HACKEN

# RENEC SECURITY ANALYSIS

## Intro

| | |
|---|---|
| **Name** | Renec |
| **Website** | https://renec.foundation/ |
| **Repository** | https://github.com/renec-chain/renec |
| **Commit** | 48757023ffc8d5e5534695831c5c2b3636b9bf19 |
| **Platform** | L1 |
| **Network** | Renec |
| **Languages** | Rust |
| **Methods** | Automated Code analysis, Manual review, Issues simulation |
| **Auditor** | s.akermoun@hacken.io |
| **Auditor** | n.lipartiia@hacken.io |
| **Approver** | l.ciattaglia@hacken.io |
| **Timeline** | 24.02.2023 - 23.03.2023 |
| **Changelog** | 24.03.2023 (Preliminary Report) |
| **Changelog** | 21.04.2023 (Final Report) |

# Table of contents

## Summary

Renec is a fork derived from the latest stable release of Solana, version 1.13.6.
It features a unique inflation model that deviates from its upstream counterpart, implementing a consistent 4.5% inflation rate annually. This approach aims to provide a stable and predictable inflationary environment for the Renec ecosystem, fostering sustainable growth and development over time.
As a Solana fork, Renec benefits from its parent project's robust and scalable architecture while introducing its distinct monetary policy. The Renec project also incorporates a custom genesis configuration that outlines the distribution of tokens among various stakeholders, including miners, Remitano, liquidity providers, marketing, and the treasury. This configuration employs different unlock schedules to ensure a balanced release of tokens over time, ranging from immediate unlock to a gradual release over several years. These measures contribute to a well-structured token distribution system, further supporting the long-term growth and stability of the Renec ecosystem.

## Documentation quality

The source code for the project is thoroughly documented, with well-structured comments and explanations that elucidate the functions, classes, and variables used throughout the codebase.
This level of detail ensures that developers and reviewers can quickly comprehend the logic and intent of the code, facilitating efficient maintenance and updates.
As such, there are no noteworthy concerns or suggestions regarding the quality of the source code documentation.

The total Documentation Quality score is **10** out of 10.

## Code quality

The Rust code implemented in the project demonstrates a high level of quality, adhering to best practices and industry standards.
However, there are two minor areas of concern.
Firstly, the current fork introduces a break in one of the unit tests inherited from the upstream repository, which should be addressed to ensure complete test coverage.
Secondly, Cargo.toml metadata files contain inaccurate or incomplete values, necessitating review and correction to guarantee proper configuration and package management.
Aside from these two issues, the overall code quality is commendable and does not warrant any additional concerns or recommendations.

The total Code Quality score is **9** out of 10.

## Architecture quality

Renec maintains the same high level of architectural quality as the upstream repository, ensuring consistency and adherence to established design patterns and blockchain principles.
The fork introduces a few modifications, including a new genesis configuration and an updated inflation system, both of which have been implemented effectively and demonstrate solid design quality. This alignment with the original architecture allows for seamless integration and compatibility with the existing Solana ecosystem, while preserving the robustness and scalability of the project.
Consequently, there are no significant concerns or recommendations regarding the architecture quality of this fork.

The architecture quality score is **10** out of 10.

## Security score

The fork has maintained the overall security level of the upstream repository, ensuring that the implemented changes do not introduce new vulnerabilities or weaken the existing security measures.
However, it has been observed that some vulnerable dependencies have not been patched, despite updates being available in the upstream repository. To maintain the highest possible security standards, it is crucial for the project team to continuously monitor the

upstream repository and promptly integrate any security upgrades, patches, or improvements.
This proactive approach will help mitigate potential risks and safeguard the project against emerging threats.

The security score is **9** out of 10.

## Total score

Considering all metrics, the total score of the report is **9.2** out of 10.

