Quantstamp Security Assessment Certificate

December 18th 2020 — Quantstamp Verified

Equilibrium

This security audit was prepared by Quantstamp, leaders in blockchain security and enterprise solutions.

Executive Summary

Type	Cross-chain money market based on Substrate
Auditors	Sebastian Banescu, Senior Research Engineer Luís Fernando Schultz Xavier da Silveira, Security Consultant Joseph Xu, Technical R&D Advisor
Timeline	2020-10-28 through 2020-12-04
Languages	Rust
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	None

Source Code

Repository	Commit
equilibrium-substrate-chain	1381045

Total Issues	**27** (24 Resolved)
High Risk Issues	**2** (2 Resolved)
Medium Risk Issues	**6** (6 Resolved)
Low Risk Issues	**6** (5 Resolved)
Informational Risk Issues	**6** (5 Resolved)
Undetermined Risk Issues	**7** (6 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

After audit: Quantstamp has audited only the `eq-aggregates`, `eq-balances`, `eq-claim` and `eq-vesting` pallets, which are part of the Equilibrium Substrate chain and has identified 20 security issues ranging from high to informational severity. Additionally, 7 more issues have been marked as having undetermined severity due to lack of clarity stemming from the missing documentation. However, we believe that depending on the use-cases that may be supported, some of these issues with undetermined severity might turn out to have a high security impact. Moreover, 11 issues in the code comments and 26 deviations from best practices were identified.

Quantstamp has not received a formal specification or documentation that explain how the claims and vesting logic should work. Therefore, we have identified the issues based on existing Substrate documentation, existing comments in the code and our own domain knowledge.

We confirm that transfers are disabled by default, however, the root account can enable and disable transfers at any time.

We recommend addressing all the identified issues as well as improving the test suite before deploying the system in production.

After 1st reaudit: The 2 high severity issues have been addressed in commit `9ab3c587`. For more details about how they were addressed please read the update messages posted below the recommendation, for each of those issues. The other issues have not been resolved.

After 2nd reaudit: The report has been updated according to the code commit `9dfcacf`. 17 of the 27 issues were marked as resolved. For the issues that remain unresolved or acknowledged we have added an update below their corresponding recommendation which indicates their current status. Additionally, there have been improvements of the test suite, best practices and code comments, all of these are marked accordingly.

After 3rd reaudit: The report has been updated according to the code commit `27ce403`. We have marked 17 of the 27 issues as resolved, 3 issues as acknowledged and 7 issues as mitigated. Mitigation has been done using the compiler flag `feature = "production"`. This flag is meant to be used for production builds and end-users can verify if its value is set correctly using the `check_production()` extrinsic.

ID	Description	Severity	Status
QSP-1	Incorrect `vesting_balance` implementation	High	Fixed
QSP-2	Incorrect vesting genesis	High	Fixed
QSP-3	Statement hash does not match expected hash	Medium	Fixed
QSP-4	Initial vesting configurations with no funds locked possible	Medium	Fixed
QSP-5	Ignored return values	Medium	Mitigated
QSP-6	Insufficient clean-up may lead to incorrect state	Medium	Mitigated
QSP-7	Locked tokens	Medium	Fixed
QSP-8	Errors may occur after extrinsics write to storage	Medium	Mitigated
QSP-9	Missing input validation	Low	Mitigated
QSP-10	Incorrect value passed when emitting event	Low	Fixed
QSP-11	Not all funds may be released due to truncation of `per_block`	Low	Fixed
QSP-12	Integer Overflow / Underflow	Low	Mitigated
QSP-13	Loss of precision during conversion of balance	Low	Fixed
QSP-14	Missing checks for dead accounts	Low	Acknowledged
QSP-15	Privileged Roles and Ownership	Informational	Mitigated
QSP-16	Unused error values	Informational	Fixed
QSP-17	Unused `enum`	Informational	Fixed
QSP-18	Loosely constrained type used for balances	Informational	Fixed
QSP-19	Functions that can set the balance of any account	Informational	Fixed
QSP-20	Unimplemented features	Informational	Acknowledged
QSP-21	Unused input parameter	Undetermined	Fixed
QSP-22	Same account can get a new vesting schedule	Undetermined	Fixed
QSP-23	Positive zero is different from negative zero	Undetermined	Fixed
QSP-24	The `Unknown` currency can be used	Undetermined	Fixed
QSP-25	The `make_free_balance_be` function does not update aggregates	Undetermined	Mitigated
QSP-26	Aggregates not updated when account is killed	Undetermined	Fixed
QSP-27	Incorrect imbalance type returned	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):

- 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
- Privileged roles and ownership
- 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 software system.
   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 code 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 system.

Toolset

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

Setup

Tool Setup:

- Rust Audit v0.13.1
- Rust-Clippy Latest

Steps taken to run the tools:

1. `cargo install cargo-audit`
2. `cargo audit`
3. `rustup component add clippy`
4. `cargo clippy`

Findings

QSP-1 Incorrect `vesting_balance` implementation

Severity: *High Risk*

Status: Fixed

File(s) affected: `pallets/eq-vesting/src/lib.rs`, `pallets/eq-claim/src/lib.rs`

Description: The `vesting_balance` function indicates that it can: "Get the amount that is currently being vested and cannot be transferred out of this account." However, this is not correct, because the vesting funds are not locked into each account in this implementation. Instead, the vesting pallet holds all the funds that are vested to various accounts. This means that each of those accounts which are being vested can transfer any amount of funds they have, because those funds already constitute vested funds.

The `vesting_balance` function is called in `process_claim` on L485 in `pallets/eq-claim/src/lib.rs` in order to decide if a claim can be processed or if the following error should be returned: "The account already has a vested balance." and `Error::<T>::VestedBalanceExists`, which would impact all end-users by denying them the claim of their rightfully vested amounts.

Recommendation: One option is to remove the use of `vesting_balance` and hence the function itself from the code base. Another option is to adjust the function implementation and its use accordingly.

Update: The issue has been fixed by changing the implementation of `vesting_balance`. However, together with this fix the comments of that function and the comments of the `locked_at` function (which is called in `vesting_balance`) were also updated. We believe these comments were wrongly updated and the old versions of the comments (stemming from Substrate templates) were more accurate. The only inaccuracy in the old comment (see description above) was "this account", which should be made more precise by indicating that it refers to the address of the vesting module.

QSP-2 Incorrect vesting genesis

Severity: *High Risk*

Status: Fixed

File(s) affected: `pallets/eq-vesting/src/lib.rs`

Description: The `add_extra_genesis` code block for the `eq-vesting` pallet incorrectly computes the `locked` amount using the current balance of the `who` address and subtracting `liquid` from it. This is not correct, because the vesting funds are not locked into each account in this implementation. Instead, the vesting pallet holds all the funds that are vested to various accounts. This means that each of those accounts which are being vested can transfer any amount of funds they have, because those funds already constitute vested funds.

Recommendation: Perform a correct computation of the `locked` amount for each `who` account. Additionally, the `assert` statement on L132 is not necessary.

Update: This issue has been mitigated by removing the `add_extra_genesis` block containing the faulty code.

