Quantstamp Security Assessment Certificate

December 8th 2021 — Quantstamp Verified

Quarry

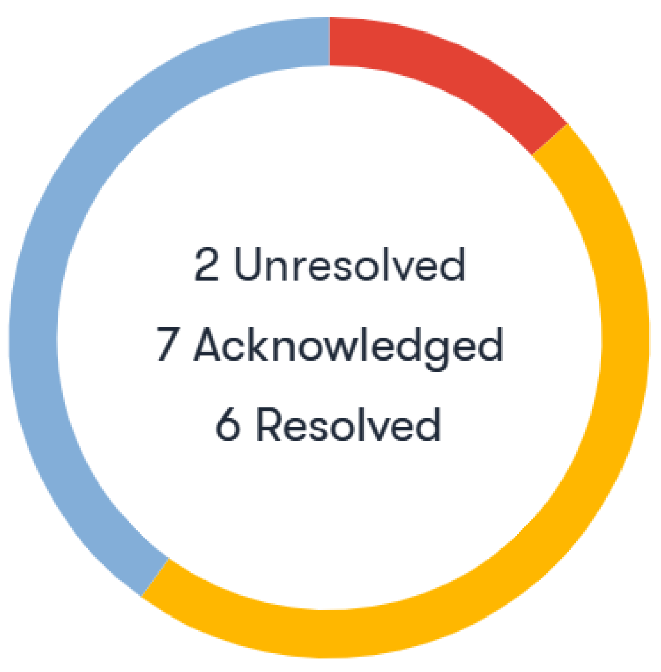
This security assessment was prepared by Quantstamp, the leader in blockchain security

Executive Summary

Type	Yield Farm Service on Solana
Auditors	Poming Lee, Research Engineer Jake Goh Si Yuan, Senior Security Researcher Mohsen Ahmadvand, Senior Research Engineer
Timeline	2021-10-19 through 2021-12-08
Languages	Rust
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	README.md
Documentation Quality	Low
Test Quality	Undetermined
Source Code	

Repository	Commit
quarry-private (only for certain programs inside it)	5635f5b
quarry	805a7c4

Total Issues	**15** (6 Resolved)
High Risk Issues	0 (0 Resolved)
Medium Risk Issues	0 (0 Resolved)
Low Risk Issues	**4** (2 Resolved)
Informational Risk Issues	**9** (3 Resolved)
Undetermined Risk Issues	**2** (1 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, we found 15 potential issues of various levels of severity: 4 low-severity issues, 2 undetermined-severity issues, as well as 9 informational-severity issues. We also made 5 best practices recommendations.
The project provided no specifications, and the information contained in the README files provided are limited. The lack of documentation concerning the system specifications and the underlying framework means that there could be many other issues that were not discovered. Lacking documentation also impairs any independent assessment in understanding the underlying logic and checking whether the code adheres to what the system should actually be doing, a prerequisite for gaining user trust and widespread adoption.
The coverage report cannot be obtained at this moment due to tests that interact with the Solana blockchain. It is highly recommended to solve this issue whenever it is possible and to have code coverage to at least 80%, in order to avoid functional bugs that are not necessarily security issues.
Disclaimer:

1. Readers should be aware that Quantstamp was requested and had audited `quarry-miner`, `quarry-mint-wrapper`, `quarry-merge-mine` programs, not the whole system.
2. This project utilized Solana blockchain, SPL programs, Anchor framework, vipers library, and many more existing infrastructures. All the dependencies and external infrastructures are not part of this audit.

2021-12-08 update: during this reaudit, the admin team has brought some of the status of findings either into fixed or acknowledged. For the others, the admin team decided not to fix them because in their opinion they are not necessary.