## Findings count and definitions

| Severity | Findings | Severity Definition |
|---|---|---|
| **Critical** | 0 | Critical vulnerabilities are usually straightforward to exploit and can lead to the loss of user funds or contract state manipulation by external or internal actors. |
| **High** | 1 | High vulnerabilities are usually harder to exploit, requiring specific conditions, or have a more limited scope, but can still lead to the loss of user funds or contract state manipulation by external or internal actors. |
| **Medium** | 0 | Medium vulnerabilities are usually limited to state manipulations but cannot lead to asset loss. Major deviations from best practices are also in this category. |
| **Low** | 1 | Low vulnerabilities are related to outdated and unused code or minor Gas optimization. These issues won't have a significant impact on code execution but affect code quality. |
| **Total** | 2 | |

# Scope of the audit

## Protocol Audit

### Changes review

- Review of all changes in sources since fork from Solana 1.13.6 (~1k changed lines)
- Review of all security, bug and segfault related issues reported in Solana since version 1.13.6 (~50)

### Chain

- Review of changes in genesis configuration and token supply configuration

## Implementation

### Code Quality

- Static Code Analysis
- Tests coverage

## Protocol Tests

### Node Tests

- Environment Setup
- Transactions & Consensus tests

# Issues

## Vulnerable dependencies in Renec blockchain

Renec blockchain uses dependencies with publicly known vulnerabilities.

| | |
|---|---|
| **ID** | REN-005 |
| **Scope** | Code Security |
| **Severity** | HIGH |
| **Status** | Fixed |

### Details

The Renec blockchain node uses the following dependencies with known vulnerabilities.
All the dependencies listed are inherited from Solana node.

| Dependency | Version | Id | Description | Recommendation (potential breaking changes) | Upstream patch |
|---|---|---|---|---|---|
| **bzip2** | 0.4.3 | RUSTSEC-2023-0004 | bzip2 Denial of Service (DoS) | Upgrade to >=0.4.4 | #30180 |
| **chrono** | 0.4.19 | RUSTSEC-2020-0159 | Potential segfault in \`localtime_r\` invocations | Upgrade to >=0.4.20 | No |
| **openssl-src** | 111.22.0+1.1.1q | RUSTSEC-2023-0007 | Timing Oracle in RSA Decryption | Upgrade to >=111.25, <300.0 OR >=300.0.12 | #30180 |
| **openssl-src** | 111.22.0+1.1.1q | RUSTSEC-2023-0006 | X.400 address type confusion in X.509 \`GeneralName\` | Upgrade to >=111.25, <300.0 OR >=300.0.12 | #30180 |
| **openssl-src** | 111.22.0+1.1.1q | RUSTSEC-2023-0009 | Use-after-free following \`BIO_new_NDEF\` | Upgrade to >=111.25, <300.0 OR >=300.0.12 | #30180 |
| **openssl-src** | 111.22.0+1.1.1q | RUSTSEC-2023-0010 | Double free after calling \`PEM_read_bio_ex\` | Upgrade to >=111.25, <300.0 OR >=300.0.12 | #30180 |
| **remove_dir_all** | 0.5.3 | RUSTSEC-2023-0018 | Race Condition Enabling Link Following and Time-of-check Time-of-use (TOCTOU) | Upgrade to >=0.8.0 | #30633 |
| **rocksdb** | 0.18.0 | RUSTSEC-2022-0046 | Out-of-bounds read when opening multiple column families with TTL | Upgrade to >=0.19.0 | Yes in v1.14 |
| **time** | 0.1.43 | RUSTSEC- | Potential segfault in the | Upgrade to >=0.2.23 | No |

| | | | | | |
|---|---|---|---|---|---|
| | | 2020-0071 | time crate | | |
| **tokio** | 1.14.1 | RUSTSEC-2023-0001 | reject_remote_clients Configuration corruption | Upgrade to >=1.18.4, <1.19.0 OR >=1.20.3, <1.21.0 OR >=1.23.1 | #29587 |

An attacker can exploit a known vulnerability in the Renec node and performs a denial-of-service attack on the network by taking down all nodes in the network.

