Cryptonics

Dock Token Migration

Audit Report - December 15, 2020

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

https://cryptonics.consulting/

Table of Contents

Disclaimer

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

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

THIS AUDIT REPORT IS NOT A SECURITY WARRANTY, INVESTMENT ADVICE, OR AN ENDORSEMENT OF THE CLIENT OR ITS PRODUCTS. THIS AUDIT DOES NOT PROVIDE A SECURITY OR CORRECTNESS GUARANTEE OF THE AUDITED SOFTWARE.

Introduction

Purpose of this Report

Cryptonics Consulting has been engaged to perform a security audit of the Dock token migration system (https://www.dock.io/). The engagement includes the backend and frontend of the token migration portal and the Substrate module used for issuing the new token.

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 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.

Codebase Submitted for the Audit

The audit has been performed on the code submitted in the following GitHub repositories:

https://github.com/docknetwork/dock-substrate/tree/migration-bonus/pallets/token_migration

commit number: `e7b6931c64a476b41894a0f912b9c6e7c1224510`

update/fixes: `4da432a73e613f6b9c619b61040f277dcf26b168`

https://github.com/docknetwork/token-migration/tree/v2

commit number: `c82e63862e929fb6e2451edaf1e0a3571a602b71`

update/fixes: `3a25e3299b11ee73170874584b7a4c4cc2f26446`

https://github.com/docknetwork/polkadot-fe/tree/master/packages/page-token-swap

commit number: `15688e967005237531657e6e14b2f6aebcc56019`

update/fixes: `1ba4952ede4792dd6d950a92eaff15bba6f4b760`

Methodology

The audit has been performed by a total of three auditors. The following steps were performed:

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
    d. Permissioning issues
    e. Logic errors
4. Report preparation

The results were then discussed between the auditors in a consensus meeting and integrated into this joint report.

Functionality Overview

The submitted code implements a token migration platform that allows users to obtain tokens on the dock substrate implementation for their existing ERC-20 token holdings. The system consists of the following components:

- **Backend:** An Node.js API connecting to the deployed ERC-20 contract
- **Front-end:** End-user frontend for token swaps
- **Substrate pallet:** Substrate token issuance model

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. In order 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.

Findings

Summary of Findings

No	Description	Severity	Module	Status
1	Potential integer over-/underflows, division by zero	Major	Substrate	Resolved
2	Migrators have free autonomy to spend/transfer funds	Minor	Substrate	Acknowledged
3	Migrators pay no fees, even in failed cases	Minor	Substrate	Acknowledged
4	Weights on some calls are incorrect	Minor	Substrate	Resolved
5	No protection of multiple migration of/bonuses to the same account	Informational	Substrate	-
6	Weights for claim bonus calls do not account for computation	Informational	Substrate	-
7	Migrations and bonus payments emit a transfer event	Informational	Substrate	-
8	Recipients can use locked funds before they are vested	Informational	Substrate	Resolved
9	Potential unsafe use of CORS wildcard	Minor	Backend	Resolved
10	Unused import statements	Informational	Frontend	-
11	Unused variable assignments	Informational	Frontend	-
12	Potentially unsafe variable shadowing	Informational	Frontend	-

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

Detailed Findings

Substrate Pallet

1. Potential integer over-/underflows, division by zero

Severity: Major

In multiple places in the pallet, over-/underflows of integers are not checked and in one instance, a division by zero could occur:

1. `bonus.total_locked_bonus += amount;` in the `add_swap_bonus` function (pallets/token_migration/src/lib.rs#L406)
2. `bonus_to_unlock += bonuses[i].0;` in the `unlock_swap_bonus` function (pallets/token_migration/src/lib.rs#L437)
3. `bonus.total_locked_bonus -= bonus_to_unlock;` in the `unlock_swap_bonus` function (pallets/token_migration/src/lib.rs#L446)
4. `bonus.total_locked_bonus += amount;` in the `add_vesting_bonus` function (pallets/token_migration/src/lib.rs#L463)
5. `let milestone_duration = vesting_duration / vesting_milestones;` in the `unlock_vesting_bonus` function (pallets/token_migration/src/lib.rs#L489)
6. `bonus_to_unlock += locked_bonus;` in the `unlock_vesting_bonus` function (pallets/token_migration/src/lib.rs#L506)
7. `let bonus_per_milestone = total_bonus / milestone_as_bal;` in the `unlock_vesting_bonus` function (pallets/token_migration/src/lib.rs#L509)
8. `T::BlockNumberToBalance::convert(milestones_passed) * bonus_per_milestone;` in the `unlock_vesting_bonus` function (pallets/token_migration/src/lib.rs#L511)
9. `bonus.total_locked_bonus -= bonus_to_unlock;` in the `unlock_vesting_bonus` function (pallets/token_migration/src/lib.rs#L531)

Recommendation

We recommend checking for over-/underflow and division by zero manually or using `checked_add`, `checked_sub`, `checked_mul` or `checked_div`.

