MultiversX xLaunchpad Guaranteed Tickets v2

MultiversX smart contract - Security audit by Arda

Repository: *https://github.com/multiversx/mx-launchpad-sc*
Smart contract path: *launchpad-guaranteed-tickets-v2*
Initial commit: *44f14ec2ffe4a317d310065b3c0cfae4c26470e3*
Final commit: *ddf0836868b054e6de30dc3bef937cafe6d438c0*

Issues

1. Incorrect distribution of leftover tickets: lead to unfair winner selection and might lead to less launchpad tokens distributed and less funds raised than expected

Status	Solved
Severity	Critical
Commit (if not initial)	
Location file (optional)	
Additional note (optional)	

Description

When trying to distribute a leftover ticket, a random ticket position is sampled in the range `[offset, last_ticket_id]`, and if the selected ticket is already winning, the offset is simply incremented by 1. Therefore:

- The ticket at position `offset` will never be selected as a winner from now on. This means that the distribution of leftover tickets is unfair, because small ticket positions have higher chances to be filtered/skipped quickly while big ticket positions have higher chances to be selected as winning.
- This might prevent distributing all leftover tickets, because incrementing the offset means we are skipping a non-winning ticket, and there might insufficiently many remaining non-winning tickets in the range `[offset + 1, last_ticket_id]`

Example: Consider the following setup:

- The project deposited launchpad tokens for 4 winners and there are 2 guaranteed tickets,
- There were 6 tickets confirmed: [1, 2, 3, 4, 5, 6].
- Guaranteed tickets are tickets 1 and 6.
- Initial winner tickets selected are tickets 1, 2, 3.

At this point, there is 1 leftover ticket left to distribute, since ticket 1 is already winning. However, based on the current algorithm, here are the chances for a non-winning ticket to be selected as a winning leftover ticket

- **Ticket 4:** it has 1/3 chances to be selected as a winner.
- **Ticket 5:** it has 1/2 chances to be selected as a winner.
- **No winners:** there is ⅙ chances that no winners are selected.

In summary, ticket 5 has more chances to win than ticket 4, and moreover, it is possible that no leftover tickets are selected, resulting in 1 less ticket sold than expected.

Recommendation

In `try_select_winning_ticket`, if the selected ticket is already winning, we suggest permuting the position of the selected ticket with the ticket at the current position. This way, the ticket at the current position will still have a chance to be selected in the subsequent selections.

This also allows to remove the comment above `distribute_leftover_tickets`:

```
None
// TODO - add a check if current_ticket_pos > last_ticket_pos
```

Indeed, with the previous fix, it is certain that the selection will eventually stop without any further check, as the stopping criterion made in `are_all_tickets_distributed` will eventually be met.

Finally, we recommend to re-introduce the unit test (discussed with Sorin Ionut Petreasca) where the above check seemed to be necessary, and to verify that the check is not necessary anymore, i.e. that the unit test passes and the distribution of leftover tickets ends successfully.

2. Guaranteed tickets that users already won are not redistributed as leftover tickets and result in less launchpad tokens distributed and less funds raised than expected

Status	Solved
Severity	Critical
Commit (if not initial)	
Location file (optional)	
Additional note (optional)	

Description

When distributing guaranteed tickets to a user, if the user already won some tickets during the previous winner selection, the excess of guaranteed tickets is not re-allocated as leftover tickets.

This is because, in `process_guaranteed_tickets`, when `guaranteed_tickets > user_winning_tickets`, we don't reallocate the winning tickets `user_winning_tickets` as leftover tickets:

```
None
if guaranteed_tickets > user_winning_tickets {
  let tickets_to_win = guaranteed_tickets - user_winning_tickets;
  self.select_additional_winning_tickets(ticket_range, tickets_to_win, op);
  // We don't reallocate, i.e. we don't do op.leftover_tickets +=
user_winning_tickets
} else {
  op.leftover_tickets += guaranteed_tickets;
}
```

Consequently, the final number of winners will be smaller than expected, and the project will have distributed less tokens and raised less funds than expected.

