SHERLOCK

Security Review For Meteora

Collaborative Audit Prepared For: **Meteora**
Lead Security Expert(s): **bin2chen**
carrotsmuggler
Date Audited: **November 10 - November 24, 2025**

Introduction

A solana program for managing token presale. It supports:

- Split of presale supply sale into multiple user groups with it's own deposit caps per user
- Fixed price, prorata and first-come-first-serve mode
- Permissionless and permissioned mode via merkle tree or operator approval with partial sign
- Full, partial locking and vesting schedule on token sold

Scope

Repository: MeteoraAg/presale

Audited Commit: 50862dad540c07ca5f41ffca25e0d5bd7cab8246

Final Commit: 46e64b65d148f5a1207e531c39f4c8b3e9461831

Files:

- merkle-tree/src/config_merkle_tree.rs
- merkle-tree/src/lib.rs
- merkle-tree/src/merkle_tree.rs
- merkle-tree/src/tree_node.rs
- merkle-tree/src/utils.rs
- programs/presale/src/constants.rs
- programs/presale/src/const_pda.rs
- programs/presale/src/instructions/create_escrow/mod.rs
- programs/presale/src/instructions/create_escrow/process_create_escrow.rs
- programs/presale/src/instructions/create_escrow/process_create_permissioned_escrow_with_creator.rs
- programs/presale/src/instructions/create_escrow/process_create_permissioned_escrow_with_merkle_proof.rs
- programs/presale/src/instructions/create_escrow/process_create_permissionless_escrow.rs
- programs/presale/src/instructions/initialize_presale/mod.rs
- programs/presale/src/instructions/initialize_presale/params.rs
- programs/presale/src/instructions/initialize_presale/process_create_presale_vault.rs

- programs/presale/src/instructions/initialize_presale/process_initialize_presale.rs
- programs/presale/src/instructions/mod.rs
- programs/presale/src/instructions/process_claim.rs
- programs/presale/src/instructions/process_close_escrow.rs
- programs/presale/src/instructions/process_close_extra_presale_params.rs
- programs/presale/src/instructions/process_close_merkle_proof_metadata.rs
- programs/presale/src/instructions/process_create_merkle_root_config.rs
- programs/presale/src/instructions/process_create_operator.rs
- programs/presale/src/instructions/process_create_permissioned_server_-metadata.rs
- programs/presale/src/instructions/process_creator_collect_fee.rs
- programs/presale/src/instructions/process_creator_withdraw.rs
- programs/presale/src/instructions/process_deposit.rs
- programs/presale/src/instructions/process_initialize_extra_presale_params.rs
- programs/presale/src/instructions/process_perform_unsold_base_token_action.rs
- programs/presale/src/instructions/process_refresh_escrow.rs
- programs/presale/src/instructions/process_revoke_operator.rs
- programs/presale/src/instructions/process_withdraw_remaining_quote.rs
- programs/presale/src/instructions/process_withdraw.rs
- programs/presale/src/lib.rs
- programs/presale/src/macros.rs
- programs/presale/src/math/claim_math.rs
- programs/presale/src/math/fee_math.rs
- programs/presale/src/math/mod.rs
- programs/presale/src/math/safe_math.rs
- programs/presale/src/presale_mode_handler/fcfs_presale.rs
- programs/presale/src/presale_mode_handler/fixed_price_presale.rs
- programs/presale/src/presale_mode_handler/mod.rs
- programs/presale/src/presale_mode_handler/prorata_presale.rs
- programs/presale/src/state/escrow.rs
- programs/presale/src/state/fixed_price_presale_params.rs
- programs/presale/src/state/merkle_root_config.rs

- programs/presale/src/state/mod.rs
- programs/presale/src/state/operator.rs
- programs/presale/src/state/permissioned_server_metadata.rs
- programs/presale/src/state/presale_registry.rs
- programs/presale/src/state/presale.rs
- programs/presale/src/token2022.rs

Final Commit Hash

46e64b65d148f5a1207e531c39f4c8b3e9461831

Findings

Each issue has an assigned severity:

- High issues are directly exploitable security vulnerabilities that need to be fixed.
- Medium issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
- Low/Info issues are non-exploitable, informational findings that do not pose a security risk or impact the system's integrity. These issues are typically cosmetic or related to compliance requirements, and are not considered a priority for remediation.

Issues Found

High	Medium	Low/Info
0	3	0

Issues Not Fixed and Not Acknowledged

High	Medium	Low/Info
0	0	0

Issue M-1: Lack of Mechanism to Disable Older Versions of MerkleRootConfig [RESOLVED]

Source: https://github.com/sherlock-audit/2025-11-meteora-nov-10th/issues/53

Vulnerability Detail

The `create_merkle_root_config()` function can be used to generate multiple versions of a `config` for the same `presale`. However, there is currently no method to disable older configurations. As a result, if a configuration needs to be modified, the older configurations remain valid.

