Kudos Audit

ZK Labs Report

DEAN EIGENMANN

2018-11-05

Contents

Kudos Audit

This audit was performed on the commit hash e18fbc1. The mentioned issues were addressed in commit hash 15de21

Kudos.sol

This contract extends the following Open Zeppelin contracts: - ERC721Token - Ownable

`SafeMath` is used for all `uint256` variables.

General issues

- Multiple functions are `payable` allowing `eth` to get stuck in this contract.

General Suggestions

- The code between `L92-106` can be moved into a seperate function. This would allow the code to be shared between both the `clone` and the `mint` functions.

mint

A `Kudo` struct is created with the passed data. If there are currently no enteries in the `kudos` array, a default struct is pushed. Then the created struct is pushed, the `clonedFromId` value for the newely pushed struct is then set to its own `id`. Finally the `_mint` and `_setTokenURI` function are called.

Suggestions

- Does this function really need to be `payable`?
- The variable declarations of `_numClonesInWild` & `_clonedFromId` can be removed.
- The content of the checks between `L49-54` could be moved into the `constructor` to avoid needing to check this every time.

clone

This function allows the cloning of a `Gen0` Kudo. It is first ensured that the number of existing clones added by the number of clones to be created is not greater than the allowed number of clones. It is

then ensured that the amount of eth sent is greater than or equal to the cost of cloning. Next, the cloning fee is calculated and the value is transfered to the contract `owner`. Next, the fee is subtracted from the cloning cost, and the amount is transfered to the `owner` of the current `ERC721` token. The number of clones is then updated to add in the clones which will be created. An iteration is done to create the amount of `Kudos` required, each of which do not allow clones.

Issues

- Surplus `eth` will get stuck in this contract, as we can send more than the `cloningCost`, but the excess is never sent back.

burn

This function is responsible for burning a `Kudos` token. Initially it retrieves the `Kudo` for the passed `id` If the passed `id` is not equal to the `clonedFromId` for the specified `Kudo`, the `clonedFromId` is used to get the parent `Kudo` where the `numClonesInWild` attribute is updated by subtracting by `1`. Finally the specified `Kudo` is removed from the `kudos` mapping. The parent `_burn` function is then called.

Suggestions

- Does this function really need to be `payable`?

setCloneFeePercentage

This function allows the `owner` to set the new cloning fee, it is ensured that the fee is between the range of `0` and `100`.

Issues

- Potential to frontrun fees by increasing the `fee` when a `clone` transaction is pending.