QSP-3 Statement hash does not match expected hash

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `pallets/eq-claim/src/lib.rs`

Description: The `get_statement_text` function indicates that the "SHA-256 multihash" of the statement located at the following URL https://equilibrium.io/tokenswap/docs/token_swap_t&cs.pdf is `Qmc1XYqT6S39WNp2UeiRUrZichUWUPpGEThDE6dAb3f6Ny`. However, at the time of this writing, the hash of that document is different.
For readers and users not familiar with the multihash format it is summarized on the reference multihash Github repo as "Multihash is a protocol for differentiating outputs from various well-established cryptographic hash functions, addressing size + encoding considerations." The following example is given:

```
# sha2-256 0x12 - sha2-256("multihash")
12209cbc07c3f991725836a3aa2a581ca2029198aa420b9d99bc0e131d9f3e2cbe47 # sha2-256 in hex
CIQJZPAHYP4ZC4SYG2R2UKSYDSRAFEMYVJBAXHMZXQHBGHM7HYWL4RY= # sha256 in base32
QmYtUc4iTCbbfVSDNKvtQqrfyezPPnFvE33wFmutw9PBBk # sha256 in base58
EiCcvAfD+ZFyWDajqipYHKICkZiqQgudmbwOEx2fPiy+Rw== # sha256 in base64
```

From this example it can be seen that the hash given in `get_statement_text` is SHA-256 encoded in base58. If we apply the same hash function SHA-256 with the base58 encoding to the document located at https://equilibrium.io/tokenswap/docs/token_swap_t&cs.pdf, we obtain the following value: `Dxr128rDgHEgs4p1Zb4zbmHj3bjns1B8i9hdLjTpHzXs`, which is different from the hash value hardcoded into the `get_statement_text` function.

Recommendation: Update the hardcoded hash value in the `get_statement_text` function.

QSP-4 Initial vesting configurations with no funds locked possible

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `pallets/eq-vesting/src/lib.rs`

Description: The configuration in the `add_extra_genesis` code block in `pallets/eq-vesting/src/lib.rs` does not check if the `liquid` parameter is greater than `balance`, which would lead to a value of `locked` equal to zero. This would also mean `per_block` is also equal to zero. Such a vesting structure would not make sense to add because it will not be vesting any amount of funds effectively.

Recommendation: Do not add `VestingInfo` where the `locked` and/or `per_block` amounts are zero. It makes sense to use something similar to the check on L239: `eq_ensure!(schedule.locked >= T::MinVestedTransfer::get(), Error::<T>::AmountLow`.

Update: This issue has been mitigated by removing the `add_extra_genesis` block containing the faulty code.

QSP-5 Ignored return values

Severity: *Medium Risk*

Status: Mitigated

File(s) affected: `pallets/eq-balances/src/lib.rs`, `pallets/eq-claim/src/lib.rs`

Description: The return values of the following function calls are ignored:

1. **[Fixed]** `<Module<T>>::deposit_creating(currency_typed, who, free, true);` on L92 of `pallets/eq-balances/src/lib.rs`
2. **[Fixed]** `Self::deposit_creating(*currency, who, d, false);` on L289 of `pallets/eq-balances/src/lib.rs`
3. **[Fixed]** `CurrencyOf::<T>::deposit_creating(&dest, initial_balance);` on L496 of `pallets/eq-claim/src/lib.rs`
4. **[Fixed]** `CurrencyOf::<T>::deposit_creating(&vesting_account_id, vs.0);` on L501 of `pallets/eq-claim/src/lib.rs`
5. **[Fixed]** `CurrencyOf::<T>::deposit_creating(&dest, balance_due);` on L509 of `pallets/eq-claim/src/lib.rs`
6. **[Mitigated]** `Self::withdraw(` on L201 of `pallets/eq-balances/src/lib.rs`.
7. **[Fixed]** `Self::withdraw(*currency, who, d, WithdrawReasons::all()` on L292 of `pallets/eq-balances/src/lib.rs` this result may be an `Err` variant, which should be handled.
8. **[Fixed]** `T::Currency::transfer(` on L324 of `pallets/eq-vesting/src/lib.rs`. Once fixed, the “can't fail” comment in line 406 might not necessarily be true.
9. **[Fixed]** `Self::deposit_creating(currency, &to, value, true);` on L179 of `pallets/eq-balances/src/lib.rs`. This function returns `PositiveImbalance::zero()` if it encounters an error.

Ignoring return values may lead to unexpected results since the execution is assuming a certain outcome of the called functions.

Recommendation: Check the return values from the call to each of the function calls enumerated above. Implement appropriate handlers for when an unexpected result is returned.

Update: Item 6 has been mitigated by disabling the `make_free_balance_be` function (where `Self::withdraw` is called) using the compiler flag `feature = "production"`. This flag is

meant to be used for production builds and end-users can verify if its value is set correctly using the `check_production()` extrinsic. All other items were fixed.

QSP-6 Insufficient clean-up may lead to incorrect state

Severity: Medium Risk

Status: Mitigated

File(s) affected: `pallets/eq-vesting/src/lib.rs`

Description: The `add_vesting_schedule()` and `remove_vesting_schedule()` functions do not clear the entry in the `Vested` hashmap. If the same Account ID receives a new vesting schedule after an old vesting schedule is removed without completing, then the new vesting schedule would not have the full amount unlocked.

Recommendation: Clear the entry in the `Vested` hashmap when `add_vesting_schedule()` or `remove_vesting_schedule()` are called.

Update: The `remove_vesting_schedule()` function was fixed by making it panic when called. Therefore, a vesting schedule cannot be removed and the description of this issue is no longer applicable.

QSP-7 Locked tokens

Severity: Medium Risk

Status: Fixed

File(s) affected: `pallets/eq-vesting/src/lib.rs`

Description: If a vesting schedule is removed using `remove_vesting_schedule()` without being completed, then the remaining tokens will be locked in the vesting module and might not be recoverable even with `force_vested_transfer()` (since it needs additional token deposit).

Recommendation: Either prevent removing vesting schedules that have not been completed or construct a way to unlock the funds after removing vesting schedules.

Update: This issue has been fixed by preventing the removal of vesting schedules.

QSP-8 Errors may occur after extrinsics write to storage

Severity: Medium Risk

Status: Mitigated

File(s) affected: `pallets/eq-balances/src/lib.rs`, `pallets/eq-claim/src/lib.rs`

Description: To avoid incorrect/unnecessary writes to storage that should be reverted/removed in case an error occurs, it is recommended to check for any potential errors before writing to storage. This rule of thumb is not followed in the following functions:

1. **[Fixed]** `currency_transfer`, where errors can occur on L589 and L594 in `pallets/eq-balances/src/lib.rs`.
2. **[Fixed]** `deposit_into_existing`, where errors can occur on L666 in `pallets/eq-balances/src/lib.rs`.
3. **[Fixed]** `deposit_creating`, where errors can occur on L698 in `pallets/eq-balances/src/lib.rs`.
4. **[Fixed]** `withdraw`, where errors can occur on L754 in `pallets/eq-balances/src/lib.rs`.
5. **[Mitigated]** `move_claim` where errors can occur on L461 in `pallets/eq-claim/src/lib.rs`
6. **[Mitigated]** `mint_claim` where errors can occur on L347 in `pallets/eq-claim/src/lib.rs`

Recommendation: Check for errors before writing to storage.

Update: The first 4 items were fixed. The last 2 items have been mitigated by disabling them using the compiler flag `feature = "production"`. This flag is meant to be used for production builds and end-users can verify if its value is set correctly using the `check_production()` extrinsic.