Example:

```
MerkleRootConfig_v1 = [root = hash [... .node:{alice, deposit_cap:100 }]]
```

Then it needs to be modified, so add v2:

```
MerkleRootConfig_v2 = [root = hash [... .node:{alice, deposit_cap:500 }]]
```

A malicious user can use the old configuration by calling `create_permissioned_escrow_with_merkle_proof(merkle_root_config = v1)`.

After that, `create_permissioned_escrow_with_merkle_proof(merkle_root_config = v2)` will fail due to seed conflicts.

Besides, there's no way to close it and no way to save rent.

Impact

Users can still use the old misconfiguration.

Code Snippet

https://github.com/sherlock-audit/2025-11-meteora-nov-10th/blob/9305f436ad11af840f2474ecaa6ec36755770434/presale/programs/presale/src/instructions/process_create_merkle_root_config.rs#L22

Tool Used

Manual Review

Recommendation

add method `close_merkle_root_config()` to turn off invalid configurations.

Discussion

bin2chen66

fixed in https://github.com/MeteoraAg/presale/pull/38 add `close_merkle_root_config()` method it restricts `progress != PresaleProgress::Ongoing`, meaning that once the presale begins, the configuration cannot be adjusted.It is recommended to note this in the UI or documentation so that users are aware of it.

Issue M-2: registry can consume other registry supply [RESOLVED]

Source: https://github.com/sherlock-audit/2025-11-meteora-nov-10th/issues/54

Vulnerability Detail

When we do `deposit()`, we use `presale_handler.get_remaining_deposit_quota()` to get the maximum amount that can be deposited.

`fixed_price_presale.get_remaining_deposit_quota()` is implemented as follows:

```
fn get_remaining_deposit_quota(&self, presale: &Presale, escrow: &Escrow) ->
↪  Result<u64> {
    let global_remaining_quota = presale.get_remaining_deposit_quota()?;
    let presale_registry =
    ↪  presale.get_presale_registry(escrow.registry_index.into())?;
    let personal_remaining_quota =
        escrow.get_remaining_deposit_quota(presale_registry.buyer_maximum_deposit_c
        ↪  ap)?;

    Ok(global_remaining_quota.min(personal_remaining_quota))
}
```

Currently it is only limited to `global_remaining_quota` and `personal_remaining_quota`, but there is no restriction on exceeding `registry.presale_supply`. This results in being able to consume other registry supply.

Example configuration: q_price = 1 , presale_maximum_cap = 200 registry_1 = { presale_supply = 100 } registry_2 = { presale_supply = 100 }

The final purchase may become:

registry_1 = { presale_supply = 100 , total_deposit = 200 } => can get supply = 200
registry_2 = { presale_supply = 100 , total_deposit = 0 } => only get supply = 0

Impact

registry can consume other registry supply

Code Snippet

https://github.com/sherlock-audit/2025-11-meteora-nov-10th/blob/9305f436ad11af840f2474ecaa6ec36755770434/presale/programs/presale/src/presale_mode_handler/fixed_price_presale.rs#L183

Tool Used

Manual Review

Recommendation

registry sold token cannot exceed `registry.presale_supply`

Discussion

bin2chen66

fixed in https://github.com/MeteoraAg/presale/pull/39

Three adjustments were made:

1. `get_remaining_deposit_quota()`
    - add `registry_remaining_deposit_quota`
    - note: Because of the round, this is possible : `registry.total_deposit / q_price > registry.presale_supply`
2. `get_total_base_token_sold()`
    - change `sold_token = min(presale_registry.total_deposit/q_price ,registry.presale_supply)`
3. `get_escrow_cumulative_claimable_token()`
    - change `total_sold_token = min( presale_registry.total_deposit/q_price , registry.presale_supply)`

Example:

presale.presale_supply = 300, presale_maximum_cap = 100,q_price = 0.333 registry_1 = {presale_supply = 100} registry_2 = {presale_supply = 100} registry_3 = {presale_supply = 100}

Final Purchase: registry_1 = {presale_supply = 100 , total_deposit = 34} => sold_token = 34 / 0.333 = 102 (Slightly greater than 100)
registry_2 = {presale_supply = 100 , total_deposit = 34} => sold_token = 34 / 0.333 = 102 (Slightly greater than 100)
registry_3 = {presale_supply = 100 , total_deposit = 32} => sold_token = 32 / 0.333 = 96 (because presale_maximum_cap,total_deposit can only be up to 32)

Refund: total_refund_amount = 300 - min(102,100) - min(102,100) - min(96,100) = 300 - 100 - 100 - 96 = 4 real_price = 100 / 296 = 0.337 (Slightly greater than 0.333)
registry_1_real_price = 34 / 100 = 0.34 (Slightly greater than 0.333) registry_2_real_price = 34 / 100 = 0.34 (Slightly greater than 0.333) registry_3_real_price = 32 / 96 = 0.333

So: more refunds and less `claimable_token`? Should we consider removing the `get_total_base_token_sold()/get_escrow_cumulative_ claimable_token()` changes without `min(sold_token,registry.presale_supply)`?

registry Slightly exceeding `registry.presale_supply` is acceptable. but Since `presale_maximum_cap` exists, The `total_sold_token` accumulated by all registries does not exceed the `presale.presale_supply`.

