Quantstamp Security Assessment Certificate

July 4th 2022 — Quantstamp Verified

Mirror World - Token Vesting

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

Executive Summary

Type	SPL Token Vesting on Solana
Auditors	Poming Lee, Senior Research Engineer Andy Lin, Senior Auditing Engineer
Timeline	2022-06-06 through 2022-07-04
Languages	Rust
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	None
Documentation Quality	Undetermined
Test Quality	Undetermined

Source Code

Repository	Commit
tkz-mirror-world	9b46848
tkz-mirror-world	2958df2

Total Issues	**8** (4 Resolved)
High Risk Issues	0 (0 Resolved)
Medium Risk Issues	**1** (1 Resolved)
Low Risk Issues	**3** (1 Resolved)
Informational Risk Issues	**4** (2 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).
Fixed	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

This audit is based on the code forked from the Chingari's token vesting contracts (specifically, the version of code at https://github.com/CoMakery/TokenLockup-SOL/tree/f8da8d5e0959f569ef75802e2a56d10171d79ee0). The differences are mainly for accommodating the Anchor version updates, also, a new utility function to get the current timestamp has been added. During the audit, Quantstamp has, on a best-effort basis, with 2 auditors working independently (and later on syncing on their findings), identified 8 issues of various levels of severity. 1 is of medium-severity, 3 are of low severity, 4 are informational findings. This report also lists several best practices recommendations. We highly recommend addressing the findings before going live.
2022-07-04 Update: The team fixed most of the issues and added clear documentation in `README.md` for those acknowledged ones.

ID	Description	Severity	Status
QSP-1	Unrestricted Data Modification Can Be Done by Utilizing `tokenlock_account`	^ Medium	Fixed
QSP-2	A `tokenlock_account` Created by a Project Admin Can be Modified by Anyone	⌄ Low	Acknowledged
QSP-3	Mistyped or Undertyped Accounts	⌄ Low	Acknowledged
QSP-4	Fishing With Fake `escrow_account`	⌄ Low	Fixed
QSP-5	Privileged Roles and Ownership	○ Informational	Acknowledged
QSP-6	The Program Inherits Anchor Vulnerabilities	○ Informational	Acknowledged
QSP-7	Abusing `transfer_spl` Utility Function	○ Informational	Fixed
QSP-8	Unclear Intention of the `Authority`	○ 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.

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

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
- 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:

- Cargo Audit 0.16.0
- Soteria build 1643895818

Steps taken to run the tools:

1. Installed via `cargo install cargo-audit`
2. Ran `cargo audit`
3. docker run -v ${ProjectDirectory}:/workspace -it greencorelab/soteria:latest /bin/bash
4. soteria .

Findings

QSP-1 Unrestricted Data Modification Can Be Done by Utilizing `tokenlock_account`

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `programs/tokenlock/src/lib.rs, programs/tokenlock/src/account.rs`

Description: The implementation uses a `TokenLockDataWrapper` to access the `tokenlock_account`. The code defines the `tokenlock_account` as the `AccountInfo` type to save the compute units. Consequently, the code will dismiss the Anchor framework's default checks for the `Account` type. This leads to the lack of verification for the discriminator and the executing program that owns the account. Missing the ownership validation allows the attacker to pass in an arbitrary `tokenlock_account` unrelated to this program without verifying the ownership. Secondly, missing the discriminator check is even more severe. The attacker can give in other accounts owned by the program and have it unexpectedly mutate the data.

Exploit Scenario: The attacker passes another account type such as `account.rs::TimelockData` as the `tokenlock_account` of `account.rs::ManagementTokenlock`. The `lib.rs::tokenlock::create_release_schedule` instruction will potentially mutate the wrong account in `TokenLockDataWrapper::add_schedule` if the `TimelockData` somehow was able to be parsed wrongly but passes the other checks.

Recommendation: We recommend adding validation that the owner of the `tokenlock_account` is the `tokenlock` program itself and verifying that the first `8` bytes from the account are exactly the expected discriminator. Anchor will automatically generate a function for the expected discriminator. The code can compare the first `8` bytes with `TokenLockData::discriminator()`. Following is the list of instructions using the `tokenlock_account` but does not have the validations in `lib.rs::tokenlock`:

- `initialize_timelock`
- `create_release_schedule`
- `fund_release_schedule`
- `transfer`
- `transfer_timelock`
- `cancel_timelock`

Update: The team adds the discriminator and checks all the instructions as pointed in the recommendation.