QSP-9 Missing input validation

Severity: Low Risk

Status: Mitigated

File(s) affected: `pallets/eq-claim/src/lib.rs`, `pallets/eq-vesting/src/lib.rs`

Description: The following functions/blocks do not check their input parameters for unexpected values:

1. **[Mitigated]** `move_claim` should check that the `old` address actually has a valid claim that can be moved to `new`. It should also check that `new` is different from the zero address. **Update:** There is no check that the `old` address actually has a valid claim. However, this function was disabled using the feature = "production" compiler flag.
2. **[Mitigated]** `mint_claim` should check that the `who` address is different from the zero address.
3. **[Fixed]** `add_extra_genesis` block in `pallets/eq-vesting/src/lib.rs` on L122 make sure no two entries refer to the same account, as then at least a vesting shall be overwritten.
4. **[Fixed]** `vested_transfer` check that `schedule.per_block` is greater than zero.
5. **[Mitigated]** `force_vested_transfer` check that `schedule.per_block` is greater than zero.
6. **[Mitigated]** `mint_claim` should check that `Claims` and `Vesting` don't contain `who`.
7. **[Mitigated]** `move_claim` should check that `Claims` and `Vesting` don't contain `new`.

Not performing proper input validation to the aforementioned parameters in the corresponding functions, could lead, e.g. to "burning" funds by transferring them to the zero address.

Recommendation: Perform input validation for each of the parameters and function combinations enumerated above.

Update: Several items have been mitigated by disabling them using the compiler flag `feature = "production"`. This flag is meant to be used for production builds and end-users can verify if its value is set correctly using the `check_production()` extrinsic.

QSP-10 Incorrect value passed when emitting event

Severity: *Low Risk*

Status: Fixed

File(s) affected: `pallets/eq-vesting/src/lib.rs`

Description: The event emitted on L337: `Self::deposit_event(RawEvent::VestingUpdated(who, to_vest));` inside the `update_lock` function should indicate (according to the comments from L154-156):

1. The vested `AccountId`
2. The amount which is left unvested (and thus locked). The second parameter is therefore incorrect, because `to_vest` indicates the amount that was transferred during this single call of `update_lock`.

Recommendation: Either the event should be redefined or the value passed to the event should be corrected.

QSP-11 Not all funds may be released due to truncation of `per_block`

Severity: *Low Risk*

Status: Fixed

File(s) affected: `pallets/eq-vesting/src/lib.rs`

Description: The integer division on L136: `let per_block = locked / length_as_balance.max(sp_runtime::traits::One::one());` truncates the resulting decimals. This would mean that the locked amount is not released in the desired amount of blocks.

Recommendation: Consider rounding up the result of the integer division to compute `per_block`.

Update: This issue has been mitigated by removing the `add_extra_genesis` block containing the faulty code.

QSP-12 Integer Overflow / Underflow

Severity: *Low Risk*

Status: Mitigated

File(s) affected: `pallets/eq-claim/src/lib.rs`, `pallets/eq-balances/src/lib.rs`, `eq-utils/src/lib.rs`

Description: Integer overflow/underflow occur when an integer hits its bit-size limit. Every integer has a set range; when that range is passed, the value loops back around. A clock is a good analogy: at 11:59, the minute hand goes to 0, not 60, because 59 is the largest possible minute. Any place in the code where primitive operations are used there should be a handler for what should happen in case of an overflow. For example:

1. **[Mitigated]** on L222 in `pallets/eq-claim/src/lib.rs` : `config.claims.iter().fold(Zero::zero(), |acc: BalanceOf<T>, &(_, b, _, _)| acc + b)`, which could cause an overflow.
2. **[Fixed]** on L323 in `pallets/eq-claim/src/lib.rs` : `<Total<T>>::mutate(|t| *t += value);`, which could cause an overflow.
3. **[Fixed]** on L36-47 in `eq-utils/src/lib.rs`, the `fixedi64_from_fixedi128` function does not check if the `raw_value` is lower than `i64::MIN`, which could cause an underflow.
4. **[Fixed]** on L784 in `pallets/eq-balances/src/lib.rs`: `SignedImbalance::Positive(PositiveImbalance::new(value + a_balance))`, which could cause an overflow.
5. **[Fixed]** The program logic in `update_group_total` seems to be based on the assumption that all balances associated with `account_id` have been recorded through the `set_usergroup()` function properly so that either of (i) `total.collateral == total cumulative positive imbalance && total.debt == 0` or (ii) `total.collateral == 0 && total.debt == total cumulative negative imbalance` would hold. However, if `update_group_total()` is called on an arbitrary `account_id` then there may be over/underflows in `total.collateral` or `total.debt`. It is recommended to handle possible over/underflows within `update_group_total()` and also check for the conditions mentioned above.

Recommendation: Implement check/handler for integer overflow/underflow. Also set `overflow-checks = true` in the `[profile.release]` section of `Cargo.toml`, otherwise no integer overflow checks are enabled during runtime and no panic will occur on overflow.

Update from dev team regarding 12.1: Normal behaviour - it should fail if there's something wrong in the genesis so the system won't start.

QSP-13 Loss of precision during conversion of balance

Severity: *Low Risk*

Status: Fixed

File(s) affected: `eq-utils/src/lib.rs`

Description: The `fixedi64_from_balance` function causes a loss in precision when converting balances into the signed 64-bit fixed-point format. This is also confirmed by the comment inside that function on L17: "need to change fixed type because of u64 -> i64 covertion problem!".

Recommendation: Remove or change the return type of this function in order to avoid the loss of precision.

Update: This issue was fixed be removing the faulty code.

QSP-14 Missing checks for dead accounts

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `pallets/eq-balances/src/lib.rs`

Description: The `currency_transfer()` and `withdraw()` functions do not check for dead accounts similar to the `deposit_into_existing()` function. This could lead to funds being lost by transferring them to a dead account.

Recommendation: Check if the account being used is dead in the aforementioned functions.

Update from dev team: Currency transfer should not check for not existing accounts, because transfer can create account. Withdraw can't be called from not existing account.

QSP-15 Privileged Roles and Ownership

Severity: *Informational*

Status: Mitigated

File(s) affected: `pallets/eq-vesting/src/lib.rs`, `pallets/eq-claim/src/lib.rs`, `pallets/eq-balances/lib.rs`

Description: Blockchain software systems will often have roles to designate the person with special privileges to make modifications to the system. We have identified the following instances of this issue in this code base:

1. **[Mitigated]** All the funds needed for vesting are transferred into the `Vesting` module itself. The `root` account could use the `force_vested_transfer` function to transfer money from the vesting pallet (`source`) to any target account. .
2. **[Mitigated]** The root role can create a vested transfer using the `force_vested_transfer` function, which takes funds from any account via a `T::Currency::transfer` call.
3. **[Mitigated]** The root role can use `force_vested_transfer` with vesting schedules that are extremely short, in order to move funds around.
4. **[Mitigated]** The root role can generate an arbitrary token claim (and an optional vesting schedule) using `mint_claim()`.

The root role of the `eq-balances` pallet can perform the following actions:

1. **[Mitigated]** Mint/deposit arbitrary amounts of any chosen currency to any account.
2. **[Mitigated]** Burn any amount from any account for any reason, which could even result in account going out of existence.

Recommendation: Privileged roles and their capabilities need to be made clear to the users via publicly available and user-facing documentation.

Update: The items marked as mitigated above have been disabled using the compiler flag `feature = "production"`. This flag is meant to be used for production builds and end-users can verify if its value is set correctly using the `check_production()` extrinsic.