### Recommendation

Actually upstream already fixed some issues in `v1.13` and is addressing other issues in `v1.14` branch, and fix will be provided in the next stable release.
Short term, update all dependencies to their newest version for `v1.13` if a fix is available, and wait for next stable release for applying patches to the remaining issues.
Long term, run `cargo-audit` as part of the CI/CD pipeline and ensure that the team is alerted to any vulnerable dependencies that are detected.

## Unmaintained and yanked dependencies in Renec blockchain

Renec blockchain uses dependencies which are unmaintained or yanked.

| | |
|---|---|
| **ID** | REN-006 |
| **Scope** | Code Security |
| **Severity** | LOW |
| **Status** | Acknowledged (Dev team will monitor upstream repo for next stable releases) |

### Details

The Renec blockchain node uses the following dependencies which are no more maintained or yanked by authors.
All the dependencies listed are inherited from Solana node.

| Dependency | Version | Id | Status | Remediation | Upstream testnet patch |
|---|---|---|---|---|---|
| **ansi_term** | **0.11.0** | RUSTSEC-2021-0139 | **unmaintained** | **Use alternative crates: anstyle, console, nu-ansi-term, owo-colors, stylish, yansi** | **No** |
| **net2** | **0.2.37** | RUSTSEC-2020-0016 | **unmaintained** | **\`net2\` crate has been deprecated; use \`socket2\` instead** | **No** |
| **serde_cbor** | **0.11.2** | RUSTSEC-2021-0127 | **unmaintained** | **Use alternative crates: ciborium, minicbor** | **Yes** |
| **stdweb** | **0.4.20** | RUSTSEC-2020-0056 | **unmaintained** | **Use alternative crates: wasm-bindgen, js-sys, web-sys** | **Yes** |
| **block-buffer** | **0.10.0** | N/A | **yanked** | **upgrade** | **Yes** |
| **cpufeatures** | **0.2.1** | N/A | **yanked** | **upgrade** | **No** |
| **crossbeam-channel** | **0.5.3** | N/A | **yanked** | **upgrade** | **Yes** |

| | | | | | |
|---|---|---|---|---|---|
| **crossbeam-utils** | **0.8.5** | N/A | **yanked** | **upgrade** | **Yes** |
| **quinn-udp** | **0.1.0** | N/A | **yanked** | **upgrade** | **No** |
| **zeroize_derive** | **1.2.0** | N/A | **yanked** | **upgrade** | **No** |

### Recommendation

Fixing these issues may result in breaking changes, as some dependencies are replaced by alternatives.
As most of these issues are fixed on testnet upstream version `v1.14`, we recommend to monitor upstream updates and apply updates accordingly.

## Cargo.toml manifest files contain wrong metadata

Crates within the workspace have a misconfigured manifest file.

| | |
|---|---|
| **ID** | REN-001 |
| **Scope** | Code Quality, Project Structure |
| **Status** | Fixed |

### Details

`Cargo.toml` manifest files contain wrong metadata for `repository` field in the `[package]` section.
When crates are published on crates.io, and so on docs.rs, the `repository` field will show the source code location of the crate.
All source code location defined in `repository` field are pointing to old `https://github.com/remitano/renec` repository or `https://github.com/solana-labs/solana` instead of `https://github.com/renec-chain/renec` which can mislead developers and contributors. Some modified crates from fork, with Renec related code are also still pointing to `https://github.com/solana-labs/solana` in their `repository` field. Even without publishing a crate publicly, source code location in `repository` field should be consistent with the actual location of the code.

