Audit Report

Suberra

v1.0

March 17, 2022

Table of Contents

License

Disclaimer

This audit has been performed by

Oak Security

https://oaksecurity.io/
info@oaksecurity.io

Introduction

Purpose of This Report

Oak Security has been engaged by Suberra Labs Pte. Ltd to perform a security audit of the Suberra payments infrastructure smart contracts.

The objectives of the audit are as follows:

1. Determine the correct functioning of the protocol, 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/suberra/suberra-contracts

Commit hash: `23f3a404a3c5fe312bd17744aaa299d712bb24be`

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

Suberra is a web 3.0 payment application that enables users to make both recurring and activity based payments using stablecoins. The application is implemented through a number of smart contracts that include the Suberra wallet and factory, job and product registration, and implementations of token streams, and recurring and peer-to-peer payments.

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**.

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 documentation**, and **test coverage**. We include a table with these criteria below.

Note that high complexity or low 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	Subwallet factory updates could lead to inconsistent state	Major	Resolved
2	Product factory updates could lead to inconsistent state	Major	Resolved
3	Protocol fee decimal places incorrectly specified	Major	Resolved
4	Owner cannot freeze admins	Minor	Resolved
5	Owner cannot execute set permissions	Minor	Resolved
6	Admin whitelist cannot be unfrozen	Minor	Resolved
7	Protocol fee decimal places should be defined as constants	Minor	Resolved
8	Unbounded number of contract admins	Informational	Resolved
9	Total number of subscriptions increments on cancellation undo	Informational	Resolved
10	Unnecessary variable assignment	Informational	Resolved
11	Wallet permission logic function unused	Informational	Acknowledged
12	Overflow checks not enabled for release profile	Informational	Resolved

Code Quality Criteria

Criteria	Status	Comment
Code complexity	Medium	-
Code readability and clarity	Medium-High	-
Level of documentation	Medium-High	-
Test coverage	Medium-High	-

Detailed Findings

1. Subwallet factory updates could lead to inconsistent state

Severity: Major

During the creation of a subwallet, the addresses of the Anchor Money Market and Terra token contracts are passed from the subwallet factory config to the newly instantiated wallet - see `contracts/subwallet-factory/src/contract.rs:107`. These addresses are then stored in the config of the new subwallet.

The subwallet factory contract owner is able to update the addresses of the Anchor Money Market and Terra token. These changes would not be reflected in the existing subwallets. This could lead the subwallets to use incorrect addresses for the Anchor market and Terra token.

Recommendation

We recommend that subwallets query the subwallet factory contract config for both the Anchor market and Terra token contract addresses. Additionally, we recommend removing the ability to update these values from the `update_config` function in the subwallet contract.

Status: Resolved

Resolved in `d46c191`

2. Product factory updates could lead to inconsistent state

Severity: Major

During the creation of a fixed recurring subscription contract the address of the job registry contract is specified - see `contracts/sub1-fixed-recurring-subscriptions/src/contract.rs:74`. This address is then stored in the config of the subscription contract.

The product factory contract is able to update the address of the job registry contract. This change would not be reflected in the existing recurring subscription contract – unlike the protocol fee and other configuration parameters defined in the product factory, which are queried prior to use rather than stored. This could lead the subscriptions to use an incorrect address for the job registry contract.

Recommendation

We recommend that the subscription contract query the product factory contract config for the job registry address as is done for the protocol fee - `contracts/sub1-fixed-recurring-subscriptions/src/contract.rs:693`.

Additionally, we recommend removing the ability to update these values from the `update_config` function in the subscription contract.

Status: Resolved

Resolved in d46c191

3. Protocol fee decimal places incorrectly specified

Severity: Major

The protocol fee is defined in basis points during instantiation of the `product-factory` contract config. During both, the instantiation and update of the config, the protocol fee is verified to ensure that is less than the maximum of 5%. As the fee is represented in basis points 5% is defined as 500, which implies that in fixed-point arithmetic 4 decimal places are being used - i.e. 500 / 10,000 = 0.05.

On lines `sub1-fixed-recurring-subscriptions/src/contract.rs:346` and `697`, decimal places are applied incorrectly when calculating the ratio of the protocol fee. In each case, the protocol fee could be greater than 100%. Which could lead to users being charged excessive amounts for using the protocol.

Recommendation

We recommend that the decimal places used for protocol fee calculations be applied consistently with the product factory definition and thus be updated to 10,000 as used correctly elsewhere in the code.

Status: Resolved

Resolved in d46c191

4. Owner cannot freeze admins

Severity: Minor

When executing the function `execute_freeze`, as found in `contracts/admin-core/src/contract.rs:84`, the contract verifies that the sender is a whitelisted admin. Once this function is executed the contract owner will be unable to append new admins or otherwise unfreeze the admin list. This is contrary to the documentation definition of the owner being the highest-privilege access.

Further access verification to the `execute_freeze` function uses the `can_modify` function which checks that the sender is on the admin list. However, in the case that the owner's address is not on said list this check would fail. Preventing the owner from successfully calling the `execute_freeze` function.