Example: Let's say the project announces a total of 6 final winners: `nr_winning_tickets = 6.` Alice and Bob are the only 2 participants, and Alice and Bob both have 2 guaranteed tickets. So now `nr_winning_tickets = 2` and `total_guaranteed_tickets = 4.`

- Alice and Bob confirm 3 tickets each.
- During winner selection, Both Alice and Bob earn 1 ticket (recall `nr_winning_tickets = 2` so no more winners can be selected).
- Now `process_guaranteed_tickets` takes place. Both Alice and Bob earn `guaranteed_tickets - user_winning_tickets = 1` additional ticket.

There are no leftover tickets. So overall 4 tickets are winning, although we expected 6 winners.

This would have worked fine if, in step 5, we would have increased the leftover by `user_winning_tickets = 1` both for Alice and Bob, leading to 2 additional tickets being distributed.

Recommendation

In `process_guaranteed_tickets`, when `guaranteed_tickets > user_winning_tickets`, we recommend reallocating the winning tickets `user_winning_tickets` as leftover tickets:

```
None
if guaranteed_tickets > user_winning_tickets {
  let tickets_to_win = guaranteed_tickets - user_winning_tickets;
  self.select_additional_winning_tickets(ticket_range, tickets_to_win, op);
  op.leftover_tickets += user_winning_tickets
} else {
  op.leftover_tickets += guaranteed_tickets;
}
```

Finally, we recommend implementing a unit test witnessing that already won tickets are correctly re-distributed. The scenario with Alice and Bob from the example above would work.

3. Random selection of winners might be manipulated

Status	Solved
Severity	Major
Commit (if not initial)	
Location file (optional)	
Additional note (optional)	Resolved differently by allowing only owner or user account to make 1st call to select_winners,and distribute_guaranteed_tickets_endpoint.

Description

Anyone can call `select_winners`, and the 1st call determines the random seed that will be used to determine all winners.

Therefore, a malicious user could call `select_winners` with a smart contract and revert the transaction if he is unsatisfied with the first few selected winners. Over multiple consecutive txs in a single block, he could repeat the operation until he is satisfied with the first few selected winners.

Moreover, the manipulation of the selection of winners for leftover tickets is also possible, although much harder to accomplish because the randomness is decided at the time guaranteed tickets are distributed. Therefore, the attacker would have to simulate on his own the result of the whole distribution of guaranteed tickets (which is deterministic), and based on this result, determine if the random seed will result in the selection he wants for winners of leftover tickets.

Recommendation

For the 1st call to `select_winners`, and `distribute_guaranteed_tickets_endpoint`, we suggest verifying that the caller is either the owner, or an account from another shard. This will prevent manipulations.

Note that we should not assume that the launchpad smart contract is on shard 1. Rather, we can compute the shard of the launchpad smart contract, and verify that it differs from the caller's shard.

4. Unset unlock schedule prevents users from claiming tokens

Status	Solved
Severity	Major
Commit (if not initial)	
Location file (optional)	
Additional note (optional)	

Description

In case no unlock schedule is set, `compute_claimable_tokens` returns that no tokens can be claimed, preventing users from ever withdrawing their launchpad tokens.

Recommendation

In case no unlock schedule is set, we recommend considering by default that there is no vesting for the launchpad tokens, and that users can withdraw all their launchpad tokens as soon as the claim period has started.

We further recommend adding a unit test covering the scenario where no unlock schedule is set at claim time.

5. Unlock schedule can be set after users bought tickets

Status	Solved
Severity	Major
Commit (if not initial)	
Location file (optional)	
Additional note (optional)	

Description

In case no unlock schedule is set, the owner can call `set_unlock_schedule` and set any unlock schedule. However, this means that users who bought their tickets now are forced to wait a possibly unwanted vesting period before they can withdraw.

Recommendation

In `set_unlock_schedule`, we recommend always checking that users did not start buying tickets, i.e. we replace the requirement that

```
None
current_block < confirmation_period_start_block ||
self.unlock_schedule().is_empty()
```

with the requirement that `get_launch_stage == LaunchStage::AddTickets`.

Moreover, we recommend adding a unit test showing that the unlock schedule can't be set even if empty, once the confirmation period has started.