ID	Description	Severity	Status
QSP-1	Misuse of variable `_mint_bump`	Low	Fixed
QSP-2	Missing rewards whenever `set_rewards_share` is called	Low	Acknowledged
QSP-3	Retroactive changes to mined amount with `set_annual_rewards`	Low	Acknowledged
QSP-4	Use of unreferenced magic number may cause future bricking of claim operation	Low	Fixed
QSP-5	MintWrapper can `transfer_admin` to same admin	Informational	Fixed
QSP-6	`current_ts` should be strictly greater than `last_checkpoint_ts` in the payroll logic	Informational	Unresolved
QSP-7	`NewRewarder`'s `Validate` should also verify `claim_fee_token_account` owner and mint	Informational	Fixed
QSP-8	Possible truncation and precision loss in the annual rewards rate calculation	Informational	Fixed
QSP-9	Privileged roles and ownership	Informational	Acknowledged
QSP-10	Quarry inherits Anchor vulnerabilities	Informational	Unresolved
QSP-11	`NewRewarder.authority` should be the signer	Informational	Acknowledged
QSP-12	`MergeMiner.owner` is not checked	Informational	Acknowledged
QSP-13	Missing check to match `quarry.token_mint_key` and `miner_vault.mint`	? Undetermined	Fixed
QSP-14	Graceful account closure	Informational	Acknowledged
QSP-15	`InitMergeMiner.validate` does not explicitly call validate on the pool account	? Undetermined	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):

- Missing account-ownership check
- Missing account-signature check
- Missing account-type check
- Missing account-program-id check
- 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
- 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.
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:

- Rust Audit v0.15.0
- Rust-Clippy Latest

Steps taken to run the tools:

Rust Audit:
1. `cargo install cargo-audit`
2. `cargo audit`

Rust-Clippy:
1. `rustup component add clippy`
2. `cargo clippy`

Findings

QSP-1 Misuse of variable `_mint_bump`

Severity: Low Risk

Status: Fixed

File(s) affected: `programs\quarry-merge-mine\src\lib.rs`

Description: On `L136` of `programs\quarry-merge-mine\src\lib.rs` the `bump` uses `mint_bump`, but in `L34` the name of the input parameter is `_mint_bump` instead.

Recommendation: Please make sure if this is intended or it is indeed erroneous code.

QSP-2 Missing rewards whenever `set_rewards_share` is called

Severity: Low Risk

Status: Acknowledged

File(s) affected: `programs\quarry-mine\src\lib.rs`

Description: The `set_rewards_share()` function sets the `last_update_ts` of the quarry to the time of execution. Since `last_update_ts` is a critical factor in the calculation of rewards accrued, a change of `last_update_ts` to the existing time would mean that potential rewards accrued are disregarded and erased entirely. In detail, when `set_rewards_share()` is called, `quarry.last_update_ts` is set to `now` and it is done without performing `update_rewards_internal`. The amount of `rewards_per_token_stored` that should be accumulated in the `now - quarry.last_update_ts` period is ignored and not added.

Recommendation: Run an update to the quarry before moving on the change. For instance, perform the full `update_rewards_internal()` operation in function `set_rewards_share()` instead of only updating `quarry.last_update_ts`.

Update: The admin team will call `syncQuarryRewards` client side on all the quarries immediately after making changes to quarry rewards shares and/or annual reward rate client side. Can refer to linkToCode for more details.

QSP-3 Retroactive changes to mined amount with `set_annual_rewards`

Severity: Low Risk

Status: Acknowledged

File(s) affected: `programs/quarry-mine/src/lib.rs`

Description: All calculations deriving rewards use `annual_rewards_rate` along with time staked to find out the mined amount. It is currently possible for `Rewarder.authority` to change the `annual_rewards_rate` through `set_annual_rewards()`, without calling any of the respective quarry(s)' update function. This leads to a retroactive change to the mined amount, depending on the change to `annual_rewards_rate`

Recommendation: Consider documenting this effect, or trigger an update to all quarry(s) before affecting this change.

Update: The admin team will call `syncQuarryRewards` client side on all the quarries immediately after making changes to quarry rewards shares and/or annual reward rate client side. Can refer to linkToCode for more details.

