OFFSIDE LABS

Meteora Presale

Smart Contract Security Assessment

October 2025

Prepared for:

Meteora

Prepared by:

Offside Labs

Sirius Xie

Zhoujie Tong

Contents

1 About Offside Labs

Offside Labs is a leading security research team, composed of top talented hackers from both academia and industry.

We possess a wide range of expertise in modern software systems, including, but not limited to, *browsers*, *operating systems*, *IoT devices*, and *hypervisors*. We are also at the forefront of innovative areas like *cryptocurrencies* and *blockchain technologies*. Among our notable accomplishments are remote jailbreaks of devices such as the **iPhone** and **PlayStation 4**, and addressing critical vulnerabilities in the **Tron Network**.

Our team actively engages with and contributes to the security community. Having won and also co-organized *DEFCON CTF*, the most famous CTF competition in the Web2 era, we also triumphed in the **Paradigm CTF 2023** within the Web3 space. In addition, our efforts in responsibly disclosing numerous vulnerabilities to leading tech companies, such as *Apple*, *Google*, and *Microsoft*, have protected digital assets valued at over **$300 million**.

In the transition towards Web3, Offside Labs has achieved remarkable success. We have earned over **$9 million** in bug bounties, and **three** of our innovative techniques were recognized among the **top 10 blockchain hacking techniques of 2022** by the Web3 security community.

 **https://offside.io/**

 **https://github.com/offsidelabs**

 **https://twitter.com/offside_labs**

2 Executive Summary

Introduction

Offside Labs completed a security audit of *Meteora Presale* smart contracts, starting on September 3rd, 2025, and concluding on September 18th, 2025.

Project Overview

This project is a presale platform designed to support new token launches with flexible and secure presale mechanisms. It allows creators to run presales in multiple formats, giving full control over how tokens are sold and how unsold tokens are handled.

The platform supports both permissionless and permissioned presales, enabling either open participation or curated access. Its modular and decentralized design makes it easy for any project to launch a fair and transparent token sale on-chain.

Audit Scope

The assessment scope contains mainly the smart contracts of the presale program for the *Meteora Presale* project.

The audit is based on the following specific branches and commit hashes of the codebase repositories:

- Meteora Presale
  - Codebase: `https://github.com/MeteoraAg/presale`
  - Commit Hash: 3b6a02f3ad7e55e00493637ee969fe9e37797724
  - Branch: dev

We listed the files we have audited below:

- Meteora Presale
  - programs/presale/src/const_pda.rs
  - programs/presale/src/constants.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/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/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.rs
- programs/presale/src/instructions/process_withdraw_remaining_quote.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/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/operator.rs
- programs/presale/src/state/permissioned_server_metadata.rs
- programs/presale/src/state/presale.rs
- programs/presale/src/state/presale_registry.rs
- programs/presale/src/token2022.rs

Findings

The security audit revealed:

- 0 critical issue
- 0 high issue
- 4 medium issues
- 4 low issues
- 3 informational issues

Further details, including the nature of these issues and recommendations for their remediation, are detailed in the subsequent sections of this report.

3 Summary of Findings

ID	Title	Severity	Status
01	Presale Progress Can Be Manipulated in FixedPrice Mode	Medium	Fixed
02	User Cannot Close Escrow on Failed Presale	Medium	Fixed
03	Missing Validate Total Deposit Fee when Closing Escrow during Ongoing Stage	Medium	Fixed
04	Missing Mut On Owner Base Token In Claim IX	Medium	Fixed
05	Possible DoS Due to Missing vest_duration Validation	Low	Fixed
06	immediately_release_bps Cannot be Set to MAX_FEE_BASIS_POINTS in intialize_presale IX	Low	Fixed
07	Users May Be Unable to Withdraw Remaining Quote in FixedPrice Mode	Low	Fixed
08	Missing Buyer Cap Check in withdraw IX	Low	Fixed
09	Possible DoS on Claim via Malicious Permissionless refresh_escrow IX	Informational	Fixed
10	Overly Restrictive Stage Check in close_permissioned_server_metadata	Informational	Fixed
11	Reminder About Precision Residues	Informational	Acknowledged