6. One user can have too big ticket range and could stuck a whole launch

Status	Solved
Severity	Medium

Commit (if not initial)	
Location file (optional)	
Additional note (optional)	Solved but with a bigger constant `MAX_TICKETS_ALLOWANCE=255`

Description

There is no limit on the number of tickets in a user's ticket range.

If this number is too big, this could make some endpoints run out of gas, e.g. endpoints calling `select_additional_winning_tickets` and `winning_tickets_in_range`, and the launchpad would be stuck, preventing users from withdrawing their launchpad tokens and payment tokens.

Recommendation

In `add_tickets_with_guaranteed_winners`, we recommend verifying for each user that the maximum allowed confirmed tickets `total_tickets_allowance` is smaller than a limit ~~`MAX_GUARANTEED_TICKETS_INFO`~~, `MAX_TICKETS_ALLOWANCE` e.g. 25.

7. One user can have too many guaranteed tickets info and could stuck a whole launch

Status	Solved
Severity	Medium
Commit (if not initial)	
Location file (optional)	
Additional note (optional)	

Description

There is no limit on the number of guaranteed tickets info `guaranteed_tickets_info` that a user can have.

If this number is too big, the distribution of guaranteed tickets could fail by running out of gas, and the launchpad would be stuck, preventing users from withdrawing their launchpad tokens and payment tokens.

Recommendation

In `add_tickets_with_guaranteed_winners`, we recommend verifying for each user that the length of his guaranteed tickets information `guaranteed_tickets_info` is smaller than a limit `MAX_GUARANTEED_TICKETS_INFO`, e.g. 10.

8. Vesting schedule can have too many unlock milestones and prevent users from withdrawing their funds

Status	Solved
Severity	Medium
Commit (if not initial)	
Location file (optional)	
Additional note (optional)	

Description

There is no limit on the number of milestones in the vesting schedule of launchpad tokens.

If there are too many milestones, the iteration over milestones in `compute_claimable_tokens` could fail and prevent users from withdrawing their funds.

Recommendation

In `set_unlock_schedule`, we recommend enforcing that the number of milestones is smaller than a limit `MAX_UNLOCK_MILESTONES`, e.g. 50.

9. Possible underflow of “nr_winning_tickets” when unblacklisting

Status	Solved
Severity	Medium
Commit (if not initial)	
Location file (optional)	
Additional note (optional)	

Description

When a user is unblacklisted, in `remove_guaranteed_tickets_from_blacklist`, his previously recorded guaranteed tickets `guaranteed_tickets_added` are added back to the storage `total_guaranteed_tickets` and depleted from the storage `nr_winning_tickets`. However, it is possible that `guaranteed_tickets_added > nr_winning_tickets`, in which case the decrease of `nr_winning_tickets` would underflow, leading to a huge number of tickets `nr_winning_tickets` recorded in storage.

In turn, this would force the project to deposit an amount of launchpad tokens which is considerably bigger than expected.

Recommendation

In `remove_guaranteed_tickets_from_blacklist`, we suggest requiring that `guaranteed_tickets_added <= nr_winning_tickets`.

We further recommend adding a unit test witnessing that unblacklisting fails when it would actually make `nr_winning_tickets` underflow.

10. The launch stages are determined by block but block duration can change

Status	Not Solved
Severity	Medium
Commit (if not initial)	
Location file (optional)	
Additional note (optional)	This will not be solved, as the team considers it would be easy not to make a launch if there is an upcoming change of the block duration.

Description

The start time of each stage of a token launch (`AddTickets, Confirm, WinnerSelection, Claim`) is determined by a specific block. However, the duration of a block, currently ~6 seconds, is expected to be reduced in the future. If a reduction happens while a launch is ongoing, it would then shorten the duration of a stage.

For example, after a reduction of the block duration, the launch stage could quickly transition from `Confirm` to `WinnerSelection`, leaving insufficient time for users to buy tickets.

Recommendation

We suggest using timestamps instead of blocks in order to determine the start of the different stages.

11. Owner can postpone at will the claim period and users would withdraw their funds too late (or never)

Status	Not Solved
Severity	Medium
Commit (if not initial)	
Location file (optional)	
Additional note (optional)	It is considered unnecessary to resolve this issue by the team because the owner anyway has full control on the launchpad SC (he could upgrade it).

