Known Origin Contracts v4 Audit

October 2022

By CoinFabrik

Introduction

CoinFabrik was asked to audit the contracts for the KnownOrigin project. First we will provide a summary of our discoveries, and then we will show the details of our findings.

Scope

The audited files are from the git repository located at https://github.com/knownorigin/known-origin-contracts-v4. The audit is based on the commit `b897b08d7d9a5777dc5ba37b5d460308fa85655a`.

The audited files are:

- `./contracts/KODABaseUpgradeable.sol`
- `./contracts/KODASettings.sol`
- `./contracts/Konstans.sol`
- `./contracts/ERC721/ERC721KODACreator.sol`
- `./contracts/ERC721/ERC721KODACreatorFactory.sol`
- `./contracts/ERC721/ERC721KODAEditions.sol`
- `./contracts/ERC721/extensions/ERC721KODACreatorBuyWithBuyItNow.sol`
- `./contracts/ERC721/interfaces/IERC721.sol`
- `./contracts/ERC721/interfaces/IERC721KODACreator.sol`
- `./contracts/ERC721/interfaces/IERC721KODACreatorWithBuyItNow.sol`
- `./contracts/ERC721/interfaces/IERC721KODAEditions.sol`
- `./contracts/errors/KODAErrors.sol`
- `./contracts/handlers/ICollabFundsHandler.sol`
- `./contracts/interfaces/IKODAAccessControlsLookup.sol`
- `./contracts/interfaces/IKODABaseUpgradeable.sol`
- `./contracts/interfaces/IKODASettings.sol`
- `./contracts/interfaces/IKODATokenUriResolver.sol`

The scope of the audit is limited to those files. No other files in this repository were audited. Its dependencies are assumed to work according to their documentation. Also, no tests were reviewed for this audit.

Analyses

Without being limited to them, the audit process included the following analyses:

- Arithmetic errors
- Outdated version of Solidity compiler
- Race conditions

- Reentrancy attacks
- Misuse of block timestamps
- Denial of service attacks
- Excessive gas usage
- Missing or misused function qualifiers
- Needlessly complex code and contract interactions
- Poor or nonexistent error handling
- Insufficient validation of the input parameters
- Incorrect handling of cryptographic signatures
- Centralization and upgradeability

Summary of Findings

We found 2 medium issues and 6 minor issues. Also, several enhancements were proposed.

Security Issues

ID	Title	Severity	Status
ME-01	Unbounded Commissions May Be Greater Than %100	Medium	Fixed
ME-02	Commissions May Be Updated To Values Larger Than The Maximum	Medium	Fixed
ME-02	Artist or Creator With Larger-Than-Expected Commissions	Medium	Acknowledged
MI-01	Unchecked Initialization Values Leading To Denial Of Service	Minor	Fixed
MI-02	Favor Pull Over Push On Commission Payments	Minor	Acknowledged
MI-03	Function `_mintTransfer()` May Be Called On Minted Token	Minor	Fixed
MI-04	Mint To `address(0)` Possible	Minor	Fixed
MI-05	Admin May Overwrite Implementation	Minor	Fixed
MI-06	Function flagBannedContract() Fails to Provide Functionality	Minor	Fixed

Security Issues Found

Severity Classification

Security risks are classified as follows:

- **Critical:** These are issues that we manage to exploit. They compromise the system seriously. They must be fixed **immediately**.
- **Medium:** These are potentially exploitable issues. Even though we did not manage to exploit them or their impact is not clear, they might represent a security risk in the near future. We suggest fixing them **as soon as possible**.
- **Minor:** These issues represent problems that are relatively small or difficult to take advantage of, but can be exploited in combination with other issues. These kinds of issues do not block deployments in production environments. They should be taken into account and be fixed **when possible**.

Issues Status

An issue detected by this audit can have four distinct statuses:

- **Unresolved**: The issue has not been resolved.
- **Acknowledged**: The issue remains in the code, but is a result of an intentional decision.
- **Resolved**: Adjusted program implementation to eliminate the risk.
- **Partially resolved**: Adjusted program implementation to eliminate part of the risk. The other part remains in the code, but is a result of an intentional decision.
- **Mitigated**: Implemented actions to minimize the impact or likelihood of the risk

Critical Severity Issues

No issues found.

Medium Severity Issues

ME-01 Unbounded Commissions May Be Greater Than %100

While `KODASettings::platformPrimaryCommission`, `platformSecondaryCommission()`, `_editionRoyaltyPercentage[]` may be

bounded from above, their sum is not when computing `platformProceeds` and could exceed 100%. For example, in `ERC721KodaCreatorWithBuyItNow.buyEditionToken()` or `buyToken()`. When this happens, each call will increasingly transfer funds from the contract (draining the contract) and revert when funds are not available (causing a denial of service).