QSP-4 Use of unreferenced magic number may cause future bricking of claim operation

Severity: Low Risk

Status: Fixed

File(s) affected: `programs/quarry-mine/src/rewarder.rs`

Description: There is a validation of `require!(max_claim_fee_kbps < 10_000 * 1_000, InvalidMaxClaimFee);` on `L63`. This assumes that `max_claim_fee_kbps`, which is typically set only once when the rewarder is created, is always less than the magical `10_000 * 1_000`. This may not be the case, and future updates may inadvertently set it larger or equal to `10_000 * 1_000` and therefore halt the function of the rewarder.

Recommendation: Reference the constant (i.e., `DEFAULT_CLAIM_FEE_KBPS`) set for `max_claim_fee_kbps` and `10_000 * 1_000` together.

QSP-5 MintWrapper can `transfer_admin` to same admin

Severity: Informational

Status: Fixed

File(s) affected: `programs/quarry-mint-wrapper/src/lib.rs`

Description: It is possible for the mint wrapper admin to execute `transfer_admin()`, which would theoretically set up the next admin to `accept_admin()` afterwards. Both functions emit events at the end of successful execution. Currently, it is possible for the admin to `transfer_admin()` to itself, which would emit nonsensical events thereafter.

Recommendation: Validate that the pending admin is not the current admin when `transfer_admin()`.

QSP-6 `current_ts` should be strictly greater than `last_checkpoint_ts` in the payroll logic

Severity: Informational

Status: Unresolved

File(s) affected: `programs/quarry-mine/src/payroll.rs:L91, programs/quarry-mine/src/payroll.rs:L131`

Description: For

```
require!(current_ts >= self.last_checkpoint_ts, InvalidTimestamp);
```

and later in the reward calculation is becomes apparent that having `current_ts=last_checkpoint_ts` yields `compute_time_worked=0` rendering the operation useless:

```
/// Gets the latest time rewards were being distributed.
pub fn last_time_reward_applicable(&self, current_ts: i64) -> i64 {
    cmp::min(current_ts, self.famine_ts)
}

/// Calculates the amount of seconds the [Payroll] should have applied rewards for.
fn compute_time_worked(&self, current_ts: i64) -> Option<i64> {
    Some(cmp::max(
        0,
        self.last_time_reward_applicable(current_ts)
            .checked_sub(self.last_checkpoint_ts)?,
    ))
}
```

Recommendation: Consider changing the checks to strictly greater than, see: `require!(current_ts > self.last_checkpoint_ts, InvalidTimestamp);`

QSP-7 `NewRewarder`'s `Validate` should also verify `claim_fee_token_account` owner and mint

Severity: *Informational*

Status: Fixed

File(s) affected: `programs/quarry-mine/src/account_validators.rs`

Description: After `claim_fee_token_account`'s `assert_ata!` it is expected to have the following checks as well:

```
assert_ata!(
    self.claim_fee_token_account,
    self.rewarder,
    self.rewards_token_mint
);
//Missing checks
assert_keys!(self.claim_fee_token_account.owner, self.rewarder, "fee account owner");
assert_keys!(self.claim_fee_token_account.mint, self.rewards_token_mint, "fee account mint");
//End missing checks
```

This is how this is done for `CreateMiner` (`programs/quarry-mine/src/account_validators.rs:L133-L135`):

```
assert_ata!(self.miner_vault, self.miner, self.token_mint, "miner vault");
assert_keys!(self.miner_vault.owner, self.miner, "miner vault owner");
assert_keys!(self.miner_vault.mint, self.token_mint, "miner vault mint");
```

Recommendation: Consider adding the missing checks.