QSP-16 Unused error values

Severity: *Informational*

Status: Fixed

File(s) affected: `pallets/eq-claim/src/lib.rs`, `pallets/eq-aggregates/src/lib.rs`, `pallets/eq-balances/src/lib.rs`

Description: The following error values are not used:

1. The `ValidityError::NoPermission` value of the `pub enum ValidityError` defined in `pallets/eq-claim/src/lib.rs` is never used in the code base. The comment corresponding to this value says it should be used when: "No permission to execute the call".
2. Unused errors `AlreadyAdded` and `AlreadyRemoved`. Were these intended to be part of checks in the `set_usergroup()` function? Currently these cases are treated as no-ops so either remove the error or implement with error. Also, the naming `AlreadyRemoved` is rather confusing when it refers to the case "/// User was not in this group".
3. The `ExistingVestingSchedule` error declared on L122 in the `eq-balances` pallet is never used.

Recommendation: Either use the error values or remove them.

QSP-17 Unused `enum`

Severity: *Informational*

Status: Fixed

File(s) affected: `pallets/eq-claim/src/lib.rs`

Description: The `pub enum StatementKind` declared on L91-99 in `pallets/eq-claim/src/lib.rs` and which has a `Default` implementation on L101-105, is not used anywhere in the file.

Recommendation: Clarify if this `enum` is necessary. Remove this `enum` or use it appropriately.

QSP-18 Loosely constrained type used for balances

Severity: *Informational*

Status: Fixed

File(s) affected: `eq-primitives/src/signed_balances.rs`

Description: The `SignedBalance<Balance>` implementation exclusively uses `AtLeast32Bit` for the type of the balance. However, from the code it is clear that only unsigned values are expected for the balance. Therefore, this type is not sufficiently constrained. This is also confirmed by the fact that all other pallets in the scope of this audit (i.e. `eq-aggregates`, `eq-balances` and `eq-vesting`) use `AtLeast32BitUnsigned` exclusively.

Recommendation: Align the type used in the implementation of `SignedBalance` with that used by other pallets, while at the same time properly constraining the balance values to unsigned by replacing `AtLeast32Bit` with `AtLeast32BitUnsigned`.

QSP-19 Functions that can set the balance of any account

Severity: *Informational*

Status: Fixed

File(s) affected: `pallets/eq-balances/src/lib.rs`

Description: The `set_balance_with_agg_unsafe()` and `make_free_balance_be()` should be used for test purposes only. These functions can set the balance of accounts to any value.

The `set_balance_with_agg_unsafe()` is indicated as "// for tests only! #[cfg(test)] is not visible in other pallets", but does not look like this is the case.

Recommendation: Make sure these functions are not visible/usable by any other code than the test suite code.

Update: The `set_balance_with_agg_unsafe()` function was removed and `make_free_balance_be()` was changed such that it cannot be called in production.

QSP-20 Unimplemented features

Severity: Informational

Status: Acknowledged

File(s) affected: `pallets/eq-balances/src/lib.rs`

Description: Many features are incomplete and WIP. The exact mechanics are not documented.

1. L468 `fn can_slash()`
2. L480 `fn burn()`
3. L486 `fn issue()`
4. L615-621 `fn slash()`
5. L599 Deleting accounts that fall below minimum account balance `ExistentialDeposit` after transfer

Recommendation: Document and clarify in the documentation that the aforementioned functions are not implemented.

Update from dev team: The methods mentioned in points 1-4 are needed to comply with the Substrate interface. However, they are not used anywhere on the Equilibrium chain. Therefore, calling them causes panics which is a normal behaviour since these methods are not supposed to be called in the balance pallet in our implementation. The 5th point mentioned in this issue is a problem and is related to QSP-14. It will be fixed in future releases.

QSP-21 Unused input parameter

Severity: Undetermined

Status: Fixed

File(s) affected: `pallets/eq-aggregates/src/lib.rs`

Description: The first input parameter of the `update_group_total` function is called `account_id`. However, it is never used inside the function body.

Recommendation: Clarify if this is intentional. If this is intentional, prefix it with an underscore: `_account_id`.

Update: This issue was fixed by removing the unused parameter.

QSP-22 Same account can get a new vesting schedule

Severity: Undetermined

Status: Fixed

File(s) affected: `pallets/eq-vesting/src/lib.rs`

Description: The same account can get a new vesting schedule once vesting completes as there is no check that prevents insertion into the `Vesting` and `Vested` hashmaps again. This may be intentional but the `add_vesting_schedule()` and `remove_vesting_schedule()` needs to be fixed. See the issue labelled as "Locked Tokens".

Recommendation: Clarify if this is intentional. Otherwise, there must be a check in place that prevents insertion into `Vesting` and `Vested` once again after a vesting schedule is completed.

Update from dev team: Won't fix as the implemented logic was intended. Never leads to an error after deleting `remove_vesting_schedule` method.

QSP-23 Positive zero is different from negative zero

Severity: Undetermined

Status: Fixed

File(s) affected: `eq-primitives/src/signed_balances.rs`

Description: Deriving trait `PartialEq` for enum `SignedBalance` will have `Positive(0)` considered different than `Negative(0)` inside `eq-primitives/src/signed_balances.rs`.

Recommendation: Clarify if this is intentional and if so, clarify this in the documentation. Otherwise, ensure that the comparison of `Positive(0)` is equal to `Negative(0)`.

Update from dev team: Fixed by implementing correct comparison.

QSP-24 The `Unknown` currency can be used

Severity: Undetermined

Status: Fixed

File(s) affected: `eq-primitives/src/currency.rs`

Description: The `pub enum Currency` defines `Unknown` as its first value, followed by known currency types such as EQ, ETH, BTC, etc. It is unclear if users should be able to transfer amounts of the `Unknown` currency between their accounts. However, we have found that if the transfers would be enabled, users would be able to transfer amounts of the `Unknown` currency, because there are no checks in the transfer, deposit and withdraw functions to prevent them from doing so.

Similarly, we are not sure why the default `UserGroup` and `Currency` are both set to `Unknown` as opposed to `Balances` and `Eq` respectively. These do not seem to serve any function within this repo.

Recommendation: Clarify if it is intended to use the `Unknown` currency. If not, then check that the value of the `currency` parameter is different from `Unknown`, for each function that has such a parameter.

Update: This issue was fixed by adding checks to prevent transfers, deposits and burn of the `Unknown` currency.

QSP-25 The `make_free_balance_be` function does not update aggregates

Severity: *Undetermined*

Status: Mitigated

File(s) affected: `pallets/eq-balances/src/lib.rs`

Description: All other implemented functions which change account balances inside `eq-balance` also update the `Aggregates` via the `update_total` and `set_usergroup` functions. The `make_free_balance_be` function is the only exception that does not update the `Aggregates`.

Recommendation: Clarify if this is intentional. If it is not, then call the necessary functions to update the `Aggregates`.

Update: This issue has been mitigated, by disabling the `make_free_balance_be` function in production.

QSP-26 Aggregates not updated when account is killed

Severity: *Undetermined*

Status: Fixed

File(s) affected: `pallets/eq-balances/src/lib.rs`

Description: The `on_killed_account` function does not remove `who` from `UserGroup::Balances` and update `Aggregator` total

Recommendation: Clarify if this is intentional. If not, remove `who` from `UserGroup::Balances` and update `Aggregates` total.