QSP-2 A `tokenlock_account` Created by a Project Admin Can be Modified by Anyone

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `programs/tokenlock/src/lib.rs, programs/tokenlock/src/account.rs`

Description: An attacker can call `pub fn create_release_schedule` to create numerous release schedules for any `tokenlock_account`, since the ownership of that `tokenlock_account` is never checked. Although this may not cause fund loss, this could still be used by an attacker as a tool to manipulate the platform to conduct a more complex attack.

Recommendation: Add an attribute to the `tokenlock_account` stating who is the authority of the account, and check the ownership in `pub fn initialize_tokenlock`, `pub fn create_release_schedule`, and `pub fn fund_release_schedule`.

Update: The admin team stated that: `tokenlockAccount` is created and initialized once during `initialize_tokenlock`. All data is immutable and release_schedules can be added but not updated or deleted. Release schedule can be created by anyone. New release schedule is added and limited by 65535 items. Anyone can fund release schedule for recepient. First `timelockAccount` must be created (`initialize_timelock`) or found. Than new timelock is added and `signer_hash` calculated and attached to it. Timelock data is immutable except `token_transfered` property which is changed on `cancel_timelock` or `transfer` call by calculated unlocked amount and verified ownership and access control.
2022-07-04: The team added the statement to the `README.md` inside the "Access control" section.

QSP-3 Mistyped or Undertyped Accounts

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `programs/tokenlock/src/account.rs`

Description: From reference to Anchor framework's recommendation, the following examples are mistyped or undertyped and should be elevated to the recommended:

1. Replace the deprecated `ProgramAccount` usage with `Account` instead in the `programs/tokenlock/src/acount.rs`. It is deprecated after Anchor `0.19.0`: change log
    1. `L328`: `InitializeTokenLock::tokenlock_account`
    2. `L356`: `InitializeTimeLock::timelock_account`
    3. `L387`: `FundReleaseSchedule::timelock_account`
    4. `L423`: `TransferFrom::timelock_account`
    5. `L457`: `TransferTimelock::timelock_account`
    6. `L491`: `CancelTimelock::timelock_account`
2. Use `Program<'info, Token>` for `token_program` in `account.rs`. See the Anchor doc: link.
    1. `L409`: `FundReleaseSchedule::token_program`.
    2. `L443`: `TransferFrom::token_program`
    3. `L477`: `TransferTimelock::token_program`

4. `L522`: `CancelTimelock::token_program`

3. Others
    1. On `L349`: `tokenlock_account` should be of type `Account<'info, TokenLockData>`.
    2. On `L373`: `tokenlock_account` should be of type `Account<'info, TokenLockData>`.
    3. On `L381`: `tokenlock_account` should be of type `Account<'info, TokenLockData>`.
    4. On `L417`: `tokenlock_account` should be of type `Account<'info, TokenLockData>`.
    5. On `L451`: `tokenlock_account` should be of type `Account<'info, TokenLockData>`.
    6. On `L485`: `tokenlock_account` should be of type `Account<'info, TokenLockData>`.
    7. On `L406`: `to`, should be of type `Account<'info, anchor_spl::token::TokenAccount>`
    8. On `L440`: `to`, should be of type `Account<'info, anchor_spl::token::TokenAccount>`
    9. On `L474`: `to`, should be of type `Account<'info, anchor_spl::token::TokenAccount>`
    10. On `L506`: `target`, should be of type `Account<'info, anchor_spl::token::TokenAccount>`
    11. On `L362`: `target_account`, should be of type `Account<'info, anchor_spl::token::TokenAccount>`

Update:

1. For items `3.1-3.6`, the risks are mitigated based on the additional checks currently implemented in the program.
2. For items `3.10-3.11`, the issues remain unresolved.
3. All other items are fixed.

QSP-4 Fishing With Fake `escrow_account`

Severity: *Low Risk*

Status: Fixed

File(s) affected: `programs/tokenlock/src/lib.rs`

Description: On `lib.rs::L42-49`, the code finds the `_pda` and the `bump_seed` by calling `Pubkey::find_program_address` and saves the `bump_seed` to the `token_account`. The `_pda` is supposed to be the `owner` of the token account: `escrow_account`. However, the code did not check this. As a result, it is possible to provide an unrelated `escrow_account` here, and it will lead to issues later when trying to transfer the funds in the following instructions of `lib.rs::tokenlock`: `transfer`, `transfer_timelock`, and `cancel_timelock` because of not having sufficient permission to operate on the `escrow_account`. Also, this opens a fishing surface for the attacker to trick the victims into lock-in funds to a fake `escrow_account`.