QSP-8 Possible truncation and precision loss in the annual rewards rate calculation

Severity: *Informational*

Status: Fixed

File(s) affected: `programs/quarry-mine/src/rewarder.rs:L21, programs/quarry-mine/src/rewarder.rs:L67`

Description: Possible truncations on casts to `u64`:

1.
```
(self.annual_rewards_rate as u128)
    .checked_mul(quarry_rewards_share as u128)?
    .checked_div(self.total_rewards_shares as u128)?
    .to_u64()
```

2.
```
let max_claim_fee = unwrap_int!((amount_claimable as u128)
                    .checked_mul(max_claim_fee_kbps.into())
                    .and_then(|f| f.checked_div((10_000 * 1_000) as u128))
                    .and_then(|f| f.to_u64()));
```

Recommendation: Please make sure that this level of truncation is intended.

Update: Fixed by increasing the precisions.

QSP-9 Privileged roles and ownership

Severity: *Informational*

Status: Acknowledged

Description: The programs deployed are not immutable until they are marked as "final" (i.e., not upgradeable) on the Solana blockchain by the admin team.

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: The admin team stated that a DAO that works on upgrading programs is planned.

QSP-10 Quarry inherits Anchor vulnerabilities

Severity: *Informational*

Status: Unresolved

Description: Anchor is a new and constantly evolving framework that was not audited. Quarry will inherit potential vulnerabilities in Anchor. This audit is strictly limited to Quarry.

Recommendation: The risk should be taken into consideration. Consider having a pure freeze functionality, possibly without relying on Anchor.

QSP-11 `NewRewarder.authority` should be the signer

Severity: *Informational*

Status: Acknowledged

File(s) affected: `programs/quarry-mine/src/lib.rs:L60-L61`

Description: In the `new_rewarder` instruction the authority is set to an arbitrary account. The validation only verifies the signature of the `base` account.

```
rewarder.authority = ctx.accounts.authority.key();
```

In the `NewRewarder`'s `Validate` there is a require statement that checks if the base is signer and not the authority `programs/quarry-mine/src/account_validators.rs:L22`

```
require!(self.base.is_signer, Unauthorized);
```

Authority's account is also defined as unchecked:

```
pub struct NewRewarder<'info> {
    ...
    /// Initial authority of the rewarder.
    pub authority: UncheckedAccount<'info>,
    ...
```

Not having the authority to sign the transaction can potentially lead to a faulty rewarder with locked authority.

Recommendation: Consider forcing the authority to be the signer of the transaction.

Update: The admin team decided not to check it in order to keep it flexible for future works.

QSP-12 `MergeMiner.owner` is not checked

Severity: Informational

Status: Acknowledged

File(s) affected: `programs/quarry-merge-mine/src/processor/init.rs:L35`

Description: `MergeMiner.owner` is marked as unchecked but set as the owner of newly created merge-miners. Unchecked accounts may cause crashes in programs.
`programs/quarry-merge-mine/src/lib.rs:L159`

```
pub struct InitMergeMiner<'info> {
    ...
    /// Owner of the [MergeMiner].
    pub owner: UncheckedAccount<'info>,

...
```

`programs/quarry-merge-mine/src/processor/init.rs:35`

```
pub fn init_merge_miner(ctx: Context<InitMergeMiner>, bump: u8) -> ProgramResult {
   ...
    mm.owner = ctx.accounts.owner.key();
```

Recommendation: Consider checking the owner account for correctness.

Update: The admin team decided not to check it in order to keep it flexible for future works.

QSP-13 Missing check to match `quarry.token_mint_key` and `miner_vault.mint`

Severity: Undetermined

Status: Fixed

File(s) affected: `programs\quarry-mine\src\account_validators.rs`

Description: On `L135` in `programs\quarry-mine\src\account_validators.rs`, the function does not check if `quarry.token_mint_key` equals to `miner_vault.mint`. Although `UserStake` enforces that check to the transaction so this risk factor might not be easily exploited, this factor could still be leveraged by an attacker to introduce other more complex attacks that at this moment are still hard to think of.

Recommendation: Add the mentioned check to the code.