Recommendation

Check in every function paying commissions that these do not add up to more than a maximum that must be smaller than or equal to %100.

Status

Fixed. Commissions are bounded by constants and cannot be %100 or higher.

ME-02 Commissions May Be Updated To Values Larger Than The Maximum

The functions `KODASettings::updatePlatformPrimaryCommission()` and `KODASettings::updatePlatforSecondaryCommission()` can be called by a user with the admin role to modify commissions without the 95_000 restriction.

Recommendation

Require that the value is smaller than the established maximum.

Status

Fixed. Update functions now revert if the new commission is bigger than a constant equivalent to %50.

ME-03 Artist or Creator With Larger-Than-Expected Commissions

According to KODA's model, the value received by the artist is the listing price minus commissions. Since a user with the admin role can update commissions at any time (through `KODASettings::updatePlatformPrimaryCommission()` or `KODASettings::updatePlatforSecondaryCommission()`, an artist or creator may offer tokens for sale with one set of commission values but these later get changed to larger values outside of his control.

Recommendation

Allow the artist/creator to establish a maximum commission he is willing to accept.

Status

Acknowledged. Developers explained this is the intended behavior. Explicitly: "The platform commission is currently handled in this way to save the gas cost of recording a fixed value at the time of listing since this almost never changes and

would not do with a public announcement. We will make sure this is clear in any documentation."

Minor Severity Issues

MI-01 Unchecked Initialization Values Leading To Denial Of Service

The initialization function `ERC721KodaCreateFactory::initialize()` for `ERC721KodaCreateFactory.sol` is standard checks which could result in denial of service. For example, `_receiverImplementation != address(0)`.

Recommendation

Make sure all invalid situations are filtered when calling the initialization function and setters.

Status

Fixed. Initializer function now checks for zero address.

MI-02 Favor Pull Over Push On Commission Payments

Funcion `ERC721KodaCreatorWithBuyItNow::buyEditionToken()` and others use calls to pay commissions. This goes against the pull over push best practice, which indicates that the (commission) funds should be made available for the commissioned and it should be them that transfer themselves the payments. Note that if some of the recipients of these calls were to reject the payment for any reason, the buying of tokens is locked.

Recommendation

Favor pulls over push, assigning commissions to the commissioned and add a function they can use to transfer their commissions to themselves.

Status

Acknowledged. Developers mention they prefer push over pull as artists expect to receive funds immediately on sale; commissions forwarded so additional logic is not required in these creator-owned contracts to allow platform to withdraw; and [this] eliminates potential loss of funds if balance is always zero in these contracts.

MI-03 Function `_mintTransfer()` May Be Called On Minted Token

The function `ERC721KODAEditions::_mintTransfer()` can be called to mint a token with an edition ID to a specific recipient. The function does not check who the owner is or if the editionID has been already minted. It is a private function that includes calls to `_beforeTokenTransfer()` and `_afterTokenTransfer()` which

are both virtual and with empty code. So while a contract inheriting from ERC721KODAEditions could implement these, the implementation is currently missing.

No public function can be used to call `_mintTransfer()` as it stands, and hence the issue is marked as minor for its low impact. Yet, extra care needs to be put into contracts implementing minting and inheriting from `ERC721KODAEditions.`

Recommendation
Add ownership checks and validate that the edition has not been minted, ensuring a minimum of security.

Status
Fixed. Mint functions were made private, and the checks were delegated to the functions calling mint.

MI-04 Mint To address(0) Possible

In `ERC721Editions::_mintBatch()` and `ERC721Editions::_mintConsecutive()` the recipient may be `address(0)` which should be forbidden. This is checked, e.g., in `_mintSingleEdition()` on the same contact.

Recommendation
Revert when address is 0.

Status
Fixed. Functions now check for `address(0)` and revert.

MI-05 Admin May Overwrite Implementation

Function `ERC721KODACreatorFactory::addCreatorImplementation()` allows the admin role to overwrite an implementation. While the could allow modifying the pairing implementationName - implementationAddress, probably under a different function (name), this should be discouraged as this name is used to retrieve the address and deploy a funds handler (e.g., a call to `deployCreatorContractAndFundsHandler()` with a given name could be front runned so that a different contract is deployed).

Recommendation
Revert when the implementation name is assigned.

Status

Fixed. The function now checks for repeats.