4 Key Findings and Recommendations

4.1 Presale Progress Can Be Manipulated in FixedPrice Mode

Severity: Medium	Status: Fixed
Target: Smart Contract	Category: Logic Error

Description

When `presale.presale_mode` is `FixedPrice`, in the `deposit` IX, `presale_handler.end_presale_if_max_cap_reached` function will check whether `presale.total_deposit` has already met the requirement for completed.

```
138     if presale.total_deposit >= presale.presale_maximum_cap {
139         presale.advance_progress_to_completed(current_timestamp)?;
140     }
```

programs/presale/src/presale_mode_handler/fixed_price_presale.rs#L138-L140

In `advance_progress_to_completed`, all related time points are updated.

```
297     self.presale_end_time = new_presale_end_time;
298
299     self.lock_start_time = self.presale_end_time.safe_add(1)?;
300     self.lock_end_time =
        ↪ self.lock_start_time.safe_add(self.lock_duration)?;
301
302     self.vesting_start_time = self.lock_end_time.safe_add(1)?;
303     self.vesting_end_time =
        ↪ self.vesting_start_time.safe_add(self.vest_duration)?;
```

programs/presale/src/state/presale.rs#L297-L303

However, if a malicious user appends a `withdraw` IX to the same tx after the `deposit` IX, they can use that `withdraw` IX to withdraw an appropriate amount of quote mint, thereby manipulating whether the final presale result is completed or failed, because at that moment the `withdraw` still sees the presale as ongoing.

```
268     pub fn get_presale_progress(&self, current_timestamp: u64) ->
        ↪ PresaleProgress {
269         if current_timestamp < self.presale_start_time {
270             return PresaleProgress::NotStarted;
271         } else if current_timestamp <= self.presale_end_time {
272             return PresaleProgress::Ongoing;
273         }
```

programs/presale/src/state/presale.rs#L268-L273

Impact

If the malicious user's escrow satisfies `escrow.total_deposit ≥ (presale_maximum_cap - presale_minimum_cap)`, the attacker can tune the `withdraw` amount so that, after the `withdraw` IX finishes, `presale.total_deposit` changes from completed to failed.

Recommendation

It is recommended to adjust the `Ongoing` determination rules to prevent other IXs that conform to the Ongoing progress from succeeding in the same block after the `deposit` IX/TX.

Mitigation Review Log

Fixed in commit 77315b4c7038e5efe1d723232171b4897f6eb373.

4.2 User Cannot Close Escrow on Failed Presale

Severity: Medium	Status: Fixed
Target: Smart Contract	Category: Logic Error

Description

In the `close_escrow` IX, when `presale_progress` is Failed, it requires `escrow.total_deposit == 0`.

```
109 fn ensure_escrow_no_deposit(escrow: &Escrow) -> Result<()> {
110     require!(escrow.total_deposit == 0, PresaleError::EscrowNotEmpty);
111     Ok(())
112 }
```

programs/presale/src/instructions/process_close_escrow.rs#L109-L112

However, before this, when a user withdraws the quote mint they deposited via the `withdraw_remaining_quote` IX, `escrow.total_deposit` is not cleared to zero, preventing the user from meeting the "empty position" condition and closing the Escrow.

Impact

When `presale_progress` is failed, no users can close their own escrow and reclaim the rent associated with the Escrow.

Recommendation

It is recommended that:

- Relax the requirement in `close_escrow` IX for a Failed presale so that `escrow.is_remaining_quote_withdrawn == true || escrow.total_deposit == 0` is sufficient to close the escrow.
- Alternatively, when `presale_progress` is failed, after `withdraw_remaining_quote` succeeds, set `escrow.total_deposit` and `escrow.total_deposit_fee` to zero.

Mitigation Review Log

Fixed in commit 50e7f178bff3d21301e96579fd0f97f4eb40cc94.

4.3 Missing Validate Total Deposit Fee when Closing Escrow during Ongoing Stage

Severity: Medium	Status: Fixed
Target: Smart Contract	Category: Logic Error

Description

In the `close_escrow` IX, when `total_deposit` is zero but `total_deposit_fee` is not zero (e.g., after a deposit followed by a full withdrawal), the escrow can still be closed. However, the `total_deposit_fee` stored in the escrow is ignored during the closing process. If the presale later fails, these fees remain locked in the escrow and are lost permanently.

Impact

Closing an escrow with non-zero `total_deposit_fee` while the presale is still ongoing results in permanent loss of collected fees, since they are never claimed or redistributed.

Recommendation

Add a validation in the `close_escrow` IX by rejecting closure of a user escrow when the presale is ongoing, or ensuring that `total_deposit_fee` is zero (indicating the user has not deposited yet), or alternatively handling the remaining fee correctly.

Mitigation Review Log

Fixed in commit c53408d150c3fed8c335ee0eb085415d14cc4981.

4.4 Missing Mut On Owner Base Token In Claim IX

Severity: Medium	Status: Fixed
Target: Smart Contract	Category: DoS Risk

Description

In the `claim` IX, the `owner_base_token` account, which serves as the destination token account, is not marked as `mut` . As a result, the generated IDL does not indicate this account as writable. When users attempt to call this instruction, the runtime rejects the transaction because the program tries to modify a non-writable account, causing the claim to fail. It is worth noting that in testing this issue did not appear, since claim and the creation of the token account were included in the same