MultiversX Guild Factory

MultiversX smart contract - Security audit by Arda

Repository: *https://github.com/multiversx/sc-guilds-rs/*
Smart contract path: *guild-factory*
Initial commit: *7a394909b25514c329e19ce9b36d12c634cec464*
Final commit: *5583fe6e2bf12fdbf74170cfd9d05f8d0fd96c43*

Issues

The issues below have been raised over several 4 consecutive reviews, and appear from the most recent review (R4) to the oldest (R1).

B-R4.1. Solved Major There is a limit on the number of active guilds, however creating guilds and activating rewards is free, **thus a malicious user could spam the endpoints** `deploy_guild` **and** `resume_guild_endpoint` **to reach this limit, and prevent any legitimate Guild from being deployed again.**

Although it is desired to limit the number of guilds so that the amount of UTK rewards to distribute will be capped, it is expected that to create a guild the guild master would at least have to lock 500k UTK, so that spamming guild creation would practically be very expensive.

Solution: In `resume_guild_endpoint`, we suggest verifying that the guild master already made the initial stake in the guild. This way, guilds will only be added to the list of active guilds if the guild master has deposited the 500k UTK.

B-R4.2. Not Solved Minor The endpoint name `resume_guild_endpoint` is misleading because there is no state anymore in the guild, and the endpoint simply activates rewards distribution. Similarly, the name of the storage `active_guilds` is misleading.

Solution: We suggest renaming `resume_guild_endpoint` into `start_produce_rewards_in_guild_endpoint`, and renaming `active_guilds` into `guilds_producing_rewards`.

B-R3.1. Solved Minor `current_active_guilds` is an unnecessary storage. It is recording the length of the mapper `active_guilds`, thus we could directly use the length of `active_guilds` instead. In addition, having this separate storage requires carefully keeping it synchronized with each change of `active_guilds`.

In particular, this synchronization would break easily if other small code changes are made in the future. For example, if in `close_guild` we remove the requirement that non-empty payments are received, then it would be possible for a user to deploy a guild and close it even if it is not activated, leading to a decrease of `current_active_guilds` even if the guild was not active. In turn, we would have the following critical consequences:

- An attacker could decrease `current_active_guilds` until it underflow, to prevert new guilds from ever being activated again.
- Anyone could bypass the limitation on the number of active guilds.

Solution: We suggest deleting `current_active_guilds`, and directly use the length of `active_guilds` instead.

B-R2.1. Solved Medium `division_safety_constant` might be too small, e.g. 1, and that would induce high rounding errors, possibly making users earn no rewards from guilds. This is indeed a situation that occurred while interactions with the guilds were tested and integrated in the backend.

Solution: We suggest enforcing that `division_safety_constant` is bigger than 1e18.

B-R2.2. Not Solved Medium Issue B-R1.2 has been reintroduced in a weaker form, because spamming the guild creation now requires paying 500k UTK for each.

B-R1.1. Solved Major There is a maximal number of guilds `max_guilds` that can be deployed through the Guild Factory. Therefore **a malicious user could spam the guild creation endpoint to reach this limit, and prevent any legitimate Guild from being deployed again.**

Solution: We suggest removing the limit `max_guilds`, as it was deemed superfluous by the xMoney team.

B-R1.2. Solved Major **Guild migration will not work**, because the guild factory does not have the transfer role for farm tokens. Indeed, when a user migrates his funds from a guild A to another guild B, then the new farm position is sent from guild B to the Guild Factory, which forwards the position to the user. However this will fail as the Guild Factory has no transfer roles for farm tokens.

Solution: We suggest giving the transfer role to the Guild Factory. Alternatively, we can make the guild where the position is migrated forward the new position directly to the user instead of sending it back to the Guild Factory.

B-R1.3. Solved Medium The endpoint `remove_guild` won't ever be called successfully because:

- It requires that the total farm supply is 0 in the guild, but this is only possible if the guild master previously closed a guild, and,
- When the guild master closes a guild, the guild is removed from the list of known guilds. Therefore `remove_guild` would fail if called after that.

Solution: We suggest removing the endpoint `remove_guild`, or thinking of an alternative implementation which is both useful and safe, i.e. does not prevent users from withdrawing their funds.

B-R1.4. Solved Medium The project wants to not let guild master choose the token identifier and display name of the farm token and unbond token, to avoid undesired names. Rather, this information should be automatically determined by the smart contract.

Solution: We suggest having the ticker of the farm token and of the guild master hardcoded in the Guild Config. Similarly, a base display name can be hardcoded, and each time a new token is issued, we append an ID as a suffix to the base display name. This ID is incremented each time a new token is issued.

B-R1.5. Solved Medium In `remove_guild` and `remove_guild_common`, there is no check that the argument `user` is the guild master of the argument `guild`. Thus a mistake is possible, i.e. it could be possible to remove a guild and the wrong guild master.

Solution: We suggest checking that the arguments match, or alternatively just providing the `guild` argument and derive the guild master from it.

B-R1.6. Solved Medium The rewards per block (RPB) can't be changed, although it will necessarily have to be changed e.g. to accommodate for future changes of block duration. Indeed, when the block duration will be reduced from 6 seconds to 3 seconds, it would be needed to reduce the RPB by 2 in order to guarantee the same APR.

In particular, if a mistake is made when setting the initial RPB, then it would be irreversible. For example, if `per_block_reward_amount` is given 0 value, then no guilds can ever be activated, since `start_produce_rewards` requires that the RPB is positive.

Solution: We suggest adding an admin endpoint in the Guild Factory to change `per_block_reward_amount`, and another endpoint in the Guild to update the RPB used inside the Guild based on the current value of `per_block_reward_amount` in the Guild Factory.

B-R1.7. Solved Medium A Guild can be activated in `resume_guild_endpoint` even if the tiers and APRs have not been set in the Guild Config. This would then make all users' interactions fail even though they would think the Guilds are ready to accept deposits.

Solution: In `resume_guild_endpoint`, we suggest checking that the tiers and APRs have been set in the Guild Config.

B-R1.8. Solved Medium Admins are overpowered, as they can deploy the guild config smart contract and call any endpoint there, which can have a huge impact on all users. For example, if an admin is corrupted, he could briefly increase the APR to a huge value in the Guild Factory, and then aggregate rewards in a guild he created, to drain all the UTK available in the Guild Factory, and claim them, which will be irreversible.

Solution: We recommend letting only the owner call `deploy_config_sc` and `call_config_function`.

B-R1.9. Solved Medium The owner of Guild Factory has no ways to pause guilds if there is a problem, i.e. pausing all users actions except unstaking/unbonding. This is because there is no endpoint in the Guild Factory to call the `pause` endpoint of the Guild.

Solution: We suggest having a global pausing mechanism in the Guild Factory, which the owner can activate so as to pause all actions in guilds, except unstaking/unbonding.

Disclaimer