Description

By calling `set_winner_selection_start_block` or `set_claim_start_block`, the owner can postpone the time at which users will be allowed to claim and withdraw their funds. There is no constraint on the introduced delay or on the number of times the owner is allowed to postpone these times.

Recommendation

We suggest allowing the owner to call `set_winner_selection_start_block` and `set_claim_start_block` at most `MAX_CLAIM_TIME_CHANGES = 2` times, and each time to make sure that the introduced delay is not greater than `MAX_DELAY_FOR_CLAIM_TIME = 2` days.

In particular, we would have a storage `claim_time_changes` increased each time the owner calls `set_winner_selection_start_block` and `set_claim_start_block`, and in both these endpoints we would first check that `claim_time_changes < MAX_CLAIM_TIME_CHANGES`.

Alternative solution: if it is not needed to let the owner postpone the winner selection and the claim period, we can simply remove the endpoints `set_winner_selection_start_block` and `set_claim_start_block`.

12. Can’t unblacklist a user who has no guaranteed tickets

Status	Solved
Severity	Medium
Commit (if not initial)	
Location file (optional)	
Additional note (optional)	

Description

Unblacklisting a blacklisted user who had no guaranteed tickets would fail. This is because in this case, the storage `blacklist_user_ticket_status(user)` was not filled during blacklisting, and in turn `remove_guaranteed_tickets_from_blacklist` will fail when trying to read this storage:

```
None
let user_ticket_status = self.blacklist_user_ticket_status(&user).take();
```

Recommendation

In `remove_guaranteed_tickets_from_blacklist`, we recommend reading the storage `blacklist_user_ticket_status(user)` only if it is non-empty.

Moreover, we recommend adding a unit test where a user with no guaranteed tickets is unblacklisted.

13. Can’t refund non-payable smart contract or frozen address in same shard

Status	Not Solved
Severity	Medium
Commit (if not initial)	
Location file (optional)	
Additional note (optional)	SCs are forbidden in the launchpad (can’t KYC) and an explicit protection against SCs has been added. The case of a frozen account has not been handled.

Description

Blacklisting a non-payable smart contract in the same shard is impossible, as `refund_ticket_payment` would fail sending back tokens to that smart contract.

Therefore, if a malicious user wants to prevent being blacklisted, he could use a smart contract in the same shard and make it non-payable.

Similarly, it is impossible to blacklist an account in the same shard which is frozen for the payment token.

Finally, a user will be unable to claim his launchpad tokens if he is frozen for the payment token, as the refunding of the non-winning tickets would fail.

Recommendation

In `refund_ticket_payment,` we recommend verifying that the account is in the same shard and is either frozen or a non-payable smart contract. In this case, instead of sending the tokens, we would add the amount in a dedicated global storage `failed_refunds`, from which only the owner can withdraw by calling an only-owner endpoint `withdraw_failed_refunds`. In this way, if there are some failed refunds, the blacklisting would still work, and the owner would always have the ability to withdraw the failed refunds and reimburse users if needed.

14. No protection against release epochs too far in the future

Status	Solved
Severity	Medium
Commit (if not initial)	
Location file (optional)	
Additional note (optional)	

Description

There is no protection against release epochs which could be set by mistake too far in the future, e.g. 100 years, and would prevent users ever getting their vested tokens.

Recommendation

In `set_unlock_schedule`, we suggest verifying that the biggest release epoch is at most `MAX_RELEASE_EPOCH = 10 years.`

15. Project can't withdraw his launchpad tokens if no one bought tickets

Status	Solved
Severity	Medium
Commit (if not initial)	
Location file (optional)	
Additional note (optional)	

Description

If no one bought tickets, the storage `self.ticket_batch(FIRST_TICKET_ID)` is empty, and therefore the endpoint `filter_tickets` fails at the following line:

```
None
let run_result = self.run_while_it_has_gas(|| {
  let current_ticket_batch_mapper =
self.ticket_batch(first_ticket_id_in_batch);
  let ticket_batch: TicketBatch<Self::Api> = current_ticket_batch_mapper.get();
  ...
}
```

Therefore, the launch is stuck at the filtering phase, and the project will never be able to withdraw his launchpad tokens.

