLib.sol	100.00% <b>(3/3)</b>	100.00% <b>(3/3)</b>	50.00% <b>(1/2)</b>	100.00% <b>(1/1)</b>
<b>test/unit</b> /Owned.t.sol	100.00% <b>(1/1)</b>	100.00% <b>(1/1)</b>	100.00% <b>(0/0)</b>	100.00% <b>(1/1)</b>
<b>test/unit</b> /Pausable.t.sol	100.00% <b>(2/2)</b>	100.00% <b>(2/2)</b>	100.00% <b>(0/0)</b>	100.00% <b>(2/2)</b>
<b>test/unit</b> /Staking.t.sol	100.00% <b>(0/0)</b>	100.00% <b>(0/0)</b>	100.00% <b>(0/0)</b>	0.00% <b>(0/1)</b>
Total	85.26% <b>(81/95)</b>	83.94% <b>(115/137)</b>	69.57% <b>(32/46)</b>	81.82% <b>(27/33)</b>

# Changelog

- 2023-09-19 - Initial report

# About Quantstamp

Quantstamp is a global leader in blockchain security. Founded in 2017, Quantstamp's mission is to securely onboard the next billion users to Web3 through its best-in-class Web3 security products and services.

Quantstamp's team consists of cybersecurity experts hailing from globally recognized organizations including Microsoft, AWS, BMW, Meta, and the Ethereum Foundation. Quantstamp engineers hold PhDs or advanced computer science degrees, with decades of combined experience in formal verification, static analysis, blockchain audits, penetration testing, and original leading-edge research.

To date, Quantstamp has performed more than 500 audits and secured over \$200 billion in digital asset risk from hackers. Quantstamp has worked with a diverse range of customers, including startups, category leaders and financial institutions. Brands that Quantstamp has worked with include Ethereum 2.0, Binance, Visa, PayPal, Polygon, Avalanche, Curve, Solana, Compound, Lido, MakerDAO, Arbitrum, OpenSea and the World Economic Forum.

Quantstamp's collaborations and partnerships showcase our commitment to world-class research, development and security. We're honored to work with some of the top names in the industry and proud to secure the future of web3.

Notable Collaborations & Customers:

- Blockchains: Ethereum 2.0, Near, Flow, Avalanche, Solana, Cardano, Binance Smart Chain, Hedera Hashgraph, Tezos
- DeFi: Curve, Compound, Maker, Lido, Polygon, Arbitrum, SushiSwap
- NFT: OpenSea, Parallel, Dapper Labs, Decentraland, Sandbox, Axie Infinity, Illuvium, NBA Top Shot, Zora
- Academic institutions: National University of Singapore, MIT

## 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 or other making available of the report to you by Quantstamp.

## 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. 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 any website and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any output generated by such software.

## Disclaimer

The review and this report are provided on an as-is, where-is, and as-available basis. To the fullest extent permitted by law, Quantstamp disclaims 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. You agree that your access and/or use of the report and other results of the review, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. 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. This report is based on the scope of materials and documentation provided for a limited review at the time provided. You acknowledge that Blockchain technology remains under development and is subject to unknown risks and flaws and, as such, the report may not be complete or inclusive of all vulnerabilities. The review is limited to the materials identified in the report and does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. The report does not indicate the endorsement by Quantstamp of any particular project or team, nor guarantee its security, and may not be represented as such. No third party is entitled to rely on the report in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. Quantstamp does not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party, or 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, or any related services and products, any hyperlinked websites, or any other websites or mobile applications, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third party. 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.



# Quantstamp