SMART CONTRACT AUDIT

Certification

VESTING CONTRACT

SMART CONTRACT AUDITS

October 14th, 2022 / v. 0.2

Source code version: be97236bcd0346014eec5ea51b2d09141f066b3a

Structure and Organization of the Document

Some sections are more important than others. The most critical areas are at the top, and the less critical sections are at the bottom. The issues in these sections have been fixed or addressed and will show by the "Resolved" or "Unresolved" tags. Each case is written so you can understand how serious it is, with an explanation of whether it is a risk of exploitation or unexpected behavior.

CRITICAL

These issues can have a dangerous effect on the ability of the contract to work correctly.

HIGH

These issues significantly affect the ability of the contract to work correctly.

MEDIUM

These issues affect the ability of the contract to operate correctly but do not hinder its behavior.

LOW

These issues have a minimal impact on the contract's ability to operate.

INFORMATIONAL

These issues do not impact the contract's ability to operate.

Issues

1. Update total_tokens_claimed on claimTokensUnallocated()

Not addressed / HIGH

Description: The function in fact does claim the tokens that were allocated and not claimed, but the tokens are not marked as being claimed. Since this counter is used in calculus in addBeneficiary() function, unexpected results may arise.

! Possible fix to research

```
Update the claim tokens counter.
```

! Response

```
Not addressed. The storage is not meant to be updated.
```

! Status

```
Accepted & Closed. Keep in mind that if the contract is ever made public,
integrators may misunderstand the meaning of this function, better document it
well.
```

2. Add removeGroup() function

Not addressed / LOW

Description: If the user mistakes something when adding a 'key' name such as 'Team', it cannot be undone, and the name 'Team' will not be able to be reused.

! Possible fix to research

```
Add removeGroup() function. The group would probably have to have no users
added.
```

! Response

```
Not addressed. There will be a handful of groups and we won't need to delete any.
```

! Status

```
Accepted & Closed.
```

3. Ease of addBeneficiary() logic

Not addressed / **INFORMATIONAL**

Description: The function makes sure that the contract holds enough tokens in order for it to be able to give users the allocated amount of tokens. It relies on counters to achieve this:

```
let new_total_tokens_allocated = self.total_tokens_allocated().get() + &tokens_allocated;
let total_tokens_claimable =
   &new_total_tokens_allocated - &self.total_tokens_claimed().get();
let contract_balance = self.blockchain().get_esdt_balance(
   &self.blockchain().get_sc_address(),
   &self.token_identifier().get(),
   0,
);
require!(
   contract_balance >= total_tokens_claimable,
   “not enough tokens in vesting contract”
);
```

! Possible fix to research

The function can greatly be simplified by just adding on the fly the vested tokens for the user that is being added.

! Response

Not addressed. The logic is good but we do feel comfortable making this change at this moment, being very close to go-live.

! Status

Accepted & Closed.

Verification Conditions

1 Adding a new user to a group does not exceed the group total allocation

```
require!(
         new_group_current_allocation <= group_info.max_allocation,
         “group exceeds max allocation”
);
```

2 Adding a the same user to a group will not work

```
for beneficiary_id in beneficiary_ids.iter() {
        let info = self.beneficiary_info(beneficiary_id).get();
        require!(
           info.group_name != group_name,
           “beneficiary is already defined for this group”
);
}
```

3 Cannot remove an user that is marked as can_be_removed == false

```
require!(
         beneficiary_info.can_be_revoked,
         “beneficiary cannot be removed”,
);
```

Suggestions (Optional)

1. Remove **set_if_empty** with **0** values. Reading an uninitialized storage of type **BigUint** or **bool** will result in **0** and **false** values.

```
Response: Not addressed. For better visibility, the calls will stay.

Status: Accepted & Closed.
```

2. Rename **claimTokensUnallocated()** to **claimUnallocatedTokens()** or even better **claimUnclaimedTokens()** since the function seems to claim the tokens that have not yet been claimed.

```
let total_tokens_claimable =
              self.total_tokens_allocated().get() - self.total_tokens_claimed().get();
```

```
Response: Not addressed. Naming is good enough.

Status: Accepted & Closed.
```

3. Update **elrond-wasm-rs** framework version. It s best when deploying a contract to have the latest framework version, since interfaces might change and bugs may be fixed.

```
Response: Not addressed because of time trouble.

Status: Accepted & Closed.
```

