Cryptonics

Audit Report

Dock PoS Audit Report

June 23, 2021

Table of Contents

Disclaimer

THE CONTENT OF THIS AUDIT REPORT IS PROVIDED “AS IS”, WITHOUT REPRESENTATIONS AND WARRANTIES OF ANY KIND.

THE AUTHOR AND HIS EMPLOYER DISCLAIM ANY LIABILITY FOR DAMAGE ARISING OUT OF, OR IN CONNECTION WITH, THIS AUDIT REPORT.

COPYRIGHT OF THIS REPORT REMAINS WITH THE AUTHOR.

This audit has been performed by

Philip Stanislaus and **Stefan Beyer**

Cryptonics Consulting S.L.
Ramiro de Maeztu 7
46022 Valencia
SPAIN

https://cryptonics.consulting/
info@cryptonics.consulting

Introduction

Purpose of this Report

Cryptonics Consulting has been engaged to perform a security audit of the Dock PoS consensus implementation (https://www.dock.io/)

The objectives of the audit are as follows:

1. Determine the correct functioning of the system, in accordance with the project specification.
2. Determine possible vulnerabilities, which could be exploited by an attacker.
3. Determine smart contract bugs, which might lead to unexpected behavior.
4. Analyze whether best practices have been applied during development.
5. Make recommendations to improve code safety and readability.

This report represents a summary of the findings.

As with any code audit, there is a limit to which vulnerabilities can be found, and unexpected execution paths may still be possible. The author of this report does not guarantee complete coverage (see disclaimer).

Codebase Submitted for the Audit

The audit has been performed on the following GitHub repository:

https://github.com/docknetwork/dock-substrate/tree/pos-stake

Commit hash: `ed9a27c6aca4ffe1a9c5740ece0005766605ba79`

Methodology

The audit has been performed in the following steps:

1. Gaining an understanding of the code base's intended purpose by reading the available documentation.
2. Automated source code and dependency analysis.
3. Manual line by line analysis of the source code for security vulnerabilities and use of best practice guidelines, including but not limited to:
    a. Race condition analysis
    b. Under- / overflow issues
    c. Key management vulnerabilities
4. Report preparation

Functionality Overview

The submitted source code implements a substrate chain for the Dock platform. This particular release changes the consensus mechanism from PoA (Aura + Grandpa) to PoS (Babe + Grandpa).

How to read this Report

This report classifies the issues found into the following severity categories:

Severity	Description
Critical	A serious and exploitable vulnerability that can lead to loss of funds, unrecoverable locked funds, or catastrophic denial of service.
Major	A vulnerability or bug that can affect the correct functioning of the system, lead to incorrect states or denial of service.
Minor	A violation of common best practices or incorrect usage of primitives, which may not currently have a major impact on security, but may do so in the future or introduce inefficiencies.
Informational	Comments and recommendations of design decisions or potential optimizations, that are not relevant to security. Their application may improve aspects, such as user experience or readability, but is not strictly necessary. This category may also include opinionated recommendations that the project team might not share.

The status of an issue can be one of the following: **Pending,** **Acknowledged,** or **Resolved**. Informational notes do not have a status, since we consider them optional recommendations.

Note, that audits are an important step to improve the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of the system should exercise caution. To help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentatio**n, and **test coverage**. We include a table with these criteria for each module, in the corresponding findings section.

Note, that high complexity or lower test coverage does not necessarily equate to a higher risk, although certain bugs are more easily detected in unit testing than a security audit and vice versa.

Summary of Findings

No	Description	Severity	Status
1	Unlocking of validator rewards could halt block production	Critical	Pending
2	Network may be attacked due to high initial validator count	Major	Pending
3	Council can change own members, technical collective members cannot be changed	Major	Resolved
4	Reporting of equivocations through Grandpa API disabled	Major	Resolved
5	Sudo pallet is used and account is set on mainnet	Minor	Acknowledged
6	Weights for most calls do not account for computation	Minor	Acknowledged
7	Weight info not set for some pallets	Minor	Resolved
8	Mainnet chain spec contains invulnerables	Minor	Acknowledged
9	Emission reward for era is capped to one year	Minor	Resolved
10	Unlocked rewards may be lost	Minor	Resolved
11	PoA pallet contains deprecated code	Informational	Acknowledged
12	Transaction payment is not dynamic	Informational	Resolved
13	Generation of key ownership proofs through Grandpa API disabled	Informational	Resolved
14	Source of randomness can be improved	Informational	Resolved

Code Quality Criteria

Criteria	Status	Comment
Code complexity	Medium	-
Code readability and clarity	High	-
Level of Documentation	High	-
Test Coverage	Medium-high	-

Detailed Findings

1. Unlocking of validator rewards could halt block production

Severity: Critical

In the `on_initialize` handler in `pallets/poa/src/lib.rs:458` that contains logic to be run at the beginning of every new block, unlocking of validator rewards is triggered. That logic contains an unbounded loop in `pallets/poa/src/lib.rs:1311`, which is used to find the next epoch with a positive reward emission for validators. The loop contains storage reads, which are relatively expensive. Depending on the amount of iterations the loop needs, it could theoretically not finish within the time a block has to be built, halting block production.

Recommendation

We recommend changing the unlocking logic to be run on user request as opposed to in every block. Alternatively, the loop could be bounded to a predefined number of iterations, and iteration could be continued in the next block if necessary. That would spread unlocking across multiple blocks.

Status: Pending

2. Network may be attacked due to high initial validator count

Severity: Major

In `node/src/chain_spec.rs:617`, a validator count of 50 is set. This could lead to a security issue if the actual number of validators is much lower than 50. In such a situation, an attacker could cheaply enter the validator set and bring grandpa finalization to a halt, potentially pricking the whole chain.

Recommendation

We recommend adding at least 50 initial validators to the chain spec to not leave any room for attacks of the newly launched PoS chain. If that is not possible, we recommend to reduce the initial validator count to an attainable number and increase it later through governance.

Status: Pending

3. Council can change own members, technical collective members cannot be changed

Severity: Major

In Substrate, the council can change the membership of the technical collective/committee through the membership pallet. In `runtime/src/lib.rs:856` and `857`, that functionality

has been changed such that the council can change the membership of itself, but not of the technical collective. This undermines the democracy within the Dock network and prevents the technical collective to be kept up to date.

Recommendation

We recommend changing the `MembershipInitialized` and `MembershipChanged` types in `runtime/src/lib.rs:856` and `857` back to `TechnicalCommittee`.

Status: Resolved

4. Reporting of equivocations through Grandpa API disabled

Severity: Major

The function body of `submit_report_equivocation_unsigned_extrinsic` in `runtime/src/lib.rs:1493` is empty, which implies that equivocation reports submitted through the Grandpa API will be silently discarded without returning an error.

Recommendation

We recommend using Substrate's implementations of that function.

Status: Resolved

5. Sudo pallet is used and account is set on mainnet

Severity: Minor

In `runtime/src/lib.rs:1271` the sudo pallet is enabled and in `node/chain_spec.rs:591`, a sudo account is set. Documentation states that sudo will be removed on mainnet.

Recommendation

We recommend updating the documentation or removing the sudo account and pallet from mainnet configuration and builds.

Status: Acknowledged

The sudo pallet is intentionally set on mainnet as a contingency to any potential issues coming up. The Dock team plans to remove it a few weeks after the PoS network launch.

6. Weights for most calls do not account for computation

Severity: Minor

The weights for calls cover storage reads and writes in all cases, but most weights do not account for time spent running the calls.

There is a risk of some calls being overly cheap, which opens the possibility for an attacker to flood the network with transactions. When that happens, normal transactions might no longer be able to be processed.

Recommendation

We recommend running benchmarks on worst case scenarios for all calls and adding weights determined through benchmarking to the existing weights of calls. We also recommend running benchmarks for the weights of signature verification on the same hardware and updating the values in `pallets/core_mods/src/did.rs:138` to `143`.

Status: Acknowledged

The Dock team plans to fix this issue in the future.

7. Weight info not set for some pallets

Severity: Minor

In `runtime/src/lib.rs`, the `SystemWeightInfo` of `system` as well as the `WeightInfo` of `balances`, `utility`, `pallet_collective` and `pallet_scheduler` and pallets is set to the unit type (empty tuple) `()`.

Recommendation

We recommend instead to set types to `frame_system::weights::SubstrateWeight<Runtime>`, `pallet_balances::weights::SubstrateWeight<Runtime>`, `pallet_utility::weights::SubstrateWeight<Runtime>`, `pallet_collective::weights::SubstrateWeight<Runtime>` and `pallet_scheduler::weights::SubstrateWeight<Runtime>`, respectively.

Status: Resolved

8. Mainnet chain spec contains invulnerables

Severity: Minor

In `node/chain_spec.rs:751`, the initial authorities are set as invulnerables, which excludes them from slashing and removal from the active validator set. If an attacker can get

access to validator nodes that are in the set of invulnerables, they can double sign and attack the network that way without consequences, potentially pricking the network.

Recommendation

We recommend removing all validators from invulnerables on mainnet to increase the security of the chain.

Status: Acknowledged

Invulnerables are intentionally set on mainnet as a contingency to any potential issues coming up. The Dock team plans to remove them a few weeks after the PoS network launch.

9. Emission reward for era is capped to one year

Severity: Minor

In `pallets/staking_rewards/src/lib.rs:188`, the emission reward will be calculated as a fraction of yearly rewards. The calculation uses Perbill, which internally caps the fraction to 1, implying that the maximum era duration is one year. This will lead to wrong calculations if an era is longer than one year.

Recommendation

While an era duration of more than a year will likely never be set in practice, we still recommend adding a validation on the era duration that limits it to one year.

Status: Resolved

10. Unlocked rewards are incorrectly tracked

Severity: Minor

In `pallets/poa/src/lib.rs:1376`, unlocked rewards will be added to the existing unlocked rewards Option `stats.unlocked_reward` with the map function. However, if the `Option` is `None`, the unlocked rewards will not be stored, resulting in no tracking of unlocked rewards.

Recommendation

We recommend handling the case where unlocked rewards are `None`.

Status: Resolved

11. PoA pallet contains deprecated code

Severity: Informational

The PoA pallet in `pallets/poa/src/lib.rs` contains calls and implementations of the `ShouldEndSession` and `SessionManager` traits that are no longer used with the transition to PoS.

Recommendation

While not a security concern, we recommend to remove those calls to reduce complexity.

Status: Acknowledged

12. Transaction payment is not dynamic

Severity: Informational

Substrate uses a dynamic fee adjustment based on the block fullness to create a fee market. Dock does not implement that logic in `runtime/src/lib.rs:657`, which might lead to congestion of the blockchain.

Recommendation

We recommend using the dynamic fee adjustment from Substrate by setting `type FeeMultiplierUpdate = TargetedFeeAdjustment<Self, TargetBlockFullness, AdjustmentVariable, MinimumMultiplier>`.

Status: Resolved

13. Generation of key ownership proofs through Grandpa API disabled

Severity: Informational

The function `generate_key_ownership_proof` of the Grandpa API in `runtime/src/lib.rs:1503` does not return a proof.

Recommendation

We recommend using Substrate's implementations of that function.

Status: Resolved

14. Source of randomness can be improved

Severity: Informational

In `runtime/src/lib.rs:1398`, `RandomnessCollectiveFlip` is used as the source of randomness. Substrate is using the slightly stronger `RandomnessFromOneEpochAgo`.

Recommendation

Even though randomness is currently not used in the Dock codebase, it could be relevant to the EVM. We therefore recommend using `RandomnessFromOneEpochAgo` instead of `RandomnessCollectiveFlip`.

Status: Resolved