

SECURITY AUDIT REPORT

AngryPenguins customization smart contract


by  ARDA
on November 17, 2022



Table of Content

Audit Summary	3
Code issues & Recommendations	4
C1: Long NFT attributes wrongly top encoded	4
C2: No validation of NFT attributes	5
C3: Unnecessary dual storages for slots and items	6
C4: Imprecise condition on balance for unequipping	7
C5: Two different storages have the same identifier	8
C6: Equipments can't be refilled	9
C7: Overly complex Kvp type	10
C8: Scattered and unnecessary normalization of NFT attributes	11
C9: Server can change the CID of already rendered images	12
C10: Inadequate VecMapper for __images_to_render	13
C11: Useless sc_panic_self	14
C12: Confusing method names	15
C13: sc_print!(...) present	16
C14: Changing ipfs_gateway invalidates all image urls	17
C15: Unnecessary normalization of gateway	18
C16: Unnecessary get_equipable_name function	19
C17: Unnecessary intermediate functions	20
C18: Unnecessary custom top_encode / top_decode for Item	21
C19: Useless ownership checks	22
C20: Useless equipable_name_format logic	23
C21: Useless utils	24
C22: Useless ItemAttributes struct	25
C23: Useless EquipableNftAttributes methods	26
C24: Unnecessary intermediate files	27
Disclaimer	28

Audit Summary

Scope

- **Repository:** <https://github.com/Angry-Penguins-Colony/sc-customize-nft>
- **Commit:** 82c929815e6886e6785f1d16baaaee3176e75771
- **Path to Smart contract:** ./

Report objectives

1. Reporting all issues in smart contract **code** alongside recommendations
2. Reporting all issues in smart contract **test** alongside recommendations
3. Reporting all **other** issues alongside recommendations

Issues

Number of issues reported and issues remaining at last reviewed commit
7bcdf601b08456a46aaffd63006faae96ca55a0:

Severity	Reported			Remaining		
	Code	Test	Other	Code	Test	Other
Critical	0	0	0	0	0	0
Major	2	0	0	0	0	0
Medium	10	0	0	0	0	0
Minor	12	0	0	0	0	0

Code issues & Recommendations

C1: Long NFT attributes wrongly top encoded

Severity: Major

Status: Fixed

Location

src/structs/equippable_nft_attributes.rs

Description

If NFT attributes top-encode into more than 512 bytes, only the first 512 bytes will be kept in the top-encoding.

Recommendation

Use the `load_512_bytes` method in the `top_encode` method of `EquippableNftAttributes`.

C2: No validation of NFT attributes

Severity: Major

Status: Fixed

Description

The slot and name of NFT attributes can currently be any ManagedBuffer. However, there are values that the slot and name should be forbidden to take, otherwise critical issues would appear. Exhaustively:

- The slot should not contain any ":" or ";" . Otherwise the top decoding would not work as expected.
- The name should not contain any ":" or ";" or be equal to "unequipped". Otherwise the top decoding would not work as expected and items with the name "unequipped" will be lost.

Recommendation

The `__set_item_no_check`, `set_slot_of`, `__get_index` methods should verify that the slot doesn't contain any ":" or ";" . Any other method that stores a slot should do the same checks.

The `__set_item_no_check` method should verify that the name doesn't contain any ":" or ";" or is equal to "unequipped". Any other method that stores a name should do the same checks.

The value `b"unequipped"` should be stored in a constant to ensure the same value is used everywhere.

C3: Unnecessary dual storages for slots and items

Severity: Medium

Status: Fixed

Description

Currently the smart contract has:

- a storage `token_of` that associates to a couple `(slot, name)` a couple `(token_id, token_nonce)`
- a storage `__slot_of` that associates to a `token_id` a `slot`
- a method `register_item` that fills the storage `slot_of`
- a method `fill` that fills the storage `token_of`

Because of the presence of these two storages, the `customize` method has to:

1. verify that the `token_id` of an equipment has a slot associated
2. create the `item` associated to the equipment
3. verify that the `item` just created has a token associated
4. verify that the token associated matches the token that has been sent

The `customize` method has a lot to do and verify and it can be easy to forget something important.

Recommendation

A simpler model would be to only have a `token_of` storage:

- the `register_item` method would take as parameter an item and receive a token and then fill the storage `token_of` after having verifying the item is not already present,
- the `fill` method would take as parameter an item and receive a token and then verifies the token received matches the token associated to the item,
- the `customize` method would receive the token of the penguin and of the equipments and would take as parameter a `EquippableNftAttributes` that only contains the slots that changes. If the slot is `None`, then it is unequipped; if the slot is `Some<*>`, then it is equipped.

