Quantstamp Security Assessment Certificate

December 23rd 2021 — Quantstamp Verified

AStarNetwork: Staking

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

Executive Summary

Type	Staking
Auditors	Souhail Mssassi, Research Engineer Cristiano Silva, Research Engineer
Timeline	2021-11-12 through 2021-11-19
EVM Languages	Rust
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	README.md
Documentation Quality	High
Test Quality	High

Source Code

Repository	Commit
Astar	6d8c38a

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

We have reviewed the code, documentation, and test suite and found several issues of various severities. Overall, we consider the code to be well-written and with sufficient documentation and a good test suite . We have outlined suggestions to better follow best practices, and recommend addressing all the findings to tighten the contracts for future deployments or contract updates.

ID	Description	Severity	Status
QSP-1	Potential segfault in `localtime_r` invocations	^ Medium	Acknowledged
QSP-2	Mathematical Operations That Lead To Overflow	^ Medium	Fixed
QSP-3	Missing Validation In Some Variables	⌄ Low	Fixed
QSP-4	Order of Validation in Maximum Number of Stakers	○ Informational	Fixed

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:

- Tarpaulin 0.14.2
- Rust Audit v0.12.0
- Cargo Geiger 0.11.1

Steps taken to run the tools:

`cargo install cargo-tarpaulin` (on a Linux environment) ` cargo tarpaulin --verbose -o=Html --output-dir='..' `cargo install cargo-audit` `cargo audit` cargo install cargo-geiger

Findings

QSP-1 Potential segfault in `localtime_r` invocations

Severity: *Medium Risk*

Status: Acknowledged

File(s) affected: `Cargo.lock`

Related Issue(s): SWC-RUSTSEC-2020-0159

Description: In the Cargo.lock file the package `chrono` with the version `0.4.19` is used, in fact this version is affected by a data race between `localtime_r`and `setenv`. You can refer more to more details of the vulnerability in the following link : https://rustsec.org/advisories/RUSTSEC-2020-0159

Recommendation: The team should update the chrono version to the lasted version.

QSP-2 Mathematical Operations That Lead To Overflow

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `/dapps-staking/src/pallet/mod.rs`

Description: In the `on_initialize` function (file `/dapps-staking/src/pallet/mod.rs`, L244), the `next_era` variable is incremented using the `+` operator, which can cause an overflow. If one uses the debug build, it can cause a panic; however, in release build it will overflow silently. Same issue in lines 393,408,413,549,552,593,683

Recommendation:

- Instead of using the `+` operator, we recommend using the `checked_add` function to prevent an overflow.
- Instead of using the `x` operator, we recommend using the `checked_mul` function to prevent an overflow.

Update: The team has fixed all the issues except for the era index overflow and number of stakers overflow since these scenarios aren't feasible.

QSP-3 Missing Validation In Some Variables

Severity: *Low Risk*

Status: Fixed

File(s) affected: `dapps/src/pallet/mod.rs`

Description:

- In `dapps/src/pallet/mod.rs` (`L327`): the `unregister` function executes a loop N times such that N is the length of the stakers. The problem here is that there is no limit on the length of this variable, which can cause a denial of service during the execution of this function call. Same issue in line 578
- In `dapps/src/pallet/mod.rs` (`L390`): the `bond_and_stake` function verify if the value `value_to_stake` is equal to Zero value if it's the case the transaction revert, this verification is not sufficient, we should enforce it by verifying if the `value_to_stake` is greater than 0 .

Recommendation: * Enforce a limitation on the number of the `stakers`.

- Verify if `value_to_stake` is greater than the zero value.

Update:
* The number of stakers is limited by a configurable constant.

- The team did the verification using `value_to_stake > Zero::zero()`

QSP-4 Order of Validation in Maximum Number of Stakers

Severity: *Informational*

Status: Fixed

Description: In the `bond_and_stake` function (L400), after updating the ledger value and getting the latest era staking point the number of `stakers` is verified if it is less than the maximum number of stakers, if it's not the case the transaction will revert . This verification should be done before updating the ledger value.

