Audit Report

C2X

v1.0

January 26 – May 9, 2022

Table of Contents

License

Disclaimer

This audit has been performed by

Oak Security

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

Introduction

Audit process and duration

The security consulting and auditing process was executed in several phases between January 26, 2022 and the delivery date of this final report.

Purpose of This Report

Oak Security has been engaged by Com2uS Holdings Corporation to perform a security audit of the C2X 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/c2x-dev/cw20
Commit hash: `3162ffbecf3ce653592a1ee1e9f5fab57ef05c50`

https://github.com/c2x-dev/lock-nft
Commit hash: `6ec46f54f57a789e0edf51bae74ff2f3304ceea8`

https://github.com/c2x-dev/beta-game-launcher
Commit hash: `6b731a5b2d7d54f1c4730c774f33394e58e22c12`

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 code implements C2X, a blockchain gaming platform in the Terra ecosystem. The scope of the audit includes a legacy version of cw20 code, an NFT lock system, and a beta game launcher, which allows users to support sales of newly launched games onboarding to the C2X platform.

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 improving 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 in a security audit and vice versa.

Summary of Findings

No	Description	Severity	Status
1	Setting total supply incorrectly when registering a beta invitation will lead to failures when claiming and distributing tokens	Major	Resolved
2	Updating main token address may cause state inconsistency	Major	Resolved
3	Decimals are not accounted for in calculation of `main_token_amount`	Major	Resolved
4	Equal value of sold amount and soft cap causes inconsistent evaluation of condition	Major	Resolved
5	Missing validation on `main_token_distributions` during update config	Minor	Resolved
6	Duplicate account creation would inflate token total supply	Minor	Resolved
7	`Owner` and `main_token` addresses are not validated during instantiation	Minor	Resolved
8	`cw20` is based on a legacy version of the CW20 standard	Minor	Resolved
9	Incorrect contract name in `cw20`	Minor	Resolved
10	Centralization of NFT unlocking	Informational	Acknowledged
11	Admin operations implement custom logic	Informational	Acknowledged
12	No error message for query of non-existent token	Informational	Resolved
13	Overflow checks not enabled for release profile	Informational	Acknowledged
14	Canonical address transformations are inefficient	Informational	Resolved
15	Typographic error found in codebase	Informational	Resolved

Code Quality Criteria

Criteria	Status	Comment
Code complexity	**Low**	-
Code readability and clarity	**Medium-High**	-
Level of documentation	**Medium**	-
Test coverage	**Low-Medium**	Current tests did not fully pass. We also recommend that at least all happy paths are covered by tests.

Detailed Findings

1. Setting total supply incorrectly when registering a beta invitation will lead to failures when claiming and distributing tokens

Severity: Major

In `contract/src/execute.rs:203-209` of the `beta-game-launcher` repository, the total supply value is not validated to be higher than the total required amount as seen in lines `398-407` and lines `526-535`. If the total supply value is set to be lower than required, it would cause `claim`, `token_distribute`, or both to fail due to minting capacity reaching the maximum. This will cause users to be unable to claim their shares of game and fan tokens. Likewise, the admin would be unable to distribute main and game tokens to recipients, causing a loss of funds for the users and recipients.

Recommendation

We recommend removing the comments from lines `198-206` and replacing line `204` with:

```
if game_token_distribution_sum != _total_supply {
    …
}
```

Status: Resolved

Resolved in [3507e76]

2. Updating main token address may cause state inconsistency

Severity: Major

In `contract/src/execute.rs:146-148` of the `beta-game-launcher` repository, the address of the main token can be updated to a different value. If the main token address is updated while the contract has an existing `config.main_token` value, it would cause an inconsistency between the contract's state and the actual token balance held in the contract. As a result, this would cause a series of problems such as users being unable to withdraw their tokens due to insufficient contract balance.

Recommendation

We recommend removing the functionality to update the main token address.

Status: Resolved

Resolved in [3507e76]

3. Decimals are not accounted for in calculation of `main_token_amount`

Severity: Major

In `contract/src/execute.rs:300` of the `beta-game-launcher` repository, the `buy_amount` is multiplied by the `invitation_price` to ensure that sufficient `main_token` is supplied. However, as the `game_token`, `main_token` and `invitation_price` can have different numbers of decimal places, the calculation may be performed incorrectly. This could lead to a user being able to purchase at a lower price than expected.

Recommendation

We recommend removing the `invitation_price_decimals` and ensuring that `invitation_price` is always returned to be consistent with decimal places used by `main_token`.

Status: Resolved

Resolved in [bed142b]

4. Equal value of sold amount and soft cap causes inconsistent evaluation of condition

Severity: Major

In `contract/src/execute.rs:371-373` of the `beta-game-launcher` repository, the `claim` function would evaluate `passed` as true only if the sold amount is higher than the soft cap value. This is inconsistent with the `token_distribute` functionality as seen in lines `508-510` since the execution would continue if the sold amount is equal to or greater than the soft cap value.

In an edge case where both the sold amount and the soft cap value are equal, the `claim` function would determine the invitation as a failure but the `token_distribute` function would determine the invitation as a success. In a worst-case scenario, the user's funds would be incorrectly distributed to the recipients which might cause slow users unable to have their funds refunded back due to insufficient balance held in the contract.

Recommendation

We recommend modifying the `claim` function to evaluate the invitation as successful by checking whether the sold amount is equal to or higher than the soft cap amount in line `371`.