Status: Resolved

2. Migrators have free autonomy to spend/transfer funds

Severity: Minor

In the current implementation, migrators are receiving and holding funds until they are used for migration or as bonuses. This means they could (deliberately) misuse those funds.

Recommendation

There are different ways how migrators can still perform their tasks without access to the actual funds. For instance, migrators could not hold the funds, but the runtime could instead mint the tokens or transfer them from a treasury/dock controlled account. Alternatively, the funds could be transferred to migrators, but reserved. The runtime would then unreserve them whenever a recipient receives funds/bonuses.

Status: Acknowledged

Team Response: We assume that migrators will be bound by legal agreement to not do that and prove that they have the tokens (and token holders) that they are demanding for us.

3. Migrators pay no fees, even in failed cases

Severity: Minor

As a design choice, migrators do not pay transaction fees. A consequence of that design is that a DOS attack by migrators is possible, which is documented in comments. For such an attack, a migrator would just call `migrate` or `give_bonuses` with parameters that will fail, consuming network resources without a cost. An example would be a `migrate` call with a list of recipients that has a total value which is higher than the migrators balance. Even if all migrators are honest, this could be problematic in case of a bug in the migrator's code.

Recommendation

We recommend changing the logic to make the migrator pay for failed `migrate` and `give_bonuses` calls. This approach has also been applied to root calls, e. g. `add_migrator`, where successful calls require no fee payments.

Status: Acknowledged

Team Response: "We assume the migrators are not malicious and trying to spam the network as mentioned in the pallet's docs. But your point about a migrator's buggy code is valid. But as we will be the only migrator, we are skipping this point in the interest of time."

4. Weights on some calls are incorrect

Severity: Minor

Weights on some calls are incorrect, which causes mispriced calls and hence could be exploited by attackers:

1. `claim_bonus_for_other` has currently `T::DbWeight::get().reads_writes(3, 3)` but should have `T::DbWeight::get().reads_writes(2, 2)`

2. `claim_swap_bonus` has currently
   `T::DbWeight::get().reads_writes(2, 2)` but should have
   `T::DbWeight::get().reads_writes(1, 1)`
3. `claim_swap_bonus_for_other` has currently
   `T::DbWeight::get().reads_writes(3, 3)` but should have
   `T::DbWeight::get().reads_writes(2, 2)`
4. `claim_vesting_bonus` has currently
   `T::DbWeight::get().reads_writes(2, 2)` but should have
   `T::DbWeight::get().reads_writes(1, 1)`
5. `claim_vesting_bonus_for_other` has currently
   `T::DbWeight::get().reads_writes(3, 3)` but should have
   `T::DbWeight::get().reads_writes(2, 2)`

Recommendation

We recommend adjusting the calls as detailed above. Additionally, benchmarking and adding some weight for the computational complexity is recommended. Lastly, it would be useful for maintenance if the reasoning behind the weights was explained in comments.

Status: Resolved

5. No protection of multiple migration of/bonuses to the same account

Severity: Informational

The `token_migration` pallet does currently not protect against migrators migrating tokens or giving any bonuses multiple times to the same users. While trusting migrators not to be byzantine is a deliberate design choice of the system, this allows migrators to drain funds onto their personal accounts up to their balance. While Substrate comes with built-in replay protection for extrinsics, there is no replay protection on the application layer for migration and bonuses.

Recommendation

Consider storing which recipients were migrated and which ones received bonuses to prevent multiple migrations. To prevent bloating the storage with this information, a staking/bonding mechanism could be used to incentivize correct behaviour: Any double/additional spend could lead to slashing.

Team Response: “By design, we allow migrating to the same mainnet account multiple times, one example is an ERC-20 holder consolidating its funds on mainnet.”