Recommendation: The team should verify if we exceed the `MaxNumberOfStakersPerContract` before setting any variables or doing calculations.

Automated Analyses

Tarpaulin

Nov 18 17:16:25.095 INFO cargo_tarpaulin::process_handling::linux: Launching test Nov 18 17:16:25.095 INFO cargo_tarpaulin::process_handling: running /root/dapps-staking/target/debug/deps/pallet_dapps_staking-b462790de7eee140

running 38 tests test mock::__construct_runtime_integrity_test::runtime_integrity_tests ... ok test tests::bond_and_stake_different_value_is_ok ... ok test tests::bond_and_stake_different_eras_is_ok ... ok test tests::bond_and_stake_insufficient_value ... ok test tests::bond_and_stake_on_unregistered_contract_not_works ... ok test tests::bond_and_stake_history_depth_has_passed_is_ok ... ok test tests::bond_and_stake_too_many_stakers_per_contract ... ok test tests::bond_and_stake_two_different_contracts_is_ok ... ok test tests::bond_and_stake_two_stakers_one_contract_is_ok ... ok test tests::claim_contract_not_registered ... ok test tests::claim_after_unregister_is_ok ... ok test tests::claim_invalid_eras ... ok test tests::claim_for_all_valid_history_eras_is_ok ... ok test tests::claim_is_ok ... ok test

tests::claim_one_contract_one_staker ... ok test tests::claim_one_contract_two_stakers ... ok test tests::claim_twice_in_same_era ... ok test tests::new_era_forcing ... ok test tests::claim_two_contracts_three_stakers_new ... ok test tests::new_era_is_ok ... ok test tests::on_initialize_when_dapp_staking_enabled_in_mid_of_an_era_is_ok ... ok test tests::on_initialize_is_ok ... ok test tests::on_unbalanced_is_ok ... ok test tests::register_is_ok ... ok test tests::register_same_contract_twice_not_works ... ok test tests::register_twice_with_same_account_not_works ... ok test tests::register_with_pre_approve_enabled ... ok test tests::unbond_unstake_and_withdraw_contract_is_not_ok ... ok test tests::staking_info_is_ok ... ok test tests::unbond_unstake_and_withdraw_in_different_eras ... ok test tests::unbond_unstake_and_withdraw_multiple_time_is_ok ... ok test tests::unbond_unstake_and_withdraw_history_depth_has_passed_is_ok ... ok test tests::unbond_unstake_and_withdraw_value_below_staking_threshold ... ok test tests::unbond_unstake_and_withdraw_unstake_not_possible ... ok test tests::unregister_after_register_is_ok ... ok test tests::unregister_stake_and_unstake_is_not_ok ... ok test tests::unregister_with_incorrect_contract_does_not_work ... ok test tests::unregister_with_staked_contracts_is_ok ... ok

test result: ok. 38 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 1.42s

Nov 18 17:16:31.389 INFO cargo_tarpaulin::report: Coverage Results: || Tested/Total Lines: || src/lib.rs: 0/2 || src/mock.rs: 39/46 || src/pallet/mod.rs: 218/235 || src/testing_utils.rs: 74/75 || src/tests.rs: 831/831 || src/weights.rs: 16/68 || 93.72% coverage, 1178/1257 lines covered