Recommendation

In `filter_tickets`, we suggest modifying the loop such that it starts by verifying whether all the tickets were processed, instead of doing it at the end of the loop:

```
None
let run_result = self.run_while_it_has_gas(|| {

if first_ticket_id_in_batch == last_ticket_id + 1 { STOP_OP }

let current_ticket_batch_mapper = self.ticket_batch(first_ticket_id_in_batch);
let ticket_batch: TicketBatch<Self::Api> = current_ticket_batch_mapper.get();
...

CONTINUE_OP
}
```

Finally, we recommend adding a unit test where no user buys any tickets, and showing that the project can process all the steps and withdraw his tokens at the end.

16. “run_while_it_has_gas” mechanism unreliable for distributing guaranteed tickets because of different gas costs to process different users

Status	Not Solved
Severity	Medium
Commit (if not initial)	
Location file (optional)	
Additional note (optional)	The team decided that the current mechanism will be used as it has already been tested multiple times on previous versions, and on the current version, the possible gas problems are not too constraining.

Description

In `select_guaranteed_tickets`, the “`run_while_it_has_gas`” mechanism is used to distribute guaranteed tickets to multiple consecutive users. The mechanism computes the gas cost for distributing guaranteed tickets to the 1st user, and considers that this gas cost will be the same for all subsequent users. However, this won’t be the case because the gas cost for processing each user is proportional to the number of user guaranteed ticket infos `user_ticket_status.guaranteed_tickets_info` and to the size of the user’s ticket range.

Consequently, this could make the selection of guaranteed tickets run out of gas because of a wrong estimate of the gas cost for processing each user. This could further make it difficult to adjust the gas limit of a transaction so as to make sure that the transaction will pass.

Recommendation

First, we suggest following the recommendations to other issues which bound the number of ticket infos of a user and the size of his ticket range.

Second, in `select_guaranteed_tickets`, instead of using the “`run_while_it_has_gas`” mechanism, we suggest checking that the gas left is greater than a conservative value, e.g. `50_000_000`, before processing a new user, i.e. we can replace the “`run_while_it_has_gas`” loop the following code inside `select_guaranteed_tickets`:

```
None
loop {

  let gas_left = self.blockchain().get_gas_left();
  if gas_left <= MIN_GAS_FOR_SELECTING_TICKET {
    break;
  }

  if users_left == 0 {
    break;
  }

  ...

  if user_ticket_status_mapper.is_empty() {
    continue;
  }

  ...
}
```

17. No protection against harmful durations for launch phases

Status	Not Solved
Severity	Medium
Commit (if not initial)	
Location file (optional)	

Additional note (optional)	The team decided that these parameters should be triple-checked off-chain upon deployment.

Description

In `require_valid_time_periods`, we don't protect against harmful time periods:

- If `winner_selection_start_block` is too far from `confirmation_period_start_block`, it could take too long before users who bought tickets can withdraw their funds.
- If `confirmation_period_start_block` is too close to `winner_selection_start_block`, it would not leave enough time for users to buy tickets.
- If `claim_start_block` is too far in future, users can't withdraw their tokens.

Recommendation

In `require_valid_time_periods`, we suggest introducing the following protections:

- Requiring that `winner_selection_start_block - confirmation_period_start_block` is between `MIN_BUY_TICKETS_STAGE_DURATION = 1 day` and `MAX_BUY_TICKETS_STAGE_DURATION = 10 days`.
- Requiring that `claim_start_block - winner_selection_start_block` is at most `MAX_DELAY_BEFORE_CLAIM = 5 days`.

18. Views for winning tickets return wrong result after user has claimed

Status	Not Solved
Severity	Minor
Commit (if not initial)	
Location file (optional)	
Additional note (optional)	This issue is known and the front-end/microservice does not need access to the view once user has claimed. Therefore it is not necessary to resolve it.

Description

Once a user has claimed some launchpad tokens, his ticket range is cleared. Thus, from now on, the view functions `get_winning_ticket_ids_for_address` and `get_number_of_winning_tickets_for_address` will return wrong results, i.e. that the user has won no tickets.