Exploit Scenario:

1. The attacker calls the `initialize_tokenlock` instruction with an `escrow_account` that the attacker owns.
2. The attacker calls `initialize_timelock` to set up a `timelock`.
3. The victim thinks the `tokenlock` and `timelock` accounts are valid and uses those to call `create_release_schedule` and `fund_release_schedule`.
4. Once the token is transferred to the fake `escrow_account` that the attacker owns, the attacker takes away the fund.

Recommendation: In `lib.rs::tokenlock::initialize_tokenlock`, the code should check that:

1. The `owner` of the token account `escrow_account` is the `_pda` pub-key from the `Pubkey::find_program_address`.
2. The `delegate == None` or `delegated_amount == 0` for the token account `escrow_account`.

QSP-5 Privileged Roles and Ownership

Severity: *Informational*

Status: Acknowledged

File(s) affected: `programs/tokenlock/src/lib.rs`

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 team added clear instructions in the `README.md`. It includes how to deploy with the `final` flag, and how to check whether the program is immutable. Also, it states that if the program is not deployed with the `final` flag, the operating team should share the update of the program with the users.

QSP-6 The Program Inherits Anchor Vulnerabilities

Severity: *Informational*

Status: Acknowledged

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

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

Update: The team mentioned that they are aware of the risk. The team uses Anchor `0.24.0` and implements a custom wrapper class to overcome Anchor's reading/writing overhead.

QSP-7 Abusing `transfer_spl` Utility Function

Severity: *Informational*

Status: Fixed

File(s) affected: `programs/tokenlock/src/utils.rs, programs/tokenlock/src/lib.rs`

Description: The `utils.rs::transfer_spl` function would sign a PDA with the hardcoded seed of `TOKENLOCK_PDA_SEED`, `mint_address`, `tokenlock_account` and `bump_seed`. The seed above is exactly the seed for the owner of the `escrow_account` (token account). As a result, the intention of the function seems to be transferring SPL token "from" the escrow account, so it requires the signature of the PDA.
However, in the `lib.rs::tokenlock::fund_release_schedule` instruction, the `transfer_spl` is used to transfer from other accounts to the escrow account. The PDA signature here is unnecessary, and the usage of the function here is misleading. The part works because the `authority` account needs to sign when calling the `fund_release_schedule` instruction. So it will automatically sign the transfer. Note that this pattern can be dangerous as one can try to pass in the `escrow_account` as the `FundReleaseSchedule::from` to attack. The `transfer_spl` will potentially transfer funds from the `escrow_account` unexpectedly. Luckily, the constraint `from.owner == *authority.key` on the `FundReleaseSchedule::from` would stop this from happening as one cannot get the PDA signature outside that program.

Recommendation: Consider renaming the utility function as `transfer_spl_from_escrow` and refactor the `fund_release_schedule` instruction not to use this utility function and call simple transfer from the user.

Update: The team refactored the `utils.rs` to have `transfer_spl_from_escrow` and `transfer_spl`. `transfer_spl` is the simple transfer without signing used inside the instruction `fund_release_schedule`. All the other instructions are refactored to use the function `transfer_spl_from_escrow` instead.

QSP-8 Unclear Intention of the `Authority`

Severity: *Informational*

Status: Fixed

File(s) affected: `programs/tokenlock/src/account.rs, programs/tokenlock/src/lib.rs`

Description: The `account.rs::InitializeTokenLock` struct has the field `authority` as a `Signer`. However, the instruction `lib.rs::tokenlock::initialize_tokenlock` never uses the `authority` field inside. It is unclear what the intention is to ask for the signature, and anyone can sign the transaction to fulfill the signature check.

Recommendation: Please verify and clarify the intention of the `authority` account here. If the team expects only a particular actor to call the instruction, please consider validating this against a hardcoded Pub-key. Otherwise, please remove the `authority` field from the `InitializeTokenLock` struct. Note that since anyone can call the `initialize_tokenlock` instruction, the team should document this in the official accounts.

Update: The `authority` field is removed.

Automated Analyses

Cargo Audit

No warning for crates that are directly used by the contracts were found.

Soteria

Soteria was able to found `13` unsafe arithmetic issues. The reported issues were analyzed and applicable ones were included in the report.