Rust Audit

Fetching advisory database from `https://github.com/RustSec/advisory-db.git` Loaded 374 security advisories (from /root/.cargo/advisory-db) Updating crates.io index Updating crates.io index Updating git repository `https://github.com/paritytech/substrate.git` Scanning Cargo.lock for vulnerabilities (263 crate dependencies) Crate: chrono Version: 0.4.19 Title: Potential segfault in `localtime_r` invocations Date: 2020-11-10 ID: RUSTSEC-2020-0159 URL: https://rustsec.org/advisories/RUSTSEC-2020-0159 Solution: No safe upgrade is available! Dependency tree: chrono 0.4.19 😟😟😟 tracing-subscriber 0.2.25 😟😟😟 sp-tracing 4.0.0-dev 😟😟😟 sp-runtime-interface 4.0.0-dev 😟 😟😟😟 sp-io 4.0.0-dev 😟 😟 😟😟😟 sp-runtime 4.0.0-dev 😟 😟 😟 😟😟😟 sp-version 4.0.0-dev 😟 😟 😟 😟 😟😟😟 sp-api 4.0.0-dev 😟 😟 😟 😟 😟 😟😟😟 sp-timestamp 4.0.0-dev 😟 😟 😟 😟 😟 😟 😟😟😟 pallet-timestamp 4.0.0-dev 😟 😟 😟 😟 😟 😟 😟😟😟 pallet-session 4.0.0-dev 😟 😟 😟 😟 😟 😟 😟 😟😟😟 pallet-dapps-staking 1.1.0 😟 😟 😟 😟 😟 😟 😟😟😟 pallet-dapps-staking 1.1.0 😟 😟 😟 😟 😟 😟😟😟 sp-session 4.0.0-dev 😟 😟 😟 😟 😟 😟 😟😟😟 pallet-session 4.0.0-dev 😟 😟 😟 😟 😟 😟😟😟 frame-benchmarking 4.0.0-dev 😟 😟 😟 😟 😟 😟😟😟 pallet-timestamp 4.0.0-dev 😟 😟 😟 😟 😟 😟😟😟 pallet-dapps-staking 1.1.0 😟 😟 😟 😟 😟 😟😟😟 pallet-balances 4.0.0-dev 😟 😟 😟 😟 😟 😟😟😟 pallet-dapps-staking 1.1.0 😟 😟 😟 😟 😟😟😟 frame-system 4.0.0-dev 😟 😟 😟 😟 😟😟😟 pallet-timestamp 4.0.0-dev 😟 😟 😟 😟 😟😟😟 pallet-session 4.0.0-dev 😟 😟 😟 😟 😟😟😟 pallet-dapps-staking 1.1.0 😟 😟 😟 😟 😟😟😟 pallet-balances 4.0.0-dev 😟 😟 😟 😟 😟😟😟 frame-benchmarking 4.0.0-dev 😟 😟 😟 😟😟😟 sp-timestamp 4.0.0-dev 😟 😟 😟 😟😟😟 sp-staking 4.0.0-dev 😟 😟 😟 😟 😟😟😟 sp-session 4.0.0-dev 😟 😟 😟 😟 😟😟😟 pallet-session 4.0.0-dev 😟 😟 😟 😟 😟😟😟 pallet-dapps-staking 1.1.0 😟 😟 😟 😟 😟😟😟 