Update: This issue was fixed by updating the aggregates when an account is killed.

QSP-27 Incorrect imbalance type returned

Severity: *Undetermined*

Status: Acknowledged

File(s) affected: `pallets/eq-balances/src/lib.rs`, `pallets/eq-balances/src/balance_adapters.rs`

Description: According to the official documentation of Substrate for `Trait frame_support::traits::Imbalance`: "Imbalances can either be Positive (funds were added somewhere without being subtracted elsewhere - e.g. a reward) or Negative (funds deducted somewhere without an equal and opposite addition - e.g. a slash or system fee payment)." However, this is not the case for the following functions:

1. `burn` returns `PositiveImbalance` in `balance_adapter.rs` on L35 and in `lib.rs` on L356, L480.
2. `issue` returns `NegativeImbalance` in `balance_adapter.rs` on L38 and in `lib.rs` on L359, L486.

Recommendation: The official Substrate documentation says that modules should generally handle imbalances in some way. Good practice is to do so in a configurable manner using an `OnUnbalanced` type for each situation in which your module needs to handle an imbalance.

Update: The dev team has indicated that the 2 functions which return incorrect imbalance types originate from the standard Substrate code and cannot be changed without the help of the Substrate team who are in charge of updating these functions.

Automated Analyses

Rust Audit

The following 5 warnings were found. We recommend addressing the concerns indicated under the URL associated with each of the 5 warnings below:

Crate: block-cipher
Version: 0.8.0
Warning: unmaintained
Title: crate has been renamed to `cipher`
Date: 2020-10-15
ID: RUSTSEC-2020-0057
URL: https://rustsec.org/advisories/RUSTSEC-2020-0057

Crate: directories
Version: 2.0.2
Warning: unmaintained
Title: directories is unmaintained, use directoriess-next instead
Date: 2020-10-16
ID: RUSTSEC-2020-0054
URL: https://rustsec.org/advisories/RUSTSEC-2020-0054

Crate: failure
Version: 0.1.8
Warning: unmaintained
Title: failure is officially deprecated/unmaintained
Date: 2020-05-02
ID: RUSTSEC-2020-0036
URL: https://rustsec.org/advisories/RUSTSEC-2020-0036

Crate: net2
Version: 0.2.35
Warning: unmaintained
Title: `net2` crate has been deprecated; use `socket2` instead

Date: 2020-05-01
ID: RUSTSEC-2020-0016
URL: https://rustsec.org/advisories/RUSTSEC-2020-0016

Crate: stream-cipher
Version: 0.7.1
Warning: unmaintained
Title: crate has been renamed to `cipher`
Date: 2020-10-15
ID: RUSTSEC-2020-0058
URL: https://rustsec.org/advisories/RUSTSEC-2020-0058

Adherence to Specification

We have not received any formal specification for this project.

Code Documentation

The following issues were identified in the code comments:

1. Each function should have a brief description of its purpose, input parameters and returns values, if any. Several functions in the code base do not have any comments.
2. **[Fixed]** Typo on L771 in `pallets/eq-balances/src/lib.rs`: "collaterall" -> "collateral"
3. **[Fixed]** The comment "Return amount that is still locked in vesting" on L81 and L99 in `pallets/eq-vesting/lib.rs` is incorrect. The statement following this comment actually returns the amount that would be vested in total if the vesting period would be unlimited. Moreover, the statements subsequent to the statement following the comment are different in the 2 functions `locked_at` and `unlocked_at`. We recommend that the comment be revised.
4. **[Partially Fixed]** Rather than `TransfersIsDisabled` and `IsTransfersEnabled`, write `TransfersAreDisabled` and `AreTransfersEnabled` inside the `eq-vesting` and `eq-balances` pallets.
5. **[Fixed]** L224 and L265 in `pallets/eq-vesting/src/lib.rs` mention "Emits `VestingCreated`." However, there is no such event defined, nor is it emitted.
6. L233 in `pallets/eq-claim/src/lib.rs` is incorrectly specifying: "Pallet storage - the statement kind that must be signed, if any".
7. **[Fixed]** Some comments are difficult to understand as they don't consist of clear sentences. For example on L85 in `pallets/eq-balances/src/lib.rs`: "^^ begin, length, amount liquid at genesis". Such comments should be clarified.
8. **[Fixed]** There are 2 code comments on lines L598: "checks new account + can transfer" and L599: "delete account from if < min", which seem to be placed in the wrong place, because they are written just before the `Ok(())` instruction.
9. **[Fixed]** There is a comment containing several exclamation marks on L733 in `pallets/eq-balances/src/lib.rs`: "!!!!!!!! Check all balances!". It is unclear what exactly this refers to, whether it refers to the balances of all users in a group or to all the different currencies of the `who` address. However, such a check is not performed on the lines of code following this comment. Therefore, the comment should be either (re-)moved or the indicated check should be performed.
10. **[Fixed]** Typo on L342 in `pallets/eq-balances/src/lib.rs`: "Returns balance value if balance is negative or zero if negative". The last "negative" should be "positive".
11. There is no description of the `statement` parameter to the `mint_claim` function in `pallets/eq-claim/src/lib.rs`

Adherence to Best Practices

The following best practice recommendations refer to the `eq-claim` and `eq-vesting` pallets that were in scope of this security audit, as well as other pallets that are used by these 2 pallets.

1. **[Fixed]** Import with single component use path such as `use substrate_fixed;` inside `eq-utils/src/math.rs` is not necessary, and thus should be removed.
2. **[Fixed]** The `Codec` import inside `eq-primitives/src/lib.rs` do not seem to be used and can be removed.
3. **[Fixed]** The `Encode`, `Decode`, `debug`, `Deserialize`, `Serialize`, `FixedI128` and `FixedI64` imports in `pallets/eq-balances/src/lib.rs` do not seem to be used and can be removed.
4. **[Fixed]** The only reason `Copy` types implement `Clone` is for generics, not for using the `clone` method on a concrete type. Try removing the `clone` call from:
   - `self.0.clone()` on L128 and L197 in `pallets/eq-balances/src/imbalances.rs`.
   - `let a_balance = balance.clone()` on L775 and L783 in `pallets/eq-balances/src/lib.rs`. Instead, try dereferencing it: `*balance`.
   - `config.claims.iter().map(|(a, b, _, _)| (a.clone(), b.clone())).collect::<Vec<_>>()` on L217 in `pallets/eq-claim/src/lib.rs`. Instead, try dereferencing variables as: `*a` and `*b`.
   - `.filter_map(|(a, _, _, s)| Some((a.clone(), s.clone())))` in `pallets/eq-claim/src/lib.rs` on L236. Instead, try dereferencing the variables as: `*a` and `*s`.
   - `.filter_map(|(a, _, i, _)| Some((i.clone()?, a.clone())))` in `pallets/eq-claim/src/lib.rs` on L244. Instead, try dereferencing the variable as: `*a`.
5. Too complex types make the code less readable. Consider using a `type` definition to simplify the following:
   - `BTreeMap<T::AccountId, Vec<(currency::Currency, SignedBalance<T::Balance>)>>` in `pallets/eq-balances/src/lib.rs`.
   - `Vec<(T::AccountId, T::BlockNumber, T::BlockNumber, BalanceOf<T>)>` in `pallets/eq-vesting/src/lib.rs`.
   - `Vec<(EthereumAddress, BalanceOf<T>, Option<T::AccountId>, bool)>` in `pallets/eq-claim/src/lib.rs` on L249.
6. **[Fixed]** It is not always possible for the compiler to eliminate useless allocations and deallocations generated by redundant `clone()`s. It is therefore recommended to remove the call to `acc.clone()` on L263 of `pallets/eq-balances/src/lib.rs`.
7. **[Fixed]** Address and remove all TODOs from the code. Currently we have identified the following TODOs:

. L280 in `pallets/eq-balances/src/lib.rs`: "TODO: remove"

. L482 in `pallets/eq-balances/src/lib.rs`: "todo: почему unimplemented? Вроде у нас в decl_module используется withdraw для этого."