6. Weights for claim bonus calls do not account for computation

Severity: Informational

The weights of the calls `claim_bonus`, `claim_bonus_for_other`, `claim_swap_bonus`, `claim_swap_bonus_for_other`, `claim_vesting_bonus`, `claim_vesting_bonus_for_other` do only account for db read and writes, but not for computational complexity. A migrator could insert many bonuses for a recipient (up to the allowed migrations) which comes with no cost for migrators. Then a recipient could claim those bonuses without paying for the many iterations in the functions above. This issue will probably not lead to any problems in practice, since allowed migrations is a `u16` which bounds the maximum migrations to `65,535`.

Recommendation

Consider adding a weight component for the number of bonuses a user has to claim bonus calls.

Update: Fixed

7. Migrations and bonus payments emit a transfer event

Severity: Informational

Both `migrate` and `give_bonuses` calls internally use the `Currency` trait's `transfer` method, which is implemented by the `balances` pallet. That pallet has implemented a `Transfer` event, which will be emitted with the sender, receiver and value of the transfer. While this is not a security concern, tracing tokens through block explorers such as polkascan.io or subscan.io will show migrators as previous owners of such tokens. That might be confusing to the users.

Recommendation

We recommend changing the logic to either just mint the tokens to the recipient without the migrator even controlling those tokens beforehand, or use the `withdraw` and `mint` functions instead of the `transfer` one. Alternatively, a transfer from a predefined treasury that locks all the tokens might be used to prevent multiple migrators showing up as the previous owners.

8. Recipients can use locked funds before they are vested

Severity: Informational

In Substrate, locked balances can be locked for other purposes, e. g. for voting in the democracy pallet or for staking in the staking pallet. Likewise, not vested funds can be lost.

Recommendation

If the economic ownership of vesting bonuses should not be transferred to the recipient before those tokens are vested, consider using `reserve`/`unreserve` instead of locks.

Update: Fixed

Backend

9. Potential unsafe use of CORS wildcard

Severity: Minor

The file index.js implements Cross-Origin Resource Sharing with a wildcard header value.

Recommendation

Consider enforcing the Same-Origin Policy, or using a trusted third-party library for source validation to properly and safely implement CORS.

Status: Resolved

Frontend

10. Unused import statements

Severity: Informational

It is generally not advised to include more dependencies in a project than are actually being used. There are several unused import statements throughout the front-end that should be removed if there are no plans to use them.

Files impacted:

- packages/page-explorer/src/Account.tsx
    - Link, formatNumber
- packages/page-explorer/src/DBMain.tsx
    - KeyedEvent, HeaderExtended, registry, getSub, Query, ApolloProvider
- packages/page-explorer/src/DID.tsx
    - Link, AddressMini, formatNumber, FormatBalance
- packages/page-explorer/src/Epoch.tsx
    - AddressMini
- packages/page-explorer/src/EpochDetails.tsx
    - KeyedEvent, HeaderExtended, Columar, Column, registry, BlockHeaders, Events, Transfers, getSub, getQuery, blockToPolkadotBlock, eventToPolkadot, accountToPolkadot, ApolloProvider, Query, asDockAddress
- packages/page-explorer/src/GraphQL/AccountsList.tsx
    - getSub, ApolloProvider, Query
- packages/page-explorer/src/GraphQL/BlocksList.tsx
    - getQuery, ApolloProvider, Query
- packages/page-explorer/src/GraphQL/EpochsList.tsx
    - HeaderExtended, getSub, getQuery, accountToPolkadot, ApolloProvider, Query
- packages/page-explorer/src/GraphQL/EventsList.tsx
    - getQuery, blockToPolkadotBlock, ApolloProvider, Query
- packages/page-explorer/src/GraphQL/ExtrinsicsList.tsx

    - getQuery, ApolloProvider, Query
- packages/page-explorer/src/GraphQL/MasterMembersList.tsx
    - u8aToString
- packages/page-explorer/src/GraphQL/TransfersList.tsx
    - getSub, ApolloProvider, Query
- packages/page-explorer/src/GraphQL/ValidatorsList.tsx
    - getSub, ApolloProvider, Query
- packages/page-explorer/src/Summary.tsx
    - SummarySession
- packages/page-master-proposals/src/Create.tsx
    - isFunction, isHex, stringToHex, stringToU8a, u8aToHex, Button, InputAddress, BalanceFree
- packages/page-master-proposals/src/Selection.tsx
    - TxButton, isHex, BalanceFree, getTypeDef
- packages/page-master-proposals/src/SignatureDIDs.tsx
    - Password
- packages/page-token-swap/src/Status.tsx
    - useCallback, useEffect, InputAddress, Password, Extrinsic, TxButton
- packages/page-token-swap/src/Swap.tsx
    - useCallback, Password, Extrinsic, TxButton
- packages/page-token-swap/src/index.tsx
    - useMemo

Recommendation

Consider removing unused dependencies.