frame-support 4.0.0-dev 😟 😟 😟 😟 😟😟😟 pallet-timestamp 4.0.0-dev 😟 😟 😟 😟 😟😟😟 pallet-session 4.0.0-dev 😟 😟 😟 😟 😟😟😟 pallet-dapps-staking 1.1.0 😟 😟 😟 😟 😟😟😟 pallet-balances 4.0.0-dev 😟 😟 😟 😟 😟😟😟 frame-system 4.0.0-dev 😟 😟 😟 😟 😟😟😟 frame-benchmarking 4.0.0-dev 😟 😟 😟 😟😟😟 sp-session 4.0.0-dev 😟 😟 😟 😟😟😟 sp-inherents 4.0.0-dev 😟 😟 😟 😟 😟😟😟 sp-timestamp 4.0.0-dev 😟 😟 😟 😟 😟😟😟 pallet-timestamp 4.0.0-dev 😟 😟 😟 😟 😟😟😟 frame-support 4.0.0-dev 😟 😟 😟 😟😟😟 sp-api 4.0.0-dev 😟 😟 😟 😟😟😟 pallet-timestamp 4.0.0-dev 😟 😟 😟 😟😟😟 pallet-session 4.0.0-dev 😟 😟 😟 😟😟😟 pallet-dapps-staking 1.1.0 😟 😟 😟 😟😟😟 pallet-balances 4.0.0-dev 😟 😟 😟 😟😟😟 frame-system 4.0.0-dev 😟 😟 😟 😟😟😟 frame-support 4.0.0-dev 😟 😟 😟 😟😟😟 frame-benchmarking 4.0.0-dev 😟 😟 😟😟😟 sp-application-crypto 4.0.0-dev 😟 😟 😟 😟😟😟 sp-runtime 4.0.0-dev 😟 😟 😟😟😟 pallet-session 4.0.0-dev 😟 😟 😟😟😟 pallet-dapps-staking 1.1.0 😟 😟 😟😟😟 frame-system 4.0.0-dev 😟 😟 😟😟😟 frame-support 4.0.0-dev 😟 😟 😟😟😟 frame-benchmarking 4.0.0-dev 😟 😟😟😟 sp-core 4.0.0-dev 😟 😟 😟😟😟 sp-trie 4.0.0-dev 😟 😟 😟 😟😟😟 sp-state-machine 0.10.0-dev 😟 😟 😟 😟 😟😟😟 sp-io 4.0.0-dev 😟 😟 😟 😟 😟😟😟 sp-api 4.0.0-dev 😟 😟 😟 😟 😟😟😟 frame-support 4.0.0-dev 😟 😟 😟 😟😟😟 sp-io 4.0.0-dev 😟 😟 😟 😟😟😟 pallet-session 4.0.0-dev 😟 😟 😟😟😟 sp-state-machine 0.10.0-dev 😟 😟 😟😟😟 sp-session 4.0.0-dev 😟 😟 😟😟😟 sp-runtime 4.0.0-dev 😟 😟 😟😟😟 sp-keystore 0.10.0-dev 😟 😟 😟 😟😟😟 sp-io 4.0.0-dev 😟 😟 😟😟😟 sp-io 4.0.0-dev 😟 😟 😟😟😟 sp-inherents 4.0.0-dev 😟 😟 😟😟😟 sp-application-crypto 4.0.0-dev 😟 😟 😟😟😟 sp-api 4.0.0-dev 😟 😟 😟😟😟 pallet-session 4.0.0-dev 😟 😟 😟😟😟 pallet-dapps-staking 1.1.0 😟 😟 😟😟😟 frame-system 4.0.0-dev 😟 😟 😟😟😟 frame-support 4.0.0-dev 😟 😟😟😟 frame-benchmarking 4.0.0-dev 😟😟😟 sp-io 4.0.0-dev 😟😟😟 frame-support 4.0.0-dev
error: 1 vulnerability found!