C4: Imprecise condition on balance for unequipping

Severity: Medium

Status: Fixed

Description

Currently, an equipment can only be removed from a penguin if the smart contract holds at least twice this equipment. However, this constraint can be loosened: the smart contract should only require to hold once the equipment.

Recommendation

Replacing this condition:

```
... <= BigUint::from(1u64)
```

by this condition:

```
... == 0
```

C5: Two different storages have the same identifier

Severity: Medium

Status: Fixed

Description

The `equippable_name_format` and `__images_to_render` have the same storage identifier that is `equippable_name_format`. This means that they both write to the same location in storage.

Hopefully, in this precise situation, this is not too severe as the `equippable_name_format` is never being written, but it could have been a critical issue if other storages had the same identifier.

Recommendation

Changing the identifier of `__images_to_render` and verifying that all the storages have correct identifiers.

C6: Equipments can't be refilled

Severity: Medium

Status: Fixed

Description

Let's say that hats are added to the smart contract by calling the `fill` method. If it appears later on that more hats need to be added, there is currently no way to do it because calling the `fill` method will fail because of this require:

```
require!(
  self.token_of(item).is_empty(),
  "The item with name {} is already registered. Please, use another
  name.",
  item_name
);
```

Recommendation

A possible solution would be to remove this require and in the case `self.token_of(item)` is not empty, to verify that the token already associated with `item` matches the token that have been sent.

C7: Overly complex Kvp type

Severity: Medium

Status: Fixed

Description

Currently, `kvp` struct is defined as below:

```
struct Kvp<M: ManagedTypeApi> {
    pub slot: ManagedBuffer<M>,
    pub item: Option<Item<M>>,
}

pub struct Item<M: ManagedTypeApi> {
    pub name: ManagedBuffer<M>,
    pub slot: ManagedBuffer<M>,
}
```

So `kvp` contains twice the `slot` information, once in the `slot` field of `kvp` and once in the `slot` field of `Item`. This is unnecessary and because `slot` is contained twice, checks have to be done in order to be sure both slot values are equal.

Having twice the slot stored makes the data structure more difficult to understand and forces to add additional checks to be sure the same slot is stored twice.

Recommendation

Simplifying the `kvp` struct this way:

```
struct Kvp<M: ManagedTypeApi> {
    pub slot: ManagedBuffer<M>,
    pub name: Option<ManagedBuffer<M>>,
}
```

C8: Scattered and unnecessary normalization of NFT attributes

Severity: Medium

Status: Fixed

Description

The normalization of NFT attributes (i.e. the automatic conversion in the correct format) is done in multiple different places in the smart contract: in `set_slot_of`, in `is_slot_empty`, in `__set_item_no_check`, in `top_decode`, in `top_encode`. With a scattered normalization, there are more chances that the normalization is not going to be done correctly. Moreover, if someone adds a new entry point, he should think about normalizing.

The normalization of attributes is also not necessary and should be avoided because:

1. the caller might not expect the data he passes to be changed by the smart contract on his behalf,
2. normalization can usually be done outside of the smart contract, hence reducing the complexity of the smart contract.

The smart contract should rather validate data is correctly formatted.

Recommendation

The `__set_item_no_check`, `set_slot_of` and `__get_index` methods should be the only places where there should be validation. They might also verify the slot doesn't contain spaces.

The utils `to_lowercase` and `capitalize` can be removed and replaced by a `validate_slot` util.

The `equals_ignore_case` util could also be removed.

C9: Server can change the CID of already rendered images

Severity: Medium

Status: Fixed

Description

The server that generates images is authorized to change the CID of already rendered images. If the server gets compromised, the hacker would be able to change the CID of all the already rendered images. This is a privilege the server doesn't need and so should not be granted.

Recommendation

In the `set_cid_of` method, the line `self.__cid_of(&attributes).set(cid);` should be moved inside the `if`
`self.__images_to_render().has_item(&attributes) {` condition.

C10: Inadequate VecMapper for __images_to_render

Severity: Medium

Status: Fixed

Description

A `VecMapper` is used for the `__images_to_render` storage. A `VecMapper` is not made for easily checking if one item is in it or removing one of its items. However, the smart contract needs to check if an image is already in the rendering queue or remove an image from it. Therefore, in order to do so, a consequent amount of custom logic has been added.