. L503 in `pallets/eq-balances/src/lib.rs`: "wtf is this? !!!!!"

. L505 in `pallets/eq-balances/src/lib.rs`: "TODO: reason debt"

8. **[Fixed]** The `core::slice::Iter`, `Decode`, `Encode`, `FullCodec`, `debug`, `decl_event`, `dispatch::DispatchResult`, `ensure`, `storage::IterableStorageMap`, `frame_system::ensure_root`, `Deserialize`, `Serialize` and `std::fmt` imports in `pallets/eq-aggregates/src/lib.rs` do not seem to be used and can be removed.

9. **[Partially Fixed]** The `?` operator is designed to allow calls that can fail to be easily chained. For example, `foo()?.bar()` or `foo(bar()?)`. Because `Err(x)?` can't be used that way (it will always return), it is more clear to write `return Err(x)`. This occurs on:

    . L382-390 in `pallets/eq-vesting/src/lib.rs`.

    . L142-144 in `pallets/eq-claim/src/lib.rs`.

10. **[Fixed]** Arguments passed to `unwrap_or` are eagerly evaluated; if you are passing the result of a function call such as on L319: `let vested = Self::vested(&who).unwrap_or(BalanceOf::<T>::zero());` of `pallets/eq-vesting/src/lib.rs`, it is recommended to use `unwrap_or_else`, which is lazily evaluated.

11. **[Fixed]** It is more idiomatic to dereference the arguments of a `==` comparison. L173 in `pallets/eq-claim/src/lib.rs` needlessly takes the reference of both operands: `&self.0[..] == &other.0[..]`.

12. **[Fixed]** As there is no conditional check on the arguments of the `.filter_map` call on L236 in `pallets/eq-claim/src/lib.rs`: `.filter_map(|(a, _, _, s)| Some((a.clone(), s.clone())))` this could be written more simply using a `.map`.

13. Using `option.map(f)` where `f` is a function or closure that returns the unit type `()`, hampers readability. This can be written more clearly with an `if let` statement. The following instances occur in `pallets/eq-claim/src/lib.rs`:

    . on L411: `Claims::<T>::take(&old).map(|c| Claims::<T>::insert(&new, c));`

    . on L412: `Vesting::<T>::take(&old).map(|c| Vesting::<T>::insert(&new, c));`

    . on L415-417: `maybe_preclaim.map(|preclaim| Preclaims::<T>::mutate(&preclaim, |maybe_o| if maybe_o.as_ref().map_or(false, |o| o == &old) { *maybe_o = Some(new) }));`

14. **[Fixed]** The user might expect to be able to use `Default` as the type can be constructed without arguments for `PrevalidateAttests<T>`. Consider adding a `Default` implementation for `PrevalidateAttests<T>`.

15. Code clones should be avoided and instead code should be reused via internal functions. For example, L77-82 is a clone of L95-100 in `pallets/eq-vesting/src/lib.rs`.

16. **[Fixed]** The usage of unrestricted imports is discouraged save for prelude modules: `pallets/eq-balances/src/imbalances.rs`, L3.

17. **[Fixed]** The requirements `From<u64>` and `Into<u64>`, when `Balance` is defined in `eq-balances` it could result in loss of precision either in the `Into<u64>` conversion or in the `From<u64>` conversion if the balances had a type different than `u64` in the runtime. It would help at least explaining that the bit-width of the `Balance` type will be 64 bits. An assertion would be best. This reduces confusion and mitigates the possibility of truncation, loss of precision or overflow.

18. To reduce code duplication, consider:

    . Writing a function to validate an input `VestingInfo`;

    . Refactoring code in `vested_transfer` and `force_vested_transfer`.

19. There is no need for the storage maps `Vesting` and `Vested` to map to `Option`s.

20. On L313-325 in `eq-balances/src/lib.rs` the `can_change_balance()` function is re-implemented for Tuples when most of the code in this pallet seem to be working with the `can_change_balance()` from `runtime/src/lib.rs`. Renaming the function would reduce confusion.

21. `iter_account()` and `iter_total()` return different types despite seemingly similar naming. Use more descriptive names, provide comments on usages, or use the same return type.

22. **[Fixed]** The condition of the `if`-statement on L86 of `pallets/eq-aggregate/src/lib.rs` does not depend on the value of the `for`-loop iterator on L85. Therefore, as an optimization we recommend splitting this loop into 2 `for`-loops that do not have an `if`-statement inside. These `for`-loops can be placed in the `if` and `else` branches of the statements from L80-84.

23. **[Fixed]** The only reason `Copy` types implement `Clone` is for generics, not for using the `clone` method on a concrete type. Try removing the `clone` call from:

    . `currency.clone()` on L607 in `pallets/eq-balances/src/lib.rs`.

24. **[Fixed]** The `deposit_creating` function in `pallets/eq-balances/src/lib.rs`. returns `PositiveImbalance::zero()` if it encounters an error. It is recommended to offer a more explicit error message to help the end-user understand what the cause of a deposit failure was.

25. **[Fixed]** Commented code should be removed on L732 in `pallets/eq-balances/src/lib.rs`.