Cargo Geiger

Symbols: 😟 = No `unsafe` usage found, declares #![forbid(unsafe_code)] 😟 = No `unsafe` usage found, missing #![forbid(unsafe_code)] 😟😟 = `unsafe` usage found

Functions Expressions Impls Traits Methods Dependency

0/0 0/0 0/0 0/0 0/0 😟 pallet-dapps-staking 1.1.0 0/0 4/10 0/0 0/0 0/0 😟😟 😟😟😟 num-traits 0.2.14 0/0 10/10 0/0 0/0 0/0 😟😟 😟 😟😟😟 libm 0.2.1 0/0 4/4 0/0 0/0 0/0 😟😟 😟😟😟 parity-scale-codec 2.3.1 2/2 350/350 2/2 0/0 7/7 😟😟 😟 😟😟😟 arrayvec 0.7.2 0/0 4/4 0/0 0/0 0/0 😟😟 😟 😟 😟😟😟 serde 1.0.130 0/0 0/0 0/0 0/0 0/0 😟 😟 😟 😟😟😟 serde_derive 1.0.130 0/0 0/0 0/0 0/0 0/0 😟 😟 😟 😟😟😟 proc-macr o2 1.0.32 0/0 0/0 0/0 0/0 0/0 😟 😟 😟 😟 😟😟😟 unico de-xid 0.2.2 0/0 0/0 0/0 0/0 0/0 😟 😟 😟 😟😟😟 quote 1.0 .10 0/0 0/0 0/0 0/0 0/0 😟 😟 😟 😟 😟😟😟 proc- macro2 1.0.32 0/0 45/45 3/3 0/0 2/2 😟😟 😟 😟 😟😟😟 syn 1.0.8 1 0/0 0/0 0/0 0/0 0/0 😟 😟 😟 😟😟😟 proc- macro2 1.0.32 0/0 0/0 0/0 0/0 0/0 😟 😟 😟 😟😟😟 quote 1.0.10 0/0 0/0 0/0 0/0 0/0 😟 😟 😟 😟😟😟 unico de-xid 0.2.2 15/15 1105/1108 14/14 1/1 62/62 😟😟 😟 😟😟😟 bitvec 0.20.4 0/0 0/0 0/0 0/0 0/0 😟 😟 😟 😟😟😟 funty 1.1.0 0/0 0/0 0/0 0/0 0/0 😟 😟 😟 😟😟😟 radium 0.6.2 0/0 4/4 0/0 0/0 0/0 😟😟 😟 😟 😟😟😟 serde 1.0.130 0/0 0/0 0/0 0/0 0/0 😟 😟 😟 😟😟😟 tap 1.0.1 0/0 0/0 0/0 0/0 0/0 😟 😟 😟 😟😟😟 wyz 0.2.0 0/0 0/0 2/2 3/3 0/0 😟😟 😟 😟😟😟 byte-slice-cast 1.2.0 1/1 295/295 20/20 8/8 5/5 😟😟 😟 😟😟😟 generic-array 0.14.4 0/0 4/4 0/0 0/0 0/0 😟😟 😟 😟 😟😟😟 serde 1.0.130 0/0 0/0 0/0 0/0 0/0 😟 😟 😟 😟😟😟 typenum 1.14.0 0/0 0/0 0/0 0/0 0/0 😟 😟 😟😟😟 impl-trait-for-tuples 0.2.1 0/0 0/0 0/0 0/0 0/0 😟 😟 😟 😟😟😟 proc-macro2 1.0.3 2 0/0 0/0 0/0 0/0 0/0 😟 😟 😟 😟😟😟 quote 1.0.10 0/0 45/45 3/3 0/0 2/2 😟😟 😟 😟 😟😟😟 syn 1.0.81 0/0 0/0 0/0 0/0 0/0 😟 😟 😟😟😟 parity-scale-codec-de rive 2.3.1 0/0 0/0 0/0 0/0 0/0 😟 😟 😟 😟😟😟 proc-macro-crate 1.1.0 0/0 0/0 0/0 0/0 0/0 😟 😟 😟 😟 😟😟😟 thiserror 1.0 .30 0/0 0/0 0/0 0/0 0/0 😟 😟 😟 😟 😟 😟😟😟 thiserror -impl 1.0.30 0/0 0/0 0/0 0/0 0/0 😟 😟 😟 😟 😟 😟😟😟 proc- macro2 1.0.32 0/0 0/0 0/0 0/0 0/0 😟 😟 😟 😟 😟 😟😟😟 quote 1.0.10 0/0 45/45 3/3 0/0 2/2 😟😟 😟 😟 😟 😟 