Code Documentation

1. [fixed] Please add an explanation of the account with the `/// CHECK` comments in `programs/tokenlock/src/account.rs`:
    1. `L361`: `InitializeTimeLock::target_account`
    2. `L405`: `FundReleaseSchedule::to`
    3. `L409`: `FundReleaseSchedule::token_program`. However, it is better to use `Program<'info, Token'>` here.
    4. `L427`: `TransferFrom::pda_account`
    5. `L433`: `TransferFrom::authority`
    6. `L443`: `TransferFrom::token_program`. However, it is better to use `Program<'info, Token'>` here.
    7. `L461`: `TransferTimelock::pda_account`
    8. `L467`: `TransferTimelock::authority`
    9. `L477`: `TransferTimelock::token_program`. However, it is better to use `Program<'info, Token'>` here.
    10. `L495`: `CancelTimelock::pda_account`
    11. `L499`: `CancelTimelock::authority`
    12. `L505`: `CancelTimelock::target`
    13. `L522`: `CancelTimelock::token_program`. However, it is better to use `Program<'info, Token'>` here.

Adherence to Best Practices

1. [ack] Consider rename `unlocked_balance_of`in `programs/tokenlock/src/account.rs` into `claimable_balance_of` to better reflect what it really is.
2. [ack] Consider rename `unlocked_balance_of_timelock` in `programs/tokenlock/src/account.rs` into `claimable_balance_of_timelock` to better reflect what it really is.
3. [ack] It is unclear why in `programs/tokenlock/src/account.rs::L352` the space requirement is `10240`, and it seems that `10240` is used to max out the 10Kib constraint, instead of something that is intentionally designed.
4. [ack] Can `delay_until_first_release_in_seconds` be zero?
5. [fixed] `programs/tokenlock/src/account.rs::CancelTimelock::authority` only has a single account constraint of the signer. Consider using the `Signer` type instead, as recommended by the Anchor doc.

6. [fixed] Remove the empty account constraint on `programs/tokenlock/src/account.rs::L483` of the `CancelTimelock::tokenlock_account` field.
7. [fixed] In `programs/tokenlock/src/lib.rs::L250`, consider using a constant to replace the magic number of `32` when calculating the `need_space`. For instance, `account.rs::PUBKEY_SIZE` might be a good candidate.
8. [fixed] In `programs/tokenlock/src/lib.rs::L184-200`, consider to return the `Err(TokenlockErrors::DuplicatedCancelable.into())` directly when `cancelable_by[i] == cancelable_by[j]` inside the for loop. This removes the need to have the extra variable of `duplicate` and simplifies the code to be without a `break` statement.