QSP-14 Graceful account closure

Severity: Informational

Status: Acknowledged

Description: The implemented programs do not gracefully close subject-to-rent (unless exempted) accounts when they are no longer needed. This leads to having funds locked into PDAs even after they expire:

- In the `quarry-mine` program three accounts, viz. `NewRewarder.rewarder`, `CreateQuarry.quarry`, and `CreateMiner.miner`, have the `init` attribute and thus are checked by Anchor to see whether they meet the rent exemption minimum lamports.
- In the `quarry-mint-wrapper` program two accounts, namely `NewMinter.minter` and `NewWrapper.mint_wrapper` have the `init` attribute.
- `quarry-mine-merge` program has three accounts, namely `NewPool.pool`, `NewPool.replica_mint`, `InitMergeMiner.mm`, marked with the `init` attribute.

A graceful account deletion includes sending the rent-exemption lamports to an arbitrary account.

Recommendation: Confirm whether the listed accounts need not to be closed at any point. Consider adding logic for closing accounts. Reference: Anchor offers attributes for closing accounts upon instruction execution: `#[account(close = <target>)]`.

Update: The admin team stated that there is no need to close any account at this moment.

QSP-15 `InitMergeMiner.validate` does not explicitly call validate on the pool account

Severity: Undetermined

Status: Acknowledged

File(s) affected: `programs/quarry-merge-mine/src/account_validators.rs:L44-L48`

Description:

```
pub struct InitMergeMiner<'info> {
    /// [MergePool] of the underlying LP token.
    pub pool: Account<'info, MergePool>,
    ...
}
...
impl<'info> Validate<'info> for InitMergeMiner<'info> {
    fn validate(&self) -> ProgramResult {
        Ok(())
    }
}
```

Recommendation: Please make sure if this is intended.

Update: The admin team stated that there are no checks required to be added for this request at this moment.

Automated Analyses

Rust Audit

No findings.

Rust-Clippy

No findings.

Adherence to Best Practices

1. [fixed] `programs\quarry-merge-mine\src\processor\init.rs`: `L43`: should also set `primary_balance` as zero.
2. [acknowledge] `This quarry.token_mint_decimals` is not really used anywhere in the code, please make sure if this is intended.
3. [fixed] `rewarder` is named `lord` in the staking logic: `programs/quarry-mine/src/quarry.rs:L68~L72`; for the sake of consistency use `rewarder` as the argument name
4. [fixed] `programs\quarry-mine\src\lib.rs`: `L125`: to increase code consistency, consider performing `assert_keys!(ctx.accounts.authority, next_authority, "pending authority");` within the `account_validators.rs` in the validation sequence instead.
5. [acknowledge] In `programs/quarry-mine`, why do `set_pause_authority` `transfer_authority` force an unpause? The authority should be transferable despite the pause status. An authority handover process may entail pausing the program followed with a `transfer_authority` call. `set_pause_authority` also requires an unpaused status. This would block the authority until the `pause_authority` decides to resume the program. Please make sure if this limitation to the `authority`'s power is intended. If this is intended, please add this information into the specification. It is suggested to clarify which instructions are pausable and the reasoning behind those decisions.