😟😟😟 syn 1 .0.81 0/0 0/0 0/0 0/0 0/0 😟 😟 😟 😟 😟😟😟 toml 0.5.8 0/0 4/4 0/0 0/0 0/0 😟😟 😟 😟 😟 😟😟😟 serde 1.0 .130 0/0 0/0 0/0 0/0 0/0 😟 😟 😟 😟😟😟 proc-macro2 1.0.3 2 0/0 0/0 0/0 0/0 0/0 😟 😟 😟 😟😟😟 quote 1.0.10 0/0 45/45 3/3 0/0 2/2 😟😟 😟 😟 😟😟😟 syn 1.0.81 0/0 4/4 0/0 0/0 0/0 😟😟 😟 😟😟😟 serde 1.0.130 0/0 0/0 0/0 0/0 0/0 😟 😟😟😟 scale-info 1.0.0 15/15 1105/1108 14/14 1/1 62/62 😟😟 😟 😟😟😟 bitvec 0.20.4 0/0 0/0 0/0 0/0 0/0 😟 😟 😟😟😟 cfg-if 1.0.0 0/0 0/0 0/0 0/0 0/0 😟 😟 😟😟😟 derive_more 0.99.16 0/0 0/0 0/0 0/0 0/0 😟 😟 😟 😟😟😟 convert_case 0.4. 0 0/0 15/15 0/0 0/0 0/0 😟😟 😟 😟 😟 😟😟😟 rand 0.7.3 2/4 50/150 1/1 0/0 3/3 😟😟 😟 😟 😟 😟😟😟 getrandom 0.1.16 0/0 0/0 0/0 0/0 0/0 😟 😟 😟 😟 😟 😟😟😟 cfg-i f 1.0.0 0/20 12/327 0/2 0/0 2/30 😟😟 😟 😟 😟 😟 😟😟😟 libc 0.2.107 1/1 16/16 1/1 0/0 0/0 😟😟 😟 😟 😟 😟 😟😟😟 log 0 .4.14 0/0 0/0 0/0 0/0 0/0 😟 😟 😟 😟 😟 😟😟😟 c fg-if 1.0.0 0/0 4/4 0/0 0/0 0/0 😟😟 😟 😟 😟 😟 😟😟😟 s erde 1.0.130 0/20 12/327 0/2 0/0 2/30 😟😟 😟 😟 😟 😟😟😟 libc 0.2. 107 1/1 16/16 1/1 0/0 0/0 😟😟 😟 😟 😟 😟😟😟 log 0.4.1 4 0/0 0/0 0/0 0/0 0/0 😟 😟 😟 😟 😟😟😟 rand_chac ha 0.2.2 2/2 636/712 0/0 0/0 17/25 😟😟 😟 😟 😟 😟 😟😟😟 ppv-l ite86 0.2.15 0/0 22/22 0/0 0/0 0/0 😟😟 😟 😟 😟 😟 😟😟😟 rand_ core 0.5.1 2/4 50/150 1/1 0/0 3/3 😟😟 😟 😟 😟 😟 😟😟😟 g etrandom 0.1.16 0/0 4/4 0/0 0/0 0/0 😟😟 😟 😟 😟 😟 😟😟😟 s erde 1.0.130 0/0 22/22 0/0 0/0 0/0 😟😟 😟 😟 😟 😟😟😟 rand_core 0.5.1 0/0 0/0 0/0 0/0 0/0 😟 😟 😟 😟 😟😟😟 rand_pcg 0.2.1 0/0 22/22 0/0 0/0 0/0 😟😟 😟 😟 😟 😟😟😟 rand_ core 0.5.1 0/0 4/4 0/0 0/0 0/0 😟😟 😟 😟 😟 😟😟😟 serde 1.0.130 0/0 0/0 0/0 0/0 0/0 😟 😟 😟 😟😟😟 proc-macro2 1.0.3 2 0/0 0/0 0/0 0/0 0/0 😟 😟 😟 😟😟😟 quote 1.0.10 0/0 45/45 3/3 0/0 2/2 😟😟 😟 😟 😟😟😟 syn 1.0.81 0/0 4/4 0/0 0/0 0/0 😟😟 😟 😟😟😟 parity-scale-codec 2. 3.1 0/0 0/0 0/0 0/0 0/0 😟 😟 😟😟😟 scale-info-derive 1.0 .0 0/0 0/0 0/0 0/0 0/0 😟 😟 😟 😟😟😟 proc-macro-crate 1.1.0 0/0 0/0 0/0 0/0 0/0 😟 😟 😟 😟😟😟 proc-macro2 1.0.3 2 0/0 0/0 0/0 0/0 0/0 😟 😟 😟 😟😟😟 quote 1.0.10 0/0 45/45 3/3 0/0 2/2 😟😟 😟 😟 😟😟😟 syn 1.0.81 0/0 4/4 0/0 0/0 0/0 😟😟 😟 😟😟😟 serde 1.0.130 0/0 4/4 0/0 0/0 0/0 😟😟 😟😟😟 serde 1.0.130