Recommendation

We suggest:

- Either making the view fail if the claim period has start, with an explicit error message explaining that the view functions are not reliable once users have started claiming;
- Or not clearing the ticket range of a user when he claims.

19. Possible u32 overflow of ticket ids

Status	Solved
Severity	Minor
Commit (if not initial)	
Location file (optional)	
Additional note (optional)	This is minor because so unlikely

Description

In `try_create_tickets`, it is possible that the last ticket of a user exceeds the maximum u32 value and starts back from 0:

```
None
let first_ticket_id = last_ticket_id_mapper.get() + 1;
let last_ticket_id = first_ticket_id + nr_tickets - 1;
ticket_range_mapper.set(TicketRange {
  first_id: first_ticket_id,
  last_id: last_ticket_id,
});
```

This would result in tickets of different users overlapping each other, and to the blocking of the overall launch, e.g. when there will be a need to iterate from `first_ticket_id` to `last_ticket_id`, the iteration would fail and the launch would be stuck.

Recommendation

In `try_create_tickets`, we recommend requiring that `first_ticket_id < MAX::u32 - nr_tickets`.

20. Unnecessary and misleading conditional property in “filter_tickets”

Status	Solved
Severity	Minor
Commit (if not initial)	
Location file (optional)	
Additional note (optional)	

Description

In `filter_tickets`, when processing the tickets of a user, we have this condition:

```
None
if self.is_user_blacklisted(address) || nr_confirmed_tickets == 0 {
  self.ticket_range_for_address(address).clear();
  current_ticket_batch_mapper.clear();
}
```

However, the conditional property that a user is blacklisted is:

- *Unnecessary*: if a user is blacklisted, he has 0 confirmed tickets, so the other conditional property `nr_confirmed_tickets == 0` would be true.
- *Misleading:* it suggests that it would be fine executing the lines under this condition even if the user does not have 0 confirmed tickets. However this would result in a wrong ticket filtering, as we would then increase the offset `nr_removed` by `(nr_tickets_in_batch - nr_confirmed_tickets)` instead of `nr_tickets_in_batch`. That would later stuck the launch completely when processing guaranteed tickets.

Recommendation

In `filter_tickets`, we require removing the conditional property `self.is_user_blacklisted(address)`. Instead we would have:

```
None
if nr_confirmed_tickets == 0 {
  self.ticket_range_for_address(address).clear();
  current_ticket_batch_mapper.clear();
}
```

21. Unnecessary function “select_winning_ticket”

Status	Solved
Severity	Minor
Commit (if not initial)	
Location file (optional)	
Additional note (optional)	

Description

The function `select_winning_ticket` does nothing but calling another function `try_select_winning_ticket`. Therefore it would be simpler calling `try_select_winning_ticket` directly each time instead of `select_winning_ticket`.

Recommendation

We recommend calling `try_select_winning_ticket` directly each time instead of `select_winning_ticket`, and deleting the function `select_winning_ticket`.

22. Blacklist already blacklisted users and unblacklist non-blacklisted users does not fail

Status	Solved
Severity	Minor
Commit (if not initial)	
Location file (optional)	
Additional note (optional)	

Description

Blacklisting already blacklisted users does not always fail, as well as unblacklisting non-blacklisted users. This is unexpected for the caller who would see the transaction as successful and believe that he blacklisted users who were not blacklisted, or unblacklisted only blacklisted users.

Recommendation

We recommend making the blacklisting endpoint fail if the user is already blacklisted, and the unblacklisting endpoint fail if the user is not blacklisted.

23. Missing vertical space

Status	Solved
Severity	Minor
Commit (if not initial)	
Location file (optional)	
Additional note (optional)	

Description

Between consecutive functions and storages, vertical spaces are included to improve readability. However there is no vertical space separating the storages `totalGuaranteedTickets` and `userTicketStatus`.

Recommendation

We recommend adding vertical space between the storages `totalGuaranteedTickets` and `userTicketStatus`.

Inherent Risks

1. Users must trust that admins will submit correct information about guaranteed tickets and allowances. If errors are made, users might earn less launchpad tokens or have less chances to win than they should.

Disclaimer