Test Results

Test Suite Results

All tests have passed.

```
=========JS Tests:

  TokenLockup stress test
create schedule= 0
create schedule= 1
create schedule= 2
create schedule= 3
create schedule= 4
create schedule= 5
create schedule= 6
create schedule= 7
create schedule= 8
create schedule= 9
create schedule= 10
create schedule= 11
create schedule= 12
create schedule= 13
create schedule= 14
create schedule= 15
create schedule= 16
create schedule= 17
create schedule= 18
create schedule= 19
create schedule= 20
create schedule= 21
create schedule= 22
create schedule= 23
create schedule= 24
create schedule= 25
create schedule= 26
create schedule= 27
create schedule= 28
create schedule= 29
create schedule= 30
create schedule= 31
create schedule= 32
create schedule= 33
create schedule= 34
create schedule= 35
create schedule= 36
create schedule= 37
create schedule= 38
create schedule= 39
create schedule= 40
create schedule= 41
create schedule= 42
create schedule= 43
create schedule= 44
create schedule= 45
create schedule= 46
create schedule= 47
create schedule= 48
create schedule= 49
create schedule= 50
create schedule= 51
create schedule= 52
create schedule= 53
     ✓ initialize timelock with wrong tokenlock account (484ms)
     ✓ create release schedule with wrong tokenlock account (440ms)
     ✓ fund release schedule with wrong tokenlock account (1288ms)
     ✓ transfer with wrong tokenlock account (1771ms)
     ✓ transfer timelock with wrong tokenlock account (1707ms)
  TokenLockup check cancelables
     ✓ should emit an event with the correct scheduleId when the release schedule is funded and canceled (2528ms)
    Check cancel timelock after funding with multi cancelable addresses
       ✓ cancel with first canceler (415ms)
       ✓ cancel with second canceler (420ms)
       ✓ cancel with third canceler (429ms)
       ✓ cancel with non canceler reverts (67ms)
       ✓ timelock index not change after canceling (416ms)
    simple 1 month delay then 50% for 2 monthly releases
       ✓ should be able to check if the lockup is cancelable (817ms)
       ✓ 0% unlocked at start and 100% cancelable (1269ms)
       ✓ only canceler can cancel (865ms)
       ✓ cannot cancel a non existent timelock (897ms)
       ✓ only canceler of timelock can cancel (1280ms)
  TokenLockup fund release schedues
     ✓ fundReleaseSchedule emits a ScheduleFunded event (1705ms)
     ✓ timelock creation with immediately unlocked tokens (1273ms)
     ✓ must have more tokens than there are release periods (866ms)
     ✓ must have more tokens than minReleaseScheduleAmount (861ms)
     ✓ cannot specify non existent schedule id (872ms)
     ✓ returns true after fundReleaseSchdule is called (1252ms)
     ✓ cannot specify a commencement time after the allowed range (867ms)
     ✓ cannot specify a schedule with a delay until first release that is greater than the max release delay
     ✓ can specify a schedule with a delay up to the max release delay (441ms)
     ✓ cannot fund a batch after the allowed range (916ms)
     ✓ fundReleaseRelease can be scheduled in the past (1268ms)
  TokenLockup create release schedule
     ✓ inject pretender wallet as escrow account (906ms)
     ✓ tokenlock successfully created (852ms)
  TokenLockup timelock balances
     ✓ timelock creation with immediately unlocked tokens (1268ms)
     ✓ can return all balance types of locked and unlocked tokens in multiple release schedules (2123ms)
     ✓ it can set a schedule to a balance in the past (1698ms)
     ✓ creating a timelock increases the totalSupply and transferring decreases it (2953ms)
     ✓ it can set a schedule to a balance in the future within the maxCommencementTimeInSeconds (1726ms)
  Transfer Negative cases
     ✓ Injection of Any Wallet with Linked Account and pretend to be a signer
     ✓ Injection of Any Wallet with Linked Account and original signer
     ✓ Injection of Any Wallet with original signer and account
  77 passing (9m)

=========Rust Tests:

running 6 tests
test different_initial_release_portion_in_bips ... ok
test different_delay_until_first_release_in_seconds ... ok
test different_period_between_releases_in_seconds ... ok
test different_release_count ... ok
test different_signer_hash ... ok
test equal_release_schedules ... ok
test result: ok. 6 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.03s
Jun 24 07:57:54.163 INFO cargo_tarpaulin::process_handling::linux: Launching test
Jun 24 07:57:54.163 INFO cargo_tarpaulin::process_handling: running /home/runner/work/tkz-mirror-world/tkz-mirror-world/target/debug/deps/utils-f42aa95fac765f3c
running 2 tests
test singer_hash_for_zero_nonce ... ok
test singer_hash_for_ff_nonce ... ok
test result: ok. 2 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.01s
Jun 24 07:57:55.842 INFO cargo_tarpaulin::process_handling::linux: Launching test
Jun 24 07:57:55.843 INFO cargo_tarpaulin::process_handling: running /home/runner/work/tkz-mirror-world/tkz-mirror-world/target/debug/deps/tokenlock-755ec599d3a8c63b
running 8 tests
test test::test_create_release_schedule ... ok
test test::test_fund_release_schedule ... ok
test test::test_cancel_timelock ... ok
test test::test_initialize_timelock ... ok
test test::test_initialize_tokenlock ... ok
test test::test_transfer_timelock ... ok
test test::test_transfer ... ok
test test_id ... ok
test result: ok. 8 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.11s
```