23/45 2568/3068 43/45 12/12 98/134

error: Found 11 warnings

Test Results

Test Suite Results

```
running 38 tests
test mock::__construct_runtime_integrity_test::runtime_integrity_tests ... ok
test tests::bond_and_stake_different_value_is_ok ... ok
test tests::bond_and_stake_different_eras_is_ok ... ok
test tests::bond_and_stake_insufficient_value ... ok
test tests::bond_and_stake_on_unregistered_contract_not_works ... ok
test tests::bond_and_stake_history_depth_has_passed_is_ok ... ok
test tests::bond_and_stake_too_many_stakers_per_contract ... ok
test tests::bond_and_stake_two_different_contracts_is_ok ... ok
test tests::bond_and_stake_two_stakers_one_contract_is_ok ... ok
test tests::claim_contract_not_registered ... ok
test tests::claim_after_unregister_is_ok ... ok
test tests::claim_invalid_eras ... ok
test tests::claim_for_all_valid_history_eras_is_ok ... ok
test tests::claim_is_ok ... ok
test tests::claim_one_contract_one_staker ... ok
test tests::claim_one_contract_two_stakers ... ok
test tests::claim_twice_in_same_era ... ok
test tests::new_era_forcing ... ok
test tests::claim_two_contracts_three_stakers_new ... ok
test tests::new_era_is_ok ... ok
test tests::on_initialize_when_dapp_staking_enabled_in_mid_of_an_era_is_ok ... ok
test tests::on_initialize_is_ok ... ok
test tests::on_unbalanced_is_ok ... ok
test tests::register_is_ok ... ok
test tests::register_same_contract_twice_not_works ... ok
test tests::register_twice_with_same_account_not_works ... ok
test tests::register_with_pre_approve_enabled ... ok
test tests::unbond_unstake_and_withdraw_contract_is_not_ok ... ok
test tests::staking_info_is_ok ... ok
test tests::unbond_unstake_and_withdraw_in_different_eras ... ok
test tests::unbond_unstake_and_withdraw_multiple_time_is_ok ... ok
test tests::unbond_unstake_and_withdraw_history_depth_has_passed_is_ok ... ok
test tests::unbond_unstake_and_withdraw_value_below_staking_threshold ... ok
test tests::unbond_unstake_and_withdraw_unstake_not_possible ... ok
test tests::unregister_after_register_is_ok ... ok
test tests::unregister_stake_and_unstake_is_not_ok ... ok
test tests::unregister_with_incorrect_contract_does_not_work ... ok
test tests::unregister_with_staked_contracts_is_ok ... ok

test result: ok. 38 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 1.42s
```

Code Coverage

```
Nov 18 17:16:31.389 INFO cargo_tarpaulin::report: Coverage Results: || Tested/Total Lines: || src/lib.rs: 0/2 || src/mock.rs: 39/46 || src/pallet/mod.rs: 218/235 ||
src/testing_utils.rs: 74/75 || src/tests.rs: 831/831 || src/weights.rs: 16/68 || 93.72% coverage, 1178/1257 lines covered
```

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

```
3e1a3b3243b5398cc8b2412911a8e857e0786e953003a1f35a71cb4abdf039e8 ./program/benchmarking.rs
d46b728e87313cf252f27cc2e61cdf0d22b91e83d7b9c9e2b4486eb62375a4d6 ./program/lib.rs
88f1330702c23c8906daf64ce4b5d0c72ea16255726d3d85b31ab8977b28965e ./program/traits.rs
6132d8fb9c183706e0ebdc9921828c2ff33dffb66d0e0e80aff54ffecd15b8bb ./program/weights.rs
```

Tests

```
340dc23996ab0c50474d03c7ece40b6c093cef7ab4bf1d33b13676bb0307709a ./tests/testing_utils.rs
016db9b66922beef6c676471ac8135a06a0d4ab38a11b526f2167015b5ca646c ./tests/tests.rs
```

Changelog

- 2021-11-19 - 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.