Test Results

Test Suite Results

All tests passed.

```
==== unit tests ====

Run cargo test --lib
   Compiling quarry-mint-wrapper v1.10.6 (/home/runner/work/quarry/quarry/programs/quarry-mint-wrapper)
   Compiling quarry-redeemer v1.10.6 (/home/runner/work/quarry/quarry/programs/quarry-redeemer)
   Compiling quarry-mine v1.10.6 (/home/runner/work/quarry/quarry/programs/quarry-mine)
   Compiling quarry-merge-mine v1.10.6 (/home/runner/work/quarry/quarry/programs/quarry-merge-mine)
   Compiling quarry-registry v1.10.6 (/home/runner/work/quarry/quarry/programs/quarry-registry)
   Compiling quarry-operator v1.10.6 (/home/runner/work/quarry/quarry/programs/quarry-operator)
    Finished test [unoptimized + debuginfo] target(s) in 12.44s
     Running unittests (target/debug/deps/quarry_merge_mine-855543320c032d4a)

running 1 test
     Running unittests (target/debug/deps/quarry_mine-52be9af31276fbff)
test test_id ... ok

test result: ok. 1 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s


running 15 tests
test addresses::fee_to::test_id ... ok
test addresses::fee_setter::test_id ... ok
test payroll::tests::test_rewards_earned_when_zero_tokens_deposited ... ok
test payroll::tests::test_sanity_check ... ok
test payroll::tests::test_sanity_check_off_by_one_case ... ok
test payroll::tests::test_accumulated_precision_errors_epsilon ... ok
test payroll::tests::test_wpt_with_zero_annual_rewards_rate ... ok
test payroll::tests::test_wpt_when_famine ... ok
test quarry::tests::test_lifecycle_one_miner ... ok
test rewarder::tests::test_compute_quarry_annual_rewards_rate ... ok
test quarry::tests::test_lifecycle_two_miners ... ok
test rewarder::tests::test_compute_quarry_rewards_rate_with_multiple_quarries_fixed ... ok
test rewarder::tests::test_compute_rewards_rate_when_total_rewards_shares_is_zero ... ok
test test_id ... ok
test rewarder::tests::test_compute_quarry_rewards_rate_with_multiple_quarries ... ok

test result: ok. 15 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 23.60s

     Running unittests (target/debug/deps/quarry_mint_wrapper-3582b07899406538)

running 1 test
test test_id ... ok

test result: ok. 1 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s

     Running unittests (target/debug/deps/quarry_operator-68a41d455aa3892a)

running 1 test
test test_id ... ok

test result: ok. 1 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s

     Running unittests (target/debug/deps/quarry_redeemer-f1a2808be2871a26)

running 1 test
test test_id ... ok

test result: ok. 1 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s

     Running unittests (target/debug/deps/quarry_registry-5e4fd2604669fc30)

running 1 test
test test_id ... ok

test result: ok. 1 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s

==== integration tests ====
Run nix shell .#ci --command yarn test:e2e
warning: Git tree '/home/runner/work/quarry/quarry' is dirty


  Famine
     ✓ Stake and claim after famine (6084ms)
Program ATokenGPvbdGVxr1b2hvZbsiqW5xWH25efTNsLJA8knL invoke [1]
Program log: Transfer 2039280 lamports to the associated token account
Program 11111111111111111111111111111111 invoke [2]
Program 11111111111111111111111111111111 success
Program log: Allocate space for the associated token account