An `UnorderedSetMapper` would be better suited as it comes built-in with all the things the smart contract needs, making useless all the custom logic that have been added and can then be removed, thus the smart contract simpler and so more secure.

Recommendation

To use an `UnorderedSetMapper` instead of a `VecMapper` for the `__images_to_render` storage and to remove:

- the implementation of `PartialEq` for `EquippableNftAttributes`
- the file `src/utils/managed_vec_utils.rs`
- the file `src/utils/vec_mapper_utils.rs`

C11: Useless `sc_panic_self`

Severity: Medium

Status: Fixed

Location

`src/utls/macros.rs`

Description

It is crucial that smart contracts can stop the execution of a transaction whenever anormal conditions occur. Inside an Elrond smart contract, the instruction `sc_panic` can be used. Inside the method of a structure, this same instruction can't be used.

Therefore, a `sc_panic_self` instruction have been added to the codebase to be able to panic inside the method of a structure. In the meantime, a `M::error_api_impl().signal_error` instruction have been added to the Rust framework for Elrond smart contracts.

As it is crucial that the smart contract successfully panics when needed, it is safer to use the built-in instruction for this rather than an homemade one.

Recommendation

Use `M::error_api_impl().signal_error` instead of `sc_panic_self` and remove the file `src/utls/macros.rs`.

C12: Confusing method names

Severity: Medium

Status: Fixed

Description

The auditor has been confused by the name of several methods and feels their names could be improved to better explicit the role of such methods, and hence make the code easier to understand and less error prone.

Recommendation

The following list is not exhaustive and it might be beneficial if the programmer could go through all the methods and ensure the ideal naming has been chosen:

- `render_image` is a confusing name as no rendering happens in the so-called method. Rather it enqueues an image that will be later on rendered off-chain. A better name might be `add_image_to_render`.
- `kvp` is not a very explicit name. A better one might be `EquippableNftAttribute`. Moreover the field `name` might be better named `value`.
- The name `set_item` doesn't explicit the fact the slot should be empty. A better name might be `set_item_if_empty`.
- `__set_item_no_check` should rather be `set_item`.
- namings that end with `_of` such as `token_of`, `__permissions_set_cid_of`, `__slot_of`, ... are a bit problematic as we don't understand "of what?". For example, `token_of` could be renamed into `token_of_item`.
- namings that start with `__` such as `__permissions_set_cid_of`, `__images_to_render`, ... should not start with `__` as `__` doesn't say anything on what the method is doing. `__` is more of a hack for quickly naming and usually warns of potential naming issues.

C13: `sc_print!(...)` present

Severity: Minor

Status: Fixed

Description

A `sc_print!(...)` is present in `mint_equippable` and `render_image` methods. It has no impact on the functioning of the smart contract but clutters the source code. It should only be used during debugging.

Recommendation

Removing these `sc_print!(...)`.

C14: Changing ipfs_gateway invalidates all image urls

Severity: Minor

Status: Fixed

Description

If the `ipfs_gateway` is changed, then the smart contract won't be able to remember url for all the images that have already been generated.

Recommendation

For each generated image, the entire URL should be stored rather than only the CID. The `ipfs_gateway` storage can be removed.

C15: Unnecessary normalization of gateway

Severity: Minor

Status: Fixed

Description

In the case issue Changing ipfs_gateway invalidates all image urls would not be fixed, an other issue should be mentioned concerning the `gateway` that is passed to `init`.

This URL is first normalized with `gateway.append_trailing_character_if_missing(b'/')` and then stored in storage.

Normalization (i.e. automatically converting in the correct format) should be avoided as much as possible because:

1. the caller might not expect the data he passes to be changed by the smart contract on his behalf,
2. normalization can usually be done outside of the smart contract, hence reducing the complexity of the smart contract.

Normalization should usually be replaced by verification, i.e. instead of automatically converting the data in the correct format, rather verifying the data is already in the correct format.

Recommendation

Remove the `append_trailing_character_if_missing` utils and rather verify in the `init` method that the last character of `gateway` is a slash.

C16: Unnecessary get_equippable_name function

Severity: Minor

Status: Fixed

Description

The `get_equippable_name` function is not necessary as there is already a `get_token_name` that could be used.

Recommendation

Remove the `get_equippable_name` function and use `get_token_name` instead.

In the `fill` method, use `get_token_name` function to get `item_name`.