4. Update **release_percentage** to have values between (0, 100_000] for more flexibility.

```
Response: Not addressed. Percentages will be natural numbers.

Status: Accepted & Closed.
```

5. No need for instructions as:

```
let mut beneficiary_ids;
if self.beneficiary_ids(&addr).is_empty() {
         beneficiary_ids = ManagedVec::new();
} else {
         beneficiary_ids = self.beneficiary_ids(&addr).get();
}
```

Reading an uninitialized **ManagedVec** from the storage will return a new and empty one.

```
Response: Done.

Status: Accepted & Closed.
```

6. Use **SetMapper** instead of **SingleValueMapper<ManagedVec<u64>>** for **beneficiary_ids**.

```
Response: Not addressed because of time trouble.

Status: Accepted & Closed.
```

7. Add **groupName** to events. For example, if a User is part of two groups and gets revoked from one, from just the generated event, one cannot tell which group it was.

```
Response: Done.

Status: Accepted & Closed.
```

8. Make the contract **non-payable** by receiving the vested tokens while either adding a new group or adding a new beneficiary.z`

```
Response: Not addressed because of time trouble.

Status: Accepted & Closed.
```

9. Make a **claimUnclaimedTokens()** function just for tokens belonging to a certain group.

```
Response: Not addressed because we won’t need it.

Status: Accepted & Closed.
```

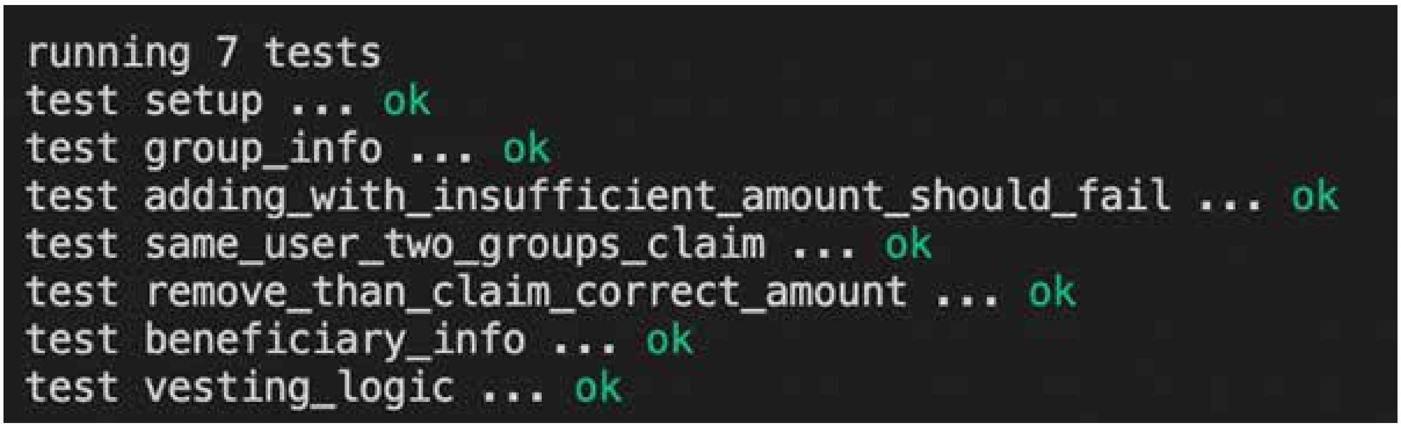
10. Port tests from **Mandos** to **Rust Testing Framework**. Having tests that are written in rust, more complex test flows can be added and the existing ones can be easily extended.

```
Response: Not addressed because of time trouble.

Status: Accepted & Closed.
```

11. Solve **Clippy** warnings.

```
Response: Done.

Status: Accepted & Closed.
```

Test results



```
running 7 tests
test setup ... ok
test group_info ... ok
test adding_with_insufficient_amount_should_fail ... ok
test same_user_two_groups_claim ... ok
test remove_than_claim_correct_amount ... ok
test beneficiary_info ... ok
test vesting_logic ... ok
```

Initially audited source code version: **be97236bcd0346014eec5ea51b2d09141f066b3a**

Afterwards, reviewed & audited **PR #17** that contains the fixes.