Program 11111111111111111111111111111111 invoke [2]
Program 11111111111111111111111111111111 success
Program log: Assign the associated token account to the SPL Token program
Program 11111111111111111111111111111111 invoke [2]
Program 11111111111111111111111111111111 success
Program log: Initialize the associated token account
Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA invoke [2]
Program log: Instruction: InitializeAccount
Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA consumed 3449 of 181475 compute units
Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA success
Program ATokenGPvbdGVxr1b2hvZbsiqW5xWH25efTNsLJA8knL consumed 22634 of 200000 compute units
Program ATokenGPvbdGVxr1b2hvZbsiqW5xWH25efTNsLJA8knL success
Program QMNeHCGYnLVDn1icRAfQZpjPLBNkfGbSKRB83G5d8KB invoke [1]
Program QMWoBmAyJLAsA1Lh9ugMTw2gciTihncciphzdNzdZYV invoke [2]
Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA invoke [3]
Program log: Instruction: MintTo
Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA consumed 2883 of 166780 compute units
Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA success
Program log: 0Q/jA6yPVSActO8N8y33mB8DSiNi69wCB/i3KKVS/3YA5XWOAh1QKRXlpR0X5UiRLiqgwiKjThGU2dWnHJRReXiyfIXy9JFpWwEAAAAAAAAuH8VFA1DabPpRgGIuurkPTo2mAvOCmRbE3tx46Te8Xw==
Program QMWoBmAyJLAsA1Lh9ugMTw2gciTihncciphzdNzdZYV consumed 15575 of 174270 compute units
Program QMWoBmAyJLAsA1Lh9ugMTw2gciTihncciphzdNzdZYV success
Program QMWoBmAyJLAsA1Lh9ugMTw2gciTihncciphzdNzdZYV invoke [2]
Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA invoke [3]
Program log: Instruction: MintTo
Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA consumed 2883 of 146338 compute units
Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA success
Program log: 0Q/jA6yPVSActO8N8y33mB8DSiNi69wCB/i3KKVS/3YA5XWOAh1QKRXlpR0X5UiRLiqgwiKjThGU2dWnHJRReXiyfIXy9JFpAAAAAAAAAACpjGdsqcRkrtwEVYJhaKrx8BbFpQp5ld6i+iSzCn0H5g==
Program QMWoBmAyJLAsA1Lh9ugMTw2gciTihncciphzdNzdZYV consumed 15575 of 153828 compute units
Program QMWoBmAyJLAsA1Lh9ugMTw2gciTihncciphzdNzdZYV success
Program log: XQ9GqjCM1Ns7IonVJ/BFwFnJp4i1L1gCCm8/LU9cDrLEOqPc67anPfkZ9E3uOyqoLC0+UT2p6JAfi9BUpt3WdfD4v63EutyiRgjD8batxpuP6RE9XsM3cMjxeTr8fl+F/DWA81nf7UxbAQAAAAAAAAAAAAAAAAAyoWpYQAAAAA=
Program QMNeHCGYnLVDn1icRAfQZpjPLBNkfGbSKRB83G5d8KB consumed 68175 of 200000 compute units
  '11111111111111111111111111111111',
  '11111111111111111111111111111111',
  '11111111111111111111111111111111',
  '11111111111111111111111111111111',
  '11111111111111111111111111111111',
  '11111111111111111111111111111111',
  '11111111111111111111111111111111',
  '11111111111111111111111111111111',
  '11111111111111111111111111111111',
  '11111111111111111111111111111111',
  '11111111111111111111111111111111',
  '11111111111111111111111111111111',
  '11111111111111111111111111111111',
  '11111111111111111111111111111111',
  '11111111111111111111111111111111',
  '11111111111111111111111111111111',
  '11111111111111111111111111111111',
  '11111111111111111111111111111111',
  '11111111111111111111111111111111',
  '11111111111111111111111111111111',
  '11111111111111111111111111111111',
  '11111111111111111111111111111111',
  '11111111111111111111111111111111'
]
     ✓ create registry (1052ms)


  40 passing (3m)
```