Recommendation

We recommend that the verification use the `can_change_admins` function, found in `contracts/admin-core/src/state.rs:29`, bringing the functionality in line with the documentation.

Status: Resolved

Resolved in `c6ac04e`

5. Owner cannot execute set permissions

Severity: Minor

In file `contracts/subwallet/src/contract.rs:389` the function `execute_set_permissions` verifies that the message sender is an admin whitelisted for the wallet. However, if the sender is the owner and not present on the admin whitelist then the execution will fail. As the owner has the highest-privilege access to a wallet this is contrary to the documentation.

Recommendation

We recommend the sender verification defined on line `contracts/subwallet/src/contract.rs:389` be changed to `!cfg.is_admin(info.sender.as_ref()) && config.owner_addr != info.sender` which would ensure that both the owner and admins are able to set permissions.

Status: Resolved

Resolved in `c6ac04e`

6. Admin whitelist cannot be unfrozen

Severity: Minor

In file `contracts/admin-core/src/contract.rs:54` the function `execute_freeze` freezes the list of whitelisted admins for a suberra wallet. Once frozen there is no way to reverse this decision, even for the wallet owner. It may be necessary to remove, or add, an admin due to scenarios such as the case of a leaked private key.

Recommendation

We recommend adding a function for the wallet owner to unfreeze the admin whitelist to enable the owner to add or remove admins from the whitelist.

Status: Resolved

Resolved in `c6ac04e`

7. Protocol fee decimal places should be defined as constants

Severity: Informational

The protocol fee is defined in basis points during instantiation of the `product-factory` contract config. During both, the instantiation and update of the config, the protocol fee is verified to ensure that is less than the maximum of 5%. As the fee is represented in basis points 5% is defined as 500, which implies that in fixed-point arithmetic 4 decimal places are being used - i.e. 500 / 10,000 = 0.05.

However, throughout the code base, there is no constant variable defined to use in protocol fee calculations. Rather values are used on an ad-hoc basis, including:

- `sub1-fixed-recurring-subscriptions/src/contract.rs:271`
- `sub1-fixed-recurring-subscriptions/src/contract.rs:346`
- `sub1-fixed-recurring-subscriptions/src/contract.rs:697`
- `sub2-p2p-recurring-transfers/src/contract.rs:451`

Recommendation

We recommend defining a constant variable, e.g. `MAX_FEE_DECIMAL = 10_000`, that can be used in all calculations associated with the protocol fee throughout the codebase.

Status: Resolved

Resolved in `d46c191`

8. Unbounded number of contract admins

Severity: Informational

There is no limit defined in `contracts/admin-core/src/state.rs:10` to the number of addresses that can be whitelisted as Suberra wallet admins. Should the number of admins be excessively large this could make execution functions costly and query functions unperformant. In extreme cases, this could cause the querier to run out of gas.

Recommendation

We recommend setting an upper bound on the number of admins that can be whitelisted upon deployment with an additional function to update the max admin number should it be required in the future.

Status: Resolved

Resolved in d46c191

9. Total number of subscriptions increments on cancellation undo

Severity: Informational

When a cancelled subscription is undone through the `execute_subscribe` function, `contracts/sub1-fixed-recurring-subscriptions/src/contract.rs:389`, the function `create_subscription` is called. This function increments the total number of subscriptions by 1. In the case of an undone cancellation, this leads to the number of subscriptions being incremented unnecessarily.

Recommendation

We recommend the function `create_subscription` does not increment the number of subscriptions in the case of an cancellation undo.

Status: Resolved

Resolved in d46c191

10. Unnecessary variable assignment

Severity: Informational

On the line `contracts/sub2-p2p-recurring-transaction/src/contract.rs:33`, the variable `init_msg` is assigned as a clone of the instantiation message. However, this variable is not needed and the passed `msg` could be used in all instances where `init_msg` is used.

Recommendation

We recommend removing the variable `init_msg` and simply using the `msg` as is done elsewhere in the same function.

Status: Resolved

Resolved in 0a1e86f

11. Wallet permission logic function unused

Severity: Informational

In file `contracts/subwallet/src/contract.rs:459` the function `can_execute` provides the ability to assess whether an arbitrary address has permission to execute a message. The identical logic is also applied in the function `execute_execute`, `contracts/subwallet/src/contract.rs:109`. However, the function `can_execute` is not used.

Duplication of the identical logic increases both the overall size of the contract and complexity.

Recommendation

We recommend that the function `can_execute` be re-used in the `execute_execute` function.

Status: Acknowledged

12.Overflow checks not enabled for release profile

Severity: Informational

The following packages and contracts do not enable `overflow-checks` for the release profile.

- `contracts/subwallet/Cargo.toml`
- `contract/admin-core/Cargo.toml`
- `packages/suberra-core/Cargo.toml`

While enabled implicitly through the workspace manifest, a future refactoring might break this assumption.

Recommendation

We recommend enabling overflow checks in all packages, including those that do not currently perform calculations, to prevent unintended consequences if changes are added in future releases or during refactoring. Note that enabling overflow checks in packages other than the workspace manifest will lead to compiler warnings.

Status: Resolved

Resolved in [0a1e86f]