MI-06 Function flagBannedContract() Fails to Provide Functionality

The function `flagBannedContract()` from `ERC721KODACreatorFactory` does check access controls but implements no further action. There is no register of flagged contracts that can be actionable, only an event is emitted.

Recommendation

Include the expected functionality, and add the corresponding documentation so the user understands the expected behavior.

Status

Fixed. Development team mentions this is the intended functionality, as an event is emitted to be used for indexing/off-chain effects. Documentation should reflect this.

Enhancements

These items do not represent a security risk. They are best practices that we suggest implementing.

ID	Title	Status
EN-01	Constant Defined Twice	Implemented
EN-02	Name Constants	Implemented
EN-03	OZ Interface/Contract Copied Not Referenced	Not implemented
EN-04	Gas Savings With External Functions	Implemented
EN-05	Different Compiler Versions	Implemented
EN-06	Unused Code	Implemented

EN-01 Constant Defined Twice

Location:

- `contracts/KODASettings.sol:23,`
- `contracts/Konstants.sol:8`

The constant MODULO is defined twice. Although it has the same value in both contracts, it is recommended to either inherit or rename.

Recommendation

Remove repeated constants.

Status
Implemented.

EN-02 Name Constants

The value 95_00000 used in ERC721KODAEditions and KODABaseUpgradeable should be a named constant.

Recommendation
Define a constant that can be referenced every time it is needed.

Status
Implemented.

EN-03 OZ Interface/Contract Copied Not Referenced

The repository includes a version of `IERC721.sol` from OZ which is not the latest and is copied instead of the preferred NPM import.

Recommendation
Import from `@openzeppelin` the latest version of the contract.

Status
Not implemented. Developers mention that interface inherits from IERC165 which is handled separately in KnownOrigin contracts, as a result the import is not possible.

EN-04 Gas Savings With External Functions

Consider setting `ERC721KODACreatorFactory.getHandler()` as external. When a function is public, it copies array arguments to memory thus consuming gas, while external functions can read the calldata directly.

Recommendation
Set the function as external instead of public.

Status
Implemented.

EN-05 Different Compiler Versions

Throughout the repository, the following compiler versions are used: `'0.8.16'`, `'^0.8.0'`, `'^0.8.1'`, `'^0.8.16'`, `'^0.8.2'`. It is a best practice to use the latest one, this should provide the best security available and consistency.

Recommendation
Use the latest pinned version of the compiler for all contracts (without using the ^).

Status
Implemented.

EN-06 Unused Code

Location: `contracts/ERC721/ERC721KODAEditions.sol:996-1007`

The function `ERC721KODAEditions::_mintBatch()` is never used.

Recommendation
Remove unused code.

Status
Implemented.

Other Considerations

The considerations stated in this section are not right or wrong. We do not suggest any action to fix them. But we consider that they may be of interest for other stakeholders of the project, including users of the audited contracts, owners or project investors.

Centralization

Management is mostly centralized, and it is the users with Admin role that can update contracts, pause others, or even disable primary sales. This limits the artists and operators' capabilities to sell tokens.

Upgrades

Contracts are upgradeable and there are also dependencies between contracts. This introduces some threats with moderate risk. For example, `MODULO` is part of `Konstants.sol` which can be upgraded. But the value is consumed by `ERC721Editions` as a denominator of `editionRoyaltyPercentage`. So a change in

the former without an accompanying change in the latter could have as a consequence that commissions are off by orders of magnitude.

Privileged Roles

These are the privileged roles that we identified on each of the audited contracts.

- **Owner**: This role is managed by the access control module. A user with this role may change royalties percentages (through `KODABaseUpgradeable::updateDefaultRoyaltyPercentage()`), pause the `KODABaseUpgradeable`, disable/enable sales in `ERC721KODAEditions` or update the edition creator, or mint and list tokens in `ERC721KODACreatorWithBuyItNow`.
- **Admin**: This is a role managed by the access control module. It is used by `KODASettings` and `ERC721KODACreatorFactory` to authorize upgrades, change the platform, update commissions, the platform or the funds receiver implementation, and update access controls.
- **VerifiedArtists**. This is a role in the access management module. A verified artist may deploy `ERC721KODACreator` instances and fund handlers.

Changelog

- 2022-10-06 – Initial report based on commit `b897b08d7d9a5777dc5ba37b5d460308fa85655a`.
- 2022-11-08 - Revision based on commit `10c62c5a9bf76f1035a098d4c0bdbd2385428c8c`.

Disclaimer: This audit report is not a security warranty, investment advice, or an approval of the KnownOrigin v4 project since CoinFabrik has not reviewed its platform. Moreover, it does not provide a smart contract code faultlessness guarantee.