### Recommendation

Update `authors` and `repository` field of the `[package]` section in `Cargo.toml` file for all crates with correct values.
With a rust toolchain **>= 1.64.0** workspace inheritance can be used to avoid duplication of common field values between crates. for example at workspace level you can define in `Cargo.toml`:

```
[workspace.package]
authors = ["RENEC Maintainers <dev@remitano.com>"]
edition = "2021"
version = "1.13.6"
repository = "https://github.com/renec-chain/renec"
homepage = "https://remitano.com/"
license = "Apache-2.0"
```

And in crates you can inherit the package section defined at workspace level. For example for `renec-genesis` crate:

```
[package]
name = "renec-genesis"
description = "Blockchain, Rebuilt for Scale"
documentation = "https://docs.rs/renec-genesis"
version = { workspace = true }
authors = { workspace = true }
repository = { workspace = true }
homepage = { workspace = true }
```

```
license = { workspace = true }
edition = { workspace = true }
```

## Confusing variable name

Confusing type assumption based on variable name.

| | |
|---|---|
| **ID** | REN-002 |
| **Scope** | Code Quality |
| **Status** | Fixed |

### Details

in *rbpf-cli/src/main.rs* we can find:

```
    let max_u64 = std::i64::MAX.to_string();
```

While reading code, a reader could infer that `max_u64` is the maximum value of an unsigned 64 bits integer, but it is actually the string representation of a signed 64 bits integer maximum value.

### Recommendation

Don't use this temporarily variable and pass directly the needed value to the list of valid arguments for the `SOLANA BPF CLI`.

```
    .arg(
        Arg::new("instruction limit")
            .help("Limit the number of instructions to execute")
            .short('l')
            .long("limit")
            .takes_value(true)
            .value_name("COUNT")
            .default_value(&std::i64::MAX.to_string()),
    )
```

## Crate stake-monitor is empty and not included in workspace

| | |
|---|---|
| **ID** | REN-008 |
| **Scope** | Code Quality |
| **Status** | Fixed |

### Details

`stake-monitor` is an empty crate with just a `Cargo.toml` file inside.
`stake-monitor` is not included as a workspace member.

### Recommendation

Delete `stake-monitor` directory. (#18020)

## Inflation unit test failed

Change in Inflation behavior breaks a unit test.

| | |
|---|---|
| **ID** | REN-004 |
| **Scope** | Code Quality |
| **Status** | Fixed |

### Details

A Change of inflation behavior and rates breaks a unit test in *sdk/src/inflation.rs*:

```
$ cargo test -p solana-sdk test_inflation
test inflation::tests::test_inflation_fixed ... ok
test inflation::tests::test_inflation_basic ... FAILED
failures:
---- inflation::tests::test_inflation_basic stdout ----
thread 'inflation::tests::test_inflation_basic' panicked at 'assertion failed: total < last', sdk/src/inflation.rs:127::
note: run with `RUST_BACKTRACE=1` environment variable to display a backtrace
failures:
    inflation::tests::test_inflation_basic
test result: FAILED. 1 passed; 1 failed; 0 ignored; 0 measured; 198 filtered out; finished in 0.00s
error: test failed, to rerun pass `-p solana-sdk --lib`
```

Inflation is defined with these constant values:

```
// Initial inflation percentage, from time=0
const DEFAULT_INITIAL: f64 = 0.045;
// Terminal inflation percentage, to time=INF
const DEFAULT_TERMINAL: f64 = 0.045;
// Rate per year, at which inflation is lowered until reaching terminal
const DEFAULT_TAPER: f64 = 0.0;
// Percentage of total inflation allocated to the foundation
const DEFAULT_FOUNDATION: f64 = 0.05;
// Duration of foundation pool inflation, in years
const DEFAULT_FOUNDATION_TERM: f64 = 7.0;
```

