DePool Contract Verification (Phase 2)

Short Description

Contract proofs in Coq according to the specifications produced in Phase 1

Motivation

Contest submission should prove the correctness of DePool contract using Coq according to the below requirements by translating the contract code into a Coq eDSL (embedded Domain Specific Language), provide and prove Eval/Exec provide formal Scenarios description for all public functions. Issue reviews, fixes and corrections, with their subsequent reproving.

Term

Each participant should provide the following:

Translation to Coq eDSL

1. Types and abstractions
2. Contract Class to be realized during the next steps

Coq Eval/Exec specification for all public functions

1. Error/Value equivalence conditions, e.g. for each function the *iff* conditions are to be formulated when the function acts normally or throws an exception (with number)
2. Eval: full return values on all branches (including exceptions throwing)
3. Exec: full state modification on all branches (in terms of contract members, including branches with exception throwing)

4. Non public functions analyzing in terms if they need to be specified, or just included to correspondent public (as inlines, without additional proofs)

Proof of Eval/Exec specification

Issues review and corrections if needed, respecification and proofs

Formal specification of scenarios in terms of the already “proven functions” and state members

1. Invariants (global, local etc). Here we need invariants hierarchy as well.
    1. Having Inv (S) we get Inv (S*), where S → S*
    2. Subclasses of invariants: value invariants (the concrete computation over the state is constant, including members themselves, as projection-like computation), predicate invariants (the concrete proposition over the state is held)
2. Direct scenarios (functional) formulation: what we get doing smth
3. Inverse scenarios (security) formulation: what we had done if we got smth

Review, fixes, corrections, respecification, reproving

The answers should be provided in the following form:

- Code equivalently mapped to the formally verifiable language (with clear indication of equivalence degree)
- Standalone/separated list of statements to be proven
- Requirements explicitly mentioned in the “Coq Eval/Exec specification for all public functions”
- Code that proves the specification and verifiable by an external trusted proof checker (such as Coq, Agda, Isabelle, Idris, K)
- The list of externally required specifications must be moved to a separate file that will allow to make a quick checkup.
- Internal specifications and proofs must be provided and fully pass build chain
- All the axioms (not proved assumptions) must also be moved to a separate file

- Check if all the requirements mentioned above are met and provide the detailed assessment
- List of notes, accepted and fixed corrections, list of rejected corrections with justification

Contest Dates:

26 October 2020 — 26 November 2020

Proposed prices:

1 place — 350,000

2 place — 250,000

3 place — 150,000

The jury:

Jury members who vote in this contest must be known experts in the field of security, smart contract audit, and formal verification.

- Jurors whose team(s) intend to participate in this contest by providing submissions lose their right to vote in this contest.
- Each juror will vote by rating each submission on a scale of 0 to 10 or can choose to reject it if it does not meet requirements, or they can choose to abstain from voting if they feel unqualified to judge.
- Jurors will provide feedback on your submissions.
- Duplicate, modifications of another submission, sub par, incomplete or inappropriate submissions will be rejected.

Jury rewards:

An amount equal to 5% of the sum total of all total tokens actually awarded to winners of this contest will be divided equally between all jurors who vote and provide feedback. Both voting and feedback are mandatory in order to collect this reward.

Procedural requirements

- Because of the very specific nature of this contest only 1 submission per team is acceptable. Submissions should not be a modified version of another submission. Jury should take special care in reviewing submissions in that respect.
- All submissions must be accessible for the jury to open and view, so please double-check your submission. If the submission is inaccessible or does not fit the criteria described, the submission may be rejected by jurors.
- Contestants must submit their work before the closing of the filing of submissions. If not submitted on time, the submission will not count.
- All submissions must contain the contestant’s contact information, preferably a Telegram ID by which jurors can verify that the submission belongs to the individual who submitted it. **If not, your submission may be rejected.**
- If your submission has links to the work performed, the content of those links must have the contestant's contact details, preferably a Telegram ID so jurors can match it and verify who the work belongs to. **If not, your submission may be rejected.**

Disclaimer

Anyone can participate, but Free TON cannot distribute tokens to US citizens or US entities.