26. Consider removing `mut` from L588 and adding `let` to L593 in `pallets/eq-balances/src/lib.rs`.

Test Results

Test Suite Results

All tests are passing

```
     Running target/debug/deps/eq_aggregates-6b3b7161bc9f9e95

running 20 tests
test tests::iter_total_empty ... ok
test tests::iter_account_empty ... ok
test tests::in_usergroup_false ... ok
test tests::in_usergroup_true ... ok
test tests::get_total_empty ... ok
test tests::empty_aggregates ... ok
test tests::iter_account_success ... ok
test tests::update_group_total_negative_delta_negative_prev ... ok
test tests::iter_total_success ... ok
test tests::get_total_success ... ok
test tests::update_group_total_positive_delta_positive_prev ... ok
test tests::update_group_total_positive_delta_negative_prev ... ok
test tests::update_group_total_zero_balance ... ok
test tests::update_group_total_negative_delta_positive_prev ... ok
test tests::deposit_withdraw_success ... ok
test tests::on_killed_account_debt ... ok
test tests::set_usergroup_add_to_group ... ok
test tests::set_usergroup_remove_from_group ... ok
test tests::on_killed_account ... ok
test tests::transfer_success ... ok

test result: ok. 20 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out

     Running target/debug/deps/eq_balances-8e75340c92b90b44

running 17 tests
test tests::test_transfer_enabling_disabling ... ok
test tests::enable_disable_transfers ... ok
test tests::burn ... ok
test tests::free_balance ... ok
test tests::make_free_balance ... ok
test tests::currency_total_issuance ... ok
test tests::on_killed_account ... ok
test tests::test_ensure_can_withdraw_and_withdraw ... ok
test tests::unknown_currency ... ok
test tests::deposit ... ok
test tests::overflow ... ok
test tests::no_balances ... ok
test tests::balance_checker_not_allow ... ok
test tests::get_balances ... ok
test tests::test_deposit ... ok
test tests::test_aggregates_balances ... ok
test tests::zero_deposit ... ok

test result: ok. 17 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out

     Running target/debug/deps/eq_claim-69dc542672f6ef13

running 29 tests
test mock::ecdsa_sig_as_ref ... ok
test mock::ecdsa_sig_comparison ... ok
test mock::err_conversion_to_u8 ... ok
test mock::invalid_attest_transactions_are_recognised ... ok
test mock::attesting_works ... ok
test mock::attest_moving_works ... ok
test mock::basic_setup_works ... ok
test mock::cannot_bypass_attest_claiming ... ok
test mock::prevalidate_attest_default ... ok
test mock::claiming_works ... ok
test mock::claim_cannot_clobber_preclaim ... ok
test mock::serde_works ... ok
test mock::claiming_while_vested_doesnt_work ... ok
test mock::claim_attest_moving_works ... ok
test mock::mint_claim_correct_error ... ok
test mock::origin_signed_claiming_fail ... ok
test mock::double_claiming_doesnt_work ... ok
test mock::non_claimant_doesnt_work ... ok
test mock::valid_attest_transactions_are_free ... ok
test mock::add_claim_works ... ok
test mock::real_eth_sig_works ... ok
test mock::add_claim_with_vesting_works ... ok
test mock::real_eth_sig_works_ci ... ok
test mock::non_sender_sig_doesnt_work ... ok
test mock::basic_claim_moving_works ... ok
test mock::add_claim_with_statement_works ... ok
test mock::attest_claiming_works ... ok
test mock::claiming_does_not_bypass_signing ... ok
test mock::validate_unsigned_works ... ok

test result: ok. 29 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out

     Running target/debug/deps/eq_distribution-2c02f6089ab95c73

running 0 tests

test result: ok. 0 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out

     Running target/debug/deps/eq_integration_testing-d2936b8c506d9f0a

running 10 tests
test ethkey::tests::test_canonical_seed ... ok
test key::tests::test_canonical_seed ... ok
test key::tests::test_random_account_key ... ok
test ethkey::tests::test_random_string ... ok
test key::tests::test_random_array_string ... ok
test ethkey::tests::test_incorrect_seed ... ok
test ethkey::tests::test_random_key ... ok
test ethkey::tests::test_signature ... ok
test ethkey::tests::test_verify ... ok
test ethkey::tests::test_address ... ok

test result: ok. 10 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out

     Running target/debug/deps/eq_bridge_testing-dd8a432bd835fab9

running 0 tests

test result: ok. 0 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out

     Running target/debug/deps/eq_integration_testing-50a2bc43f9631434

running 0 tests

test result: ok. 0 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out

     Running target/debug/deps/eq_integration_testing_macro-cf1b93204082f054

running 0 tests

test result: ok. 0 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out

     Running target/debug/deps/eq_node-394da0167f700a41

running 0 tests

test result: ok. 0 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out

     Running target/debug/deps/eq_node_runtime-ecdaa782fa3ca275

running 1 test
test __construct_runtime_integrity_test::runtime_integrity_tests ... ok

test result: ok. 1 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out

     Running target/debug/deps/eq_primitives-5a1c4e2b917869d0

running 0 tests

test result: ok. 0 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out

     Running target/debug/deps/eq_session_manager-c746aa2bb40b6ff8

running 15 tests
test tests::validators_stay_unchanged ... ok
test tests::initial_validators ... ok
test tests::first_session_no_validators ... ok
test tests::session_unregistered_validator_added ... ok
test tests::session_nonexistent_validator_removed ... ok
test tests::remove_validators ... ok
test tests::second_session_validators_from_config ... ok
test tests::add_validators ... ok
test tests::several_sessions ... ok
test tests::session_no_validator_changes ... ok
test tests::session_existing_validator_added ... ok
test tests::session_validator_removed ... ok
test tests::session_validators_added ... ok
test tests::session_change_flag ... ok
test tests::session_several_sessions ... ok

test result: ok. 15 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out

     Running target/debug/deps/eq_utils-0b8a76669299017c
```

```
running 20 tests
test math::inner_fixed_conversions ... ok
test math::ln_fails ... ok
test math::exp_test ... ok
test math::ln_test_e ... ok
test math::pow_complex_result ... ok
test math::pow_test_0_pow_0 ... ok
test math::pow_test_0_base ... ok
test math::pow_test_e ... ok
test math::pow_test ... ok
test math::ln_test ... ok
test math::sqrt_fails ... ok
test math::pow_test_pow_1 ... ok
test math::pow_test_pow_0 ... ok
test math::pow_test_neg_power ... ok
test test::test::test_fx64_no_trailing_zeros ... ok
test test::test::test_fx64_trailing_zeros ... ok
test math::sqrt_for_integers ... ok
test test::test::test_to_prec ... ok
test math::sqrt_test_small_num ... ok
test math::sqrt_for_floats ... ok

test result: ok. 20 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out

     Running target/debug/deps/eq_vesting-10c36d69b8e59ca2

running 19 tests
test tests::add_zero_vesting_schedule ... ok
test tests::add_vesting_schedule ... ok
test tests::vest_no_vesting ... ok
test tests::account_getter ... ok
test tests::transfers_disabled ... ok
test tests::forced_transfers_disabled ... ok
test tests::vest_before_start ... ok
test tests::vested_transfer_already_exists ... ok
test tests::vested_transfer_amount_low ... ok
test tests::zero_vested_transfer ... ok
test tests::vest_all_init ... ok
test tests::overflow_vesting ... ok
test tests::vest_other ... ok
test tests::forced_vested_transfer ... ok
test tests::vest_temp ... ok
test tests::vested_transfer_ok ... ok
test tests::next_vesting ... ok
test tests::vest_temp2 ... ok
test tests::vest_all ... ok

test result: ok. 19 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out
```

Code Coverage

Due to a known issue of the `Tarpaulin` tool, we were not able to determine the degree of test coverage. Therefore, we do not provide any coverage information generated by `Tarpaulin` at the time of writing.