Since inflation remains consistently high at 4.5% over years, the assertion `assert!(total < last)` will always fail.

```
#[test]
#[allow(clippy::float_cmp)]
fn test_inflation_basic() {
    let inflation = Inflation::default();
    let mut last = inflation.total(0.0);
    for year in &[0.1, 0.5, 1.0, DEFAULT_FOUNDATION_TERM, 100.0] {
        let total = inflation.total(*year);
        assert_eq!(
            total,
            inflation.validator(*year) + inflation.foundation(*year)
        );
        assert!(total < last); // THIS WILL ALWAYS FAIL
        assert!(total >= inflation.terminal);
        last = total;
    }
    assert_eq!(last, inflation.terminal);
}
```

### Recommendation

Since the inflation rate remains constant over years, assertions should check equality:

```
#[test]
#[allow(clippy::float_cmp)]
fn test_inflation_basic() {
    let inflation = Inflation::default();
    let mut last = inflation.total(0.0);
    for year in &[0.1, 0.5, 1.0, DEFAULT_FOUNDATION_TERM, 100.0] {
        let total = inflation.total(*year);
        assert_eq!(
            total,
            inflation.validator(*year) + inflation.foundation(*year)
        );
        assert_eq!(total, last);
        assert_eq!(total, inflation.terminal);
        last = total;
    }
    assert_eq!(last, inflation.terminal);
}
```

## Relax linter for undefined behavior

Warning on usage of uninitialized data.

| | |
|---|---|
| **ID** | REN-003 |
| **Scope** | Code Quality |
| **Status** | Fixed |

### Details

A warning is trigger by the compiler for `Default` implementation of `Packet` struct in *sdk/src/packet.rs*:

```
warning: the type `[u8; 1232]` does not permit being left uninitialized
   --> sdk/src/packet.rs:122:30
    |
122 |             buffer: unsafe { std::mem::MaybeUninit::uninit().assume_init() },
    |                              ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    |                              |
    |                              this code causes undefined behavior when executed
    |                              help: use `MaybeUninit<T>` instead, and only call `assume_init` after initialization
    |
    = note: integers must be initialized
    = note: `#[warn(invalid_value)]` on by default
```

### Recommendation

To avoid this warning, the buffer initialization should be done in 2 steps:

```
impl Default for Packet {
    fn default() -> Self {
        let buffer = std::mem::MaybeUninit::<[u8; PACKET_DATA_SIZE]>::uninit();
        Self {
            buffer: unsafe { buffer.assume_init() },
            meta: Meta::default(),
        }
    }
}
```

It is good practice to eliminate all compiler warnings for release builds.

## Unmerged pull requests from Upstream to Renec

There are fixes released on upstream v1.13 & v1.14 that could be applied to Renec v1.13.6 and future versions.

| | |
|---|---|
| **ID** | REN-007 |
| **Scope** | Informational |

### details

| Description | Version | Scope | status | Pull Request | Recommendation |
|---|---|---|---|---|---|
| ci: resolve outstanding v1.13 cargo-audit errors | v1.13 | Security | Merged | #30180 | Apply |
| stops nodes from broadcasting slots twice | v1.13 | Performance | Merged | #30684 | Apply (if applied, also merge 30717) |
| revert unintentional debug change introduced by d373e87 | v1.13 | Bug | Merged | #30717 | Apply if 30684 is merged. As it solved a bug introduced by this pull request. |
| Improves RPC path sanitation (backport #29931) | v1.14 | Security/Bug | Merged | #29946 | Wait for next stable release and merge |
| Panic when shred index exceeds the max per slot (backport of #30555) | v1.14 | Bug | Merged | #30605 | Wait for next stable release and merge |
| Exit when stuck in an unrecoverable repair/purge loop (backport of #28596) | v1.14 | Bug | Merged | #30562 | Wait for next stable release and merge |
| validators always skip clean/shrink on startup (backport of #30710) | v1.14 | Performance | Open | #30714 | Wait for next stable release and merge |

### Recommendation

Non breaking v1.13 pull requests should be merged.
Upstream updates should be monitored for future releases and merged pull requests.

# Disclaimers

## Hacken disclaimer

## Technical disclaimer