@codewithgun If you have time, please take a look. change it or still use PR 39? (There will be only minor differences).

codewithgun

fixed in MeteoraAg/presale#39
Three adjustments were made:

```
1. `get_remaining_deposit_quota()`

   * add `registry_remaining_deposit_quota`
   * note: Because of the round, this is possible : `registry.total_deposit /
   ↪  q_price > registry.presale_supply`

2. `get_total_base_token_sold()`

   * change `sold_token = min(presale_registry.total_deposit/q_price
   ↪  ,registry.presale_supply)`

3. `get_escrow_cumulative_claimable_token()`

   * change  `total_sold_token = min( presale_registry.total_deposit/q_price
   ↪  ,registry.presale_supply)`
```

Example:
presale.presale_supply = 300, presale_maximum_cap = 100,q_price = 0.333 registry_1 = {presale_supply = 100} registry_2 = {presale_supply = 100} registry_3 = {presale_supply = 100}
Final Purchase: registry_1 = {presale_supply = 100 , total_deposit = 34} => sold_token = 34 / 0.333 = 102 (Slightly greater than 100) registry_2 = {presale_supply = 100 , total_deposit = 34} => sold_token = 34 / 0.333 = 102 (Slightly greater than 100) registry_3 = {presale_supply = 100 , total_deposit = 32} => sold_token = 32 / 0.333 = 96 (because presale_maximum_cap,total_deposit can only be up to 32)
Refund: total_refund_amount = 300 - min(102,100) - min(102,100) - min(96,100) = 300 - 100 - 100 - 96 = 4 real_price = 100 / 296 = 0.337 (Slightly greater than 0.333) registry_1_real_price = 34 / 100 = 0.34 (Slightly greater than 0.333) registry_2_real_price = 34 / 100 = 0.34 (Slightly greater than 0.333) registry_3_real_price = 32 / 96 = 0.333
So: more refunds and less `claimable_token`? Should we consider removing the `get_total_base_token_sold()/get_escrow_cumulative_ claimable_token()` changes without `min(sold_token,registry.presale_supply)`?
registry Slightly exceeding `registry.presale_supply` is acceptable. but Since `presale_maximum_cap` exists, The `total_sold_token` accumulated by all registries does not

> exceed the `presale.presale_supply`.
> @codewithgun If you have time, please take a look. change it or still use PR 39? (There will be only minor differences).

Hi, @bin2chen66, we will keep using the fix in PR 39

Issue M-3: fixed price presale may unable to complete [RESOLVED]

Source: https://github.com/sherlock-audit/2025-11-meteora-nov-10th/issues/55

Vulnerability Detail

When we call `deposit()`, the protocol does `presale_handler.suggest_deposit_amount(amount)` on the deposited amount

if using `FixedPrice`, `suggested_deposit_amount()` will removes the excess amount, which may cause `presale.total_deposit` to never equal `presale.presale_maximum_cap`.

Example: presale_maximum_cap = 100 , q_price = 9.

1. alice deposit(90)
    - total_deposit = 90
2. bob deposit(10) , then `suggested_deposit_amount = 9`
    - total_deposit = 99

After that , `total_deposit` can only be maximized to `99` , not `100`.

With `presale_minimum_cap == presale_maximum_cap`, even `PresaleProgress::Completed` will never be reached.

Impact

`end_presale_if_max_cap_reached()` cannot be executed properly, it needs to wait for the long `presale_end_time`. when `presale_minimum_cap == presale_maximum_cap` , presale may unable to complete

Code Snippet

https://github.com/sherlock-audit/2025-11-meteora-nov-10th/blob/9305f436ad11af840f2474ecaa6ec36755770434/presale/programs/presale/src/presale_mode_handler/fixed_price_presale.rs#L283

Tool Used

Manual Review

Recommendation

suggest_deposit_amount() on presale_maximum_cap.

Discussion

bin2chen66

fixed in https://github.com/MeteoraAg/presale/pull/40

add `ensure_gap_between_min_and_max_presale_cap()` There's enough gap to reach`PresaleProgress::Completed`

However, it may still fail to execute `end_presale_if_max_cap_reached()` properly.

Example: Purchase 1M, but may not be able to end early because of 1 token, need to wait for 3 months (presale_end_time)

This can be quite annoying, so it is recommended that `disable_earlier_presale_end_once_cap_reached = false` and `presale_maximum_cap` has an excess amount, to give a hint to the user in the UI or documentation . the user is aware of it.

codewithgun

Hi @bin2chen66, we'll acknowledge this.

Disclaimers

Sherlock does not provide guarantees nor warranties relating to the security of the project.

Usage of all smart contract software is at the respective users' sole risk and is the users' responsibility.