Status: Resolved

Resolved in [3507e76]

5. Missing validation on `main_token_distributions` during update config

Severity: Minor

In `contract/src/execute.rs:154-156` of the `beta-game-launcher` repository, the distribution values in `main_token_distributions` are not validated as seen in lines `27-36`. This means that there's a possibility that the sum of the distribution rate can be over 100%, potentially causing the `token_distribute` function in lines `515-524` to either distribute more funds than intended or fail due to insufficient funds.

We consider this to be a minor issue since it can only be caused by the owner.

Recommendation

We recommend verifying the total distribution rate in `main_token_distributions` to be equal to `Decimal::one()` in lines `154-156`.

Status: Resolved

Resolved in `3507e76`

6. Duplicate accounts creation would inflate token total supply

Severity: Minor

In `src/contract.rs:63-71` of the `cw20` repository, duplicate accounts are not verified when creating initial accounts during the contract instantiation phase. If the same account address is passed twice in `create_accounts`, the account's balance would be overwritten via `BALANCES.save` but `total_supply` would still record the balance amount of both. As a result, the token's total supply would be inflated.

We consider this to be a minor issue since it can only be caused by the owner.

Recommendation

We recommend returning an error if duplicate accounts exist in the `initial_balances` vector that's passed into `create_accounts`.

Status: Resolved

Resolved in `88d715a`

7. `Owner` and `main_token` addresses are not validated during instantiation

Severity: Minor

In `contract/src/execute.rs:20` of the `beta-game-launcher` repository, during the handling of the `Instantiate` message, `owner` and `main_token` addresses are not validated. This may cause the contract to be initialized with invalid values.

This issue is also present during the config update phase in lines `146-152`.

Recommendation

We recommend validating the `owner` and `main_token` addresses before saving them in the state during contract instantiation and update config phase.

Status: Resolved

Resolved in `3507e76`

8. `cw20` is based on a legacy version of the CW20 standard

Severity: Minor

In the `cw20` repository there is a legacy implementation of cw20 token. As stated in that implementation repository[1]:

"This contract is modified for the purpose of migration from a Columbus-4 cw20 token contract to Columbus-5. Using the official version of cw20 is strongly recommended for other usages"

Unlike the latest version, the legacy version does not support the `marketing` field which is required in `contract/src/execute.rs` in lines `254-277`. As a result, using the legacy version of CW20 token code id would cause `register_beta_invitation` functionality to fail due to the differences between the latest CW20 contract token interface and the one used in this contract.

Recommendation

We recommend following the Terra suggestion using a more recent version of the CW20 implementation, in order to have better support and the latest fixes. As an example, issue 6 is mitigated in the latest version of the CW20 token contract.

Status: Resolved

Resolved in `e806bb2`

[1] https://github.com/terra-money/cosmwasm-contracts/tree/main/contracts/cw20-legacy

9. Incorrect contract name in `cw20`

Severity: Minor

In `src/contract.rs:21` of the `cw20` contract, the `CONTRACT_NAME` is set equal to `crates.io:cw20-base`. This is the template contract name of the CW20 contract.

Recommendation

We recommend changing the contract name to a custom one.

Status: Resolved

Resolved in `88d715a`

10. Centralization of NFT unlocking

Severity: Informational

In `contract/src/execute.rs:95-98` of the `lock-nft` repository, only the owner of the contract can unlock users' NFTs and return them back to them. If the owner key is compromised, the remaining locked NFTs in the contract will be inaccessible.

Recommendation

We recommend following appropriate best practices when managing the owner key, for example using a multi-signature and hardware wallets.

Status: Acknowledged

11. Admin operations implement custom logic

Severity: Informational

In all contracts there is the concept of `Admin`, which is an account that has exclusive permissions to execute some messages. As there is already a battle-tested implementation of this that takes care of validation and it's ready out of the box, it should be better to use that one instead of using custom logic.

Recommendation

We recommend using https://docs.rs/cw-controllers/latest/cw_controllers/struct.Admin.html from `cw-controllers` crate.

Status: Acknowledged

12. No error message for query of non-existent token

Severity: Informational

In contract/src/query.rs:28-35 of the lock-nft repository, users can query the owner of a token using an nft_address and token_id as the key. In the case that there is no token with such a key an empty string is returned as the owner. This leads to a worsened user experience as incorrect queries cannot be easily diagnosed.

Recommendation

We recommend that in the case of a query for a non-existent token a descriptive error message is returned.

Status: Resolved

Resolved in 02ee183

13. Overflow checks not enabled for release profile

Severity: Informational

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

- lock-nft/contract/cargo.toml
- beta-game-launcher/contract/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: Acknowledged

14. Canonical address transformations are inefficient

Severity: Informational

While previously recommended as a best practice, usage of canonical addresses for storage is no longer encouraged. The background is that canonical addresses are no longer stored in a canonical format, so the transformation just adds overhead without much benefit. Additionally, the codebase is more complicated with address transformations.

Recommendation

We recommend removing any transformation from human to canonical addresses and using the new `Addr` type for validated addresses instead.

Status: Resolved

Resolved in `3507e76`

15. Typographical errors found in codebase

Severity: Informational

In `contract/src/execute.rs` of the `beta-game-launcher` repository, there were several typographical errors found in lines `35-301`. Specifically, the word “queal” should be replaced with “equal” while the word “worng” should be replaced with “wrong”.

Recommendation

We recommend correcting the misspellings mentioned above.

Status: Resolved

Resolved in `3507e76`