Code Coverage

The combined coverage report for both tests in JavaScripts and Rust cannot be obtained at this moment. It is highly recommended to solve this issue whenever it is possible and to have code coverage to at least 90%, in order to avoid functional bugs that are not necessarily security issues.

```
Jun 24 07:57:58.669 INFO cargo_tarpaulin::report: Coverage Results: || Uncovered Lines: || programs/tokenlock/src/account.rs: 76, 81, 135-136, 169, 190, 220, 242, 254, 260-261, 263, 295, 298-302, 308-309, 311, 314-315, 317 || programs/tokenlock/src/lib.rs: 18, 49, 69, 93, 121, 139-140, 160, 168, 171, 182, 186, 192, 204, 211, 241, 248, 260, 270, 300, 308, 316, 319, 348-350, 365, 388, 396, 404, 407, 445, 453, 456, 470 || programs/tokenlock/src/test.rs: 127, 136, 153, 157, 183, 198, 202, 206, 232, 236, 240, 266, 270, 274, 300, 304, 308, 318, 321, 402 || programs/tokenlock/src/wrappers.rs: 8-10, 12-13, 19-21, 23-25, 43-45, 47, 79-80, 82, 84-87, 121, 135 || Tested/Total Lines: || programs/tokenlock/src/account.rs: 75/99 || programs/tokenlock/src/lib.rs: 211/246 || programs/tokenlock/src/test.rs: 480/500 || programs/tokenlock/src/utils.rs: 30/30 || programs/tokenlock/src/wrappers.rs: 54/78 || programs/tokenlock/tests/release_schedule.rs: 30/30 || programs/tokenlock/tests/utils.rs: 10/10 || 89.63% coverage, 890/993 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

```
0c1277f796da425b51318af25d9befac7578ea4095e7e9e593537da5583c6f8f ./src/lib.rs
64291cb58f7c38c95d11b33e3bf7f43bc62a6213ed228c7c90aeb6a340ae4952 ./src/wrappers.rs
1f7206635ef2944a711be9dc44184cfef701fed93d6a9bef2a3ebbb1a0dff01f ./src/error.rs
b1328076756c2f472e8de6f73e158cb551e420b683df8283c3fcbd96df6f6886 ./src/account.rs
a420310297dc365ef3f7ab1e8761b18e170fadf2faca0658ff71532ac44610ff ./src/utils.rs
```

Tests

```
91fc55792a8fbf5ed057ca2868229c0b8470b422e302e9209e9c245d2e01182e ./test.rs
f46790988134c02259ba77fb350be44453ee78d23c96e82411de5e35a45d78aa ./release_schedule.rs
4009aa4d5e02199ec19579020ba9f211923887883edd862914b43c838281a0c1 ./utils.rs
286616af60e6fcfbbe02d2ed7228e92d8308efa3a069bbcc377b82bc6c60557a ./tests/lib.js
3de5ec966ce9db8b6ebb280f1849dc0b320725ca8fcbf1411c44926775084749 ./tests/utils.js
2f318dd2b8ae2a84a7109ce464594d3614aec493640b70b5a8fe4c6d18340232 ./tests/lib/lib.js
39827809119d3af2642d28540352440c9e2b79d65fd1db32498b082555fd93b0 ./tests/lib/utils.js
d97b263bdb3ec26a9ea24fb844741e412b1d82538f7cb36c9aa6a3ca75fefb9e ./tests/tokenlock/transfer-negative.test.js
dc0c9fc6080a29639cc5d5833e114272555702750c8101f9b7b230685aeee8a8 ./tests/tokenlock/createReleaseSchedule.test.js
62748a6f607f58b0a71fc5cc8906dfb27890fe51c1e817e516ba2449d44d2129 ./tests/tokenlock/fundReleaseSchedule.test.js
c31d940b7e4bc0fcaa481e3e41e86c0f2bf65e3df157b88b4a7fff8bb6b13247 ./tests/tokenlock/fundCancelableReleaseSchedule.test.js
b98a0a8cc5fad4f037b8fd7291fd17311e563538cff31a8ff92b5b0cc8ddfb50 ./tests/tokenlock/batchFundReleaseSchedule.test.js
b8b8a1542631b72dce24aa7d6825afd533f265e3ed66ed663476df9ffa190fbb ./tests/tokenlock/100-timelocks.test.js
69aa676403369302a5fa6b18b55c58bb23f501eca9a35bd9bb4f4fcde806775c ./tests/tokenlock/locked-and-unlockedBalanceOf.test.js
ddf544193e17a6ded77bbe80f0dd0a5ad3525d377b3e2d290e091da292067c79 ./tests/tokenlock/initializeTokenlock.test.js
faf0d3d8b9543fab7a49a80e5b48c544ce313150fec28334e48ca88a4ef56a75 ./tests/tokenlock/calculateUnlocked.test.js
848044c680bd0efcacc90623997cbfbed161ff317e4a6b6b70ad61a4bc3b1b6c ./tests/tokenlock/discriminatorValidation.test.js
```

Changelog

- 2022-06-10 - Initial report
- 2022-07-04 - 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.