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

```
b950e266f0078704b87e170dfa36de8d57aecffe67171aca3bc3df5e023f1fd1 ./equilibrium-substrate-chain/vesting.rs
48d694941767d65c0e938097cdce8c035257e9d82be2ec25748962c387521bb5 ./equilibrium-substrate-chain/balances.rs
460832aa5e5896f495661a909f1b6eea34720890b6de80c2238ebf1ce8c5d3bb ./equilibrium-substrate-chain/claim.rs
7524e4188c7fff02100caf0261ce806b814ae69c11e7fa569a50dd6d7279f48c ./equilibrium-substrate-chain/node/build.rs
7ab904cc7195535878c9fb757d093d7fd092c44e74418ef5378d54d3ece53076 ./equilibrium-substrate-chain/node/src/chain_spec.rs
35abdfce3a9a24ae0503f98acd5e85040ed03f2b038b9e65620403b88690d448 ./equilibrium-substrate-chain/node/src/command.rs
97318ffa72ad70a8b165234c6b5ac88052ecb163722f98f4c5a05545c167b5a8 ./equilibrium-substrate-chain/node/src/service.rs
0b280dd9837730a55db61b8d6388b934b387d2474f70154776b2f5aeb4f38e6b ./equilibrium-substrate-chain/node/src/main.rs
ea81af31d890ec3785cb0901b847041797ede9cdd7ef7354a6ec6990c4e1640f ./equilibrium-substrate-chain/node/src/cli.rs
2ef34cd31b1fb717860a2089d8bdf17a080362614a66b5199e5b5b64a72d8cbd ./equilibrium-substrate-chain/node/src/rpc.rs
8b255d3870da5d0a4b7c68f2d96e563d2b12057f099b37dbce9512d3c7f3aa58 ./equilibrium-substrate-chain/eq-primitives/src/lib.rs
1ce2b0d194ef81442e799e424e01df94adf70aa8c2852bef51311413f7d558ff ./equilibrium-substrate-chain/eq-primitives/src/signed_balance.rs
f45097a275616063a55f368bd12237a36860bca656f420417399a6bfcbdd3909 ./equilibrium-substrate-chain/eq-primitives/src/currency.rs
b9b1fea76ba6e5a6b559e29108b4958faa077abeb350dd0316c209a08c7d40d9 ./equilibrium-substrate-chain/eq-utils/src/test.rs
9bf7b377e52a6a162afb330ecdb5b5cb0494013908a60828d00f82dd195ef6b8 ./equilibrium-substrate-chain/eq-utils/src/ensure.rs
b93a0c1cea5648ee42d7cb153c19edced0fdf4548c6222409e9421902d292379 ./equilibrium-substrate-chain/eq-utils/src/log.rs
4793df661c1e0b514f73fccb433dd0aa5714cab3c9eae286351054be2367efb0 ./equilibrium-substrate-chain/eq-utils/src/lib.rs
a10d59f06b2a894d13ebca1c30f6be41b6e7ada763a3812967d62ccb35827d71 ./equilibrium-substrate-chain/eq-utils/src/option.rs
e20d4583997ad5ea880f5e858f2a00e9e1b951076b79dde888d40f154740a1f2 ./equilibrium-substrate-chain/eq-utils/src/math.rs
e397a69c67df5014ce3ddced39c82e19905332887f99560a06ae2be4ac09bf62 ./equilibrium-substrate-chain/pallets/eq-distribution/src/lib.rs
985fdabb21e52cb57c2467a8097be5ff57a6236e1c5a72edc6f9ff286465ba4b ./equilibrium-substrate-chain/pallets/eq-balances/src/balance_adapter.rs
d00b27ff499b1e9ccb465b088bc78a3d468fcd2471ab9f800e2e84c319da7763 ./equilibrium-substrate-chain/pallets/eq-balances/src/lib.rs
1a06a2b1974ed62d3b1c8dc52b7b99ca333025ccd6ec66e97d5fb4ecc68a4daf ./equilibrium-substrate-chain/pallets/eq-balances/src/benchmarks.rs
4418f78c6e13bab651c9e2fcabd55f2cb64fee1e7ae95dab7a2e5f63139124d1 ./equilibrium-substrate-chain/pallets/eq-balances/src/benchmarking.rs
2c409c7ea03968f166b0b3c2971b48affc40c0398bab4ab96e1b85f39154e6bf ./equilibrium-substrate-chain/pallets/eq-balances/src/mock.rs
58b76511067c90fa98d4e766389279adce2a19685d518d3a603c4df700b7d2a2 ./equilibrium-substrate-chain/pallets/eq-balances/src/imbalances.rs
8ec11098010a02d7df826461616f3208700b7990f35c18fe886e36484f67e3db ./equilibrium-substrate-chain/pallets/eq-balances/src/tests.rs
3916a66996286c8d116561b7549fbbffdcb841f18974661a835bf14e3dd580bf ./equilibrium-substrate-chain/pallets/eq-claim/src/lib.rs
467ba814298ac6ae8d6817b1df12241298b8059ff60c6bebdc9b379b3c959d80 ./equilibrium-substrate-chain/pallets/eq-claim/src/benchmarks.rs
d957a4a97edeb6966a69b8aa146070c42565303767a7442efc9304ddb650ac0b ./equilibrium-substrate-chain/pallets/eq-claim/src/benchmarking.rs
97e7a12eea51d09c7ebc8f0acc91a11f7e1b60a656f9f2cba3b14eb7cdf723ec ./equilibrium-substrate-chain/pallets/eq-claim/src/mock.rs
a855e496e1ef08543a7586630239c3e30b16d54dde2157b20cb0222dd58111d1 ./equilibrium-substrate-chain/pallets/eq-claim/src/secp_utils.rs
62b70a9248aaee43b200d6f580dbe5859daded806feb5d3aca68ba37dfdaa9be ./equilibrium-substrate-chain/pallets/eq-vesting/src/lib.rs
5550cd1547fef92e79f2fae410c115222ec56aaf3c7426b292f6a4d56b34c6f2 ./equilibrium-substrate-chain/pallets/eq-vesting/src/benchmarks.rs
5abdf956939bec8838c1bab7c707afa42d957af096fc4abfe42fb147220d0ee1 ./equilibrium-substrate-chain/pallets/eq-vesting/src/benchmarking.rs
1f2515fca932c9ec44818e68c98428e7bfa05d8b4e5ff12bd82e04f9ce1cf284 ./equilibrium-substrate-chain/pallets/eq-vesting/src/mock.rs
0781ff4568d6025d957b940e0342e8a496b9fc2613c41a7288e73f28d46a051b ./equilibrium-substrate-chain/pallets/eq-vesting/src/tests.rs
5fef03155908e5f4a8f70b3858c7d96c02da016fc0aef73061b80f1ff4f7dde9 ./equilibrium-substrate-chain/pallets/eq-session-manager/src/lib.rs
ab50c00b726611d46afefa1d4f46f9ae7bea1019dcd5bf9120dca5f75a4c3ee4 ./equilibrium-substrate-chain/pallets/eq-session-manager/src/mock.rs
bbae1f56187fb3e477e65fa2ea255321d6f65b2ea74762b39fd4825c505f1c54 ./equilibrium-substrate-chain/pallets/eq-session-manager/src/tests.rs
2a9a561f3d614820a00b95c8846162ec46de132d221d6e9530153cf3f6bedd60 ./equilibrium-substrate-chain/pallets/eq-aggregates/src/lib.rs
e77bf72d9fa2092531163b95d3fddeb6f38190d4a801c9f7d9d2f3748f621e27 ./equilibrium-substrate-chain/pallets/eq-aggregates/src/mock.rs
1ebca654889da6ee07d70eb363adb603ac8f20961215113bce2df8a0a0bd6b8b ./equilibrium-substrate-chain/pallets/eq-aggregates/src/tests.rs
e6f9dfb3ee36127a43364b379f77db2c76a78f95343af47d31dd91fca41c67b5 ./equilibrium-substrate-chain/runtime/build.rs
df3f20e21a85b11c90bcd33564fa7d08eba69819d9b72ac2e460cd54d0ca2b4a ./equilibrium-substrate-chain/runtime/src/lib.rs
```

Changelog

- 2020-11-09 - Initial report for `eq-claims` and `eq-vesting` based on commit `1381045`
- 2020-11-11 - Added findings for `eq-aggregates` and `eq-balances` based on commit `1381045`
- 2020-11-12 - Reaudit based on commit `9ab3c587`
- 2020-11-30 - Reaudit based on commit `9dfcacf`
- 2020-12-04 - Reaudit based on commit `27ce403`

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.