Code Coverage

Coverage reports cannot be obtained at this moment due to tests that interact with the Solana blockchain.

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

```
12bedb3c73c416fa4c2425cc04775394ace9407c8e942b73e30b5e3b5bb8217f ./programs/quarry-mint-wrapper/src/account_validators.rs

d05e2a14577ad9fa5ec2af26f21d929f371f62ff7dfe381738b76ccc2eed142a ./programs/quarry-mint-wrapper/src/lib.rs

1e184c3234cddcfd8abbf80ca17c361b814a6c64e25644e56ce0f677db1caf15 ./programs/quarry-mint-wrapper/src/macros.rs

22814946d6a99e05d92f87b430725b2d2299a405023b49c5e9d1aeff0280a89e ./programs/quarry-mine/src/account_validators.rs

2645a9b5a09a76f155ad8c87554909941ec0e4302e8eeac503187e07bfa253f1 ./programs/quarry-mine/src/addresses.rs

5b9ea750d7487c8b17c7837cfdb77737874b296ffc92fe522465da82adb44df7 ./programs/quarry-mine/src/lib.rs

d4e50bc49ae86140dfeac909dd94f689ac7b1a9754e1c82d09638a4ef1224d82 ./programs/quarry-mine/src/macros.rs

7c4b9b467dbce7e514cbd794b1712e3f1d5cca3d2d346ba74497ff2e5d39ea01 ./programs/quarry-mine/src/payroll.rs

d839f5b39458d965e0a398760d0a5ad95cf92f914509de940f6ed4558b33a2f7 ./programs/quarry-mine/src/quarry.rs

fc21ecd9c3e1d3316e43c833386f20606dd2f1c20c25215b0e6a7d643f1ab7e3 ./programs/quarry-mine/src/rewarder.rs

59687f821ec648764ffa065d695f09fc5ba2343f5abba61ee9669c65372511aa ./programs/quarry-merge-mine/src/account_conversions.rs

9b3677aa47c02d2e3534f32ff7b8d6f23c3e09ef29adec96736d2fd7532cdea1 ./programs/quarry-merge-mine/src/account_validators.rs

dce54b5da2249db1c9469a6d4af7d0a04f9f1a7aea4e22437f6c81e654e90e18 ./programs/quarry-merge-mine/src/events.rs

df25c32f00afc043b8ead9036815af35731d86a5c6e4fde5fe2907ec555e0bef ./programs/quarry-merge-mine/src/lib.rs

1bcfb5b0c9b11c30e45d3ad47f354f5fa48e6cae43a68ce6049c6167a8882a86 ./programs/quarry-merge-mine/src/macros.rs

04dc12d3acb838b5ce2e9b28b99c4e0c3e60f4d2ea3164606df386d390cdbe98 ./programs/quarry-merge-mine/src/mm_cpi.rs

1cdd6f0b230896c4e85fd33e285ec5ccb0f190d4e45b1efbfc38b4edfa651502 ./programs/quarry-merge-mine/src/state.rs

27695c662d5355458e77539b9803ed76514a871fd0bd6ac646901cfcf430988c ./programs/quarry-merge-mine/src/processor/claim.rs

967aeee12f354ffd5ae430b2eaaaee64882e7ffeabb7d320be8864845e2fef5a ./programs/quarry-merge-mine/src/processor/deposit.rs

5538fc7774b33af1c8995fe1f692a5deb713002477c16d424a0b1e3940132fc5 ./programs/quarry-merge-mine/src/processor/init.rs

08776f54e30d760055ef100dfdd6d933bf57b01a1e339e0931d62b5550bc8d4a ./programs/quarry-merge-mine/src/processor/mod.rs

f454f27aea8765c21f6ff67b8be736af526c4c0fb58435dcfb2329b7d1d56eb9 ./programs/quarry-merge-mine/src/processor/withdraw.rs
```

Tests

```
f055c0e86c68f336c0867a580b42e0677c2af12c1e543600ab7a26fe1c843a95 ./tests/famine.spec.ts

903823a50e2ff917569185430d829826466b2091e7cfbc6596c9eb81b0e1df92 ./tests/mine.spec.ts

aabe1e332a70c2903d076c02fe39f9c672c63acd6dc9b2cbe4a9eddd1f425b23 ./tests/mineRewards.spec.ts

d44994c985a73dfc6549fb84904f994d38695367cc3dcbc28cadc8357bfbd9e3 ./tests/mintWrapper.spec.ts

29e098510d3d2fb26dbaa733988929cbbfe81039e9efa4a605337a9853f8705e ./tests/operator.spec.ts

f63694f06f08d47491df8979b41b284566c8cc2c86788ac3d064b18bf6fb21db ./tests/quarryMergeMine.spec.ts

ae71d16ef5760f515697a8e75425bc11c3b4a945faeabb3ef3bd6b276600a1d6 ./tests/quarryRedeemer.spec.ts

790135b5d0ae1339ea91321df9cbfd22be664596825b557c5adc9a712c11e4cc ./tests/quarryUtils.ts

4040866b1255f93d6fe4c5bb67b8aedccc751c58c0faddcd288097512efb21af ./tests/registry.spec.ts

c89ddfd239db1ab3f04924081fd69a9f212927054fc07c5e314a818710f6026f ./tests/utils.ts

1d3db80558916933c2a22ca98212a6e8a74bd75c96e4c75e35a20a72ae2b1e96 ./tests/workspace.ts
```

Changelog

- 2021-11-15 - Initial report
- 2021-12-08 - final 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.