C17: Unnecessary intermediate functions

Severity: Minor

Status: Fixed

Description

Intermediate functions help making a big function to be smaller. This simplifies the big function but makes it harder to fully understand it as somebody now has to read the big function plus all the intermediate functions that will likely be in a bunch of different files.

Hence intermediate functions are not always a good idea, especially when the logic that they abstract is small. In this case, it is best to remove them.

The current source code of the smart contract contains several intermediate functions that doesn't abstract much and that would be best to remove.

Recommendation

Move the logic inside the following intermediate functions directly to the parent function:

- `update_equippable`
- `mint_equippable`
- `calculate_hash`
- `parse_equippable_attributes`
- `enqueue_image_to_render`
- `dequeue_image_to_render`
- `to_kv_buffer`
- `empty_slot`

C18: Unnecessary custom top_encode / top_decode for Item

Severity: Minor

Status: Fixed

Description

The `Item` struct doesn't need to have a custom `top_encode / top_decode`. Only the `EquippableNftAttributes` struct needs to have a custom `top_encode / top_decode`.

Recommendation

Moving the `top_encode / top_decode` logic of `Item` to the `top_encode / top_decode` of `EquippableNftAttributes`.

C19: Useless ownership checks

Severity: Minor

Status: Fixed

Description

The `register_item`, `fill`, `claim` methods already have a `#[only_owner]` annotation but still check if the caller is the owner. This logic is useless.

Recommendation

Removing this check logic.

C20: Useless `equippable_name_format` logic

Severity: Minor

Status: Fixed

Description

The logic around the concept of `equippable_name_format` is never used in the smart contract.

Recommendation

Remove:

- the `equippable_name_format` storage,
- the `EQUIPPABLE_NAME_FORMAT_NUMBER` and `ERR_INIT_MISSING_NUMBER_FORMAT` constants,
- the `get_next_equippable_name` method,
- the `replace` and `contains` utils in `src/utls/managed_buffer_utils.rs`,
- the `to_ascii` utils in `src/utls/u64_utils.rs`.

C21: Useless utils

Severity: Minor

Status: Fixed

Description

Some utils are used nowhere while bloating the codebase and making it harder to grasp.

Recommendation

Remove:

- the `to_hex` utils in `src/utils/u64_utils.rs`,
- the `split_last_occurrence`, `remove_first_char`, `remove_first_and_last_char`, `hex_to_u64`, `ascii_to_u64` utils in `src/utils/managed_buffer_utils.rs`.

C22: Useless ItemAttributes struct

Severity: Minor

Status: Fixed

Location

src/structs/item_attributes.rs

Description

The `ItemAttributes` struct in `src/structs/item_attributes.rs` is never used.

Recommendation

Remove the file `src/structs/item_attributes.rs`.

C23: Useless EquippableNftAttributes methods

Severity: Minor

Status: Fixed

Description

The `fmt`, `new` and `get_count` methods of the `EquippableNftAttributes` structure are never used.

Recommendation

Remove them.

C24: Unnecessary intermediate files

Severity: Minor

Status: Fixed

Location

`src/structs/equippable_nft_attributes.rs`

Description

Intermediate files help making a big file to be smaller. This simplifies the big file but makes it harder to fully understand it as somebody now has to read the big file plus all the intermediate files that will likely be in a bunch of different folders.

Hence intermediate files are not always a good idea, especially when the logic that they abstract is small. In this case, it is best to remove them.

After all the previous issues are fixed, the source code of the smart contract will contain several intermediate files that doesn't abstract much and that would be best to remove.

Recommendation

Currently there are 4 folders and 16 files in the `src` folder. After all the previous issues are fixed, the auditor would advise the following structure:

- a `lib.rs` file that would contain the logic of the current `lib.rs` and all the files in the `libs` folder,
- a `structs.rs` file that would contain the logic of all the files in the `structs` folder,
- a `utils.rs` file that would contain the logic of all the files in the `utils` folder,
- a `constants.rs` file that would contain the same logic as the current `constant.rs` file.

Disclaimer

The security audit report makes no statements or warranties, either expressed or implied, regarding the security of the code, the information herein or its usage. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract.

This report does not constitute legal or investment advice. It is for informational purposes only and is provided on an "as-is" basis. You acknowledge that any use of this report and the information contained herein is at your own risk. The authors of this report shall not be liable to you or any third parties for any acts or omissions undertaken by you or any third parties based on the information contained herein.

