certora

Security Assessment Final Report

CLAYNOSAURZ

Claynosaurz NFT

April 2025

Prepared for Claynosaurz

Table of contents

Project Summary

Project Scope

Project Name	Repository (link)	Latest Commit Hash	Fix commit hash	Platform
Claynosaurz NFT	https://github.com/Claynosaurz-Inc/sui-nft-smart-contract	659bb15	e9f86f9	Sui

Project Overview

This document describes the specification and verification of **Claynosaurz NFT** using manual code review findings. The work was undertaken from **Apr 11, 2025** to **Apr 18, 2025**

The following contract list is included in our scope:

```
{
    move/claynosaurz/sources/boosterpack.move
    move/claynosaurz/sources/popkins_nft.move
    move/claynosaurz/sources/cosmetic_nft.move
    move/claynosaurz/sources/reedemamble_nft.move
    move/claynosaurz/sources/registry.move
    move/claynosaurz/sources/achievement.move
    move/claynosaurz/sources/claynosaurz_nft.move
    move/claynosaurz/sources/stickers_nft.move
    move/claynosaurz/sources/display.move
}
```

Protocol Overview

Claynosaurz NFT manages the creation of various type of NTFs, the project allows users to purchase a boosterpack which would then be burned in exchange for newly created NFTs.

The project contains a registry module to manage the various roles in the protocol, and a display module to comply with Sui’s display standard.

Findings Summary

The table below summarizes the findings of the review, including type and severity details.

Severity	Discovered	Confirmed	Fixed
Critical	-	-	-
High	-	-	-
Medium	-	-	-
Low	4	4	4
Informational	1	1	1
Total	5	5	5

Severity Matrix

Impact	High	Medium	High	Critical
	Medium	Low	Medium	High
	Low	Low	Low	Medium
		Low	Medium	High
		Likelihood		

Detailed Findings

ID	Title	Severity	Status
L-01	update_nonce_expiration_window() lacks version validation	Low	Fixed
L-02	Nonce expiration isn't verified before removal	Low	Fixed
L-03	claim_boosterpack() data signature isn't typed	Low	Fixed
L-04	Bytes isn't verified to be fully consumed by claim_boosterpack()	Low	Fixed

Low Severity Issues

L-01 `update_nonce_expiration_window()` lacks version validation		
Severity: **Low**	Impact: **Low**	Likelihood: **Medium**
Files: registry.move	Status: Fixed	

Description: The function `update_nonce_expiration_window()` doesn't verify that the `registry`'s version matches the version of the module. This would allow calling the function also when a newer version is released.

```JavaScript
/// Updates the nonce expiration window.
public fun update_nonce_expiration_window(
    _: &AdminCap,
    registry: &mut Registry,
    window: u64,
) {
    registry.nonce_expiration_window = window;
}
```

Recommendations: Add version validation to the function

Customer's response: Fixed in commit f4e8671

Fix Review: Fix looks good

L-02 Nonce expiration isn't verified before removal

Severity: **Low**	Impact: **Medium**	Likelihood: **Low**
Files: {Files}	Status: Fixed	

Description: `registry::update_nonce_expiration_window()` allows the admin to remove old nonces from the registry.
However, the function doesn't verify that the nonces have expired already and relies on the admin to check that.
It might be best to double-check that programmatically and not only rely on the admin

Recommendation: Add a check in the function to verify that the nonce has expired.

Customer's response: Fixed in commit f4e8671

Fix Review: Fix looks good. Just note that in case that `nonce_expiration_window` is ever increased this might make expired and removed nonces valid again, so look out for this if the window is ever increased.

L-03 `claim_boosterpack()` data signature isn’t typed

Severity: **Low**	Impact: **Medium**	Likelihood: **Low**
Files: move/claynosaurz/sources/boosterpack.move	Status: Fixed	

Description: The `claim_boosterpack()` function receives signed data (`bytes` and `signature`) that contains information about the boosterpack to be claimed.
The signature isn’t ‘typed’ (like in EIP-712), this might allow an attacker to re-use data that was signed from the same address for other purposes (e.g. signing a Sui tx).

Recommendations: Change the function so that the signature signs a typed data hash.

Customer’s response: Fixed in e9f86f9, we’ve added a domain name at the beginning of the signed data.

Fix Review: Fix looks good, just note that if you ever intend to expand to other chains you’ll have to include the chain name/id in the domain to distinguish between chains.

L-04 Bytes isn't verified to be fully consumed by `claim_boosterpack()`

Severity: **Low**	Impact: **Medium**	Likelihood: **Low**
Files: move/claynosaurz/sources/boosterpack.move	Status: Fixed	

Description: `claim_boosterpack()` decodes a `bytes` parameter which contains encoded data about the booster pack.
However, it doesn't verify the bytes were fully consumed by decoding (i.e. the bytes vec is empty after 'peeling' all the data).
Meaning, the function might accept data that's longer than expected and contains excess bytes at its end. This kind of data wasn't signed to be used for claiming boosterpack and should be rejected by the function.

Recommendation: Verify the bytes are fully consumed after peeling, and revert if not.

Customer's response: Added the suggested check in commit e9f86f9

Fix Review: Fix looks good

Informational Severity Issues

I-01. The `mint()` function in the NFT modules should have the `registry` parameter as an immutable reference

Description: The `mint()` function takes a `registry` parameter as a mutable reference of a `Registry` share object.

However, this object is only being read from by the contract and does not modify any of its fields.
Thus, the `registry` parameter can be supplied as a simple read-only reference, preventing any unexpected alteration of the object.

Recommendation:
Change the `mint()` function signature in the following NFT contracts and remove the `mut` keyword:
- achievement.move
- claynosaurz_nft.move
- cosmetic_nft.move
- popkins_nft.move
- reedemamble_nft.move
- stickers_nft.move

Customer's response: Removed the `mut` keyword in commit f4e8671

Fix Review: Looks good

Disclaimer

About Certora

Certora is a Web3 security company that provides industry-leading formal verification tools and smart contract audits. Certora's flagship security product, Certora Prover, is a unique SaaS product that automatically locates even the most rare & hard-to-find bugs on your smart contracts or mathematically proves their absence. The Certora Prover plugs into your standard deployment pipeline. It is helpful for smart contract developers and security researchers during auditing and bug bounties.

Certora also provides services such as auditing, formal verification projects, and incident response.