Team Response: “This is a fork of polkadot.js so we would like to minimize the changes we do here but will address files related to migration.”

11. Unused variable assignments

Severity: Informational

Much in the same way that it is inadvisable to keep unused dependencies in a project, it is also inadvisable to store values in in-scope variables but never use them.

Files impacted:

- packages/page-explorer/src/DBMain.tsx
    - lines 87, 88, 89, 90
    - error, eventsError, transfersError, richAccountsError
- packages/page-explorer/src/GraphQL/AccountsList.tsx
    - lines 23, 26
    - api, accountsError
- packages/page-explorer/src/GraphQL/BlocksList.tsx
    - line 24
    - error, loading
- packages/page-explorer/src/GraphQL/EpochsList.tsx
    - lines 79, 82
    - totalRewards, setTotalRewards, headers
- packages/page-explorer/src/GraphQL/EventsList.tsx
    - line 25
    - eventsError
- packages/page-explorer/src/GraphQL/ExtrinsicsList.tsx
    - line 27
    - eventsError
- packages/page-explorer/src/GraphQL/TransfersList.tsx
    - line 30
    - transfersError
- packages/page-explorer/src/GraphQL/ValidatorsList.tsx

    - lines 29, 32
    - api, accountsError
- packages/page-master-proposals/src/Create.tsx
    - line 18
    - accountId, setAccountId
- packages/page-master-proposals/src/Selection.tsx
    - lines 50, 232
    - accountId, setAccountId, meta, method, section
- packages/page-master-proposals/src/SignatureDIDs.tsx
    - lines 217, 224
    - pairCount, apiDefaultTxSudo
- packages/page-token-swap/src/Status.tsx
    - lines 30, 32
    - signature, setSignature, submitting
- packages/page-token-swap/src/Swap.tsx
    - line 32
    - setCurrentPair
- packages/page-token-swap/src/index.tsx
    - line 11
    - t

Recommendation

Consider removing unused variable assignments.

Team Response: “This is a fork of polkadot.js so we would like to minimize the changes we do here but will address files related to migration.”

12. Potentially unsafe variable shadowing

Severity: Informational

Reusing variables in subscopes, otherwise known as variable shadowing, is considered unsafe and could allow developers to modify variables and states that they do not intend to.

Impacted files:

- packages/apps/src/initSettings.ts
    - line 29
    - apiUrl
- packages/page-accounts/src/Accounts/modals/Create.tsx
    - lines 165, 198, 217, 222,
    - keyPassword, newSeed, name, password, isPasswordValid
- packages/page-explorer/src/DBMain.tsx
    - line 105
    - richAccounts
- packages/page-explorer/src/GraphQL/AccountsList.tsx
    - line 38
    - accounts
- packages/page-explorer/src/GraphQL/BlocksList.tsx
    - line 28
    - blocks
- packages/page-explorer/src/GraphQL/EpochsList.tsx
    - lines 87, 100
    - maxIndex, maxPages
- packages/page-explorer/src/GraphQL/ValidatorsList.tsx
    - line 42
    - accounts
- packages/page-explorer/src/Summary.tsx

    - line 26
    - currentEpoch
- packages/page-master-proposals/src/Create.tsx
    - line 30
    - error
- packages/page-master-proposals/src/Selection.tsx
    - lines 72, 73, 81, 84, 85, 115, 116, 134, 166, 189, 207, 212, 223
    - proposalJSON, proposal (73,115,189), meta, isInjected, isUsable, roundNo, accountId, signature, method, error
- packages/page-master-proposals/src/SignatureDIDs.tsx
    - line 286
    - extrinsic
- packages/page-token-swap/src/Status.tsx
    - line 53
    - status
- packages/react-api/src/Api.tsx
    - lines 72, 116, 172
    - api (72, 116), store (116,172)
- packages/react-components/src/Extrinsic.tsx
    - lines 66, 73
    - isValid, method
- packages/react-components/src/InputExtrinsic/index.tsx
    - line 53
    - optionsMethod
- packages/react-components/src/InputRpc/index.tsx
    - line 61
    - optionsMethod

Recommendation

Instead of reusing a variable name from an outer scope, give the variable in the subscope a distinctive name to avoid developer confusion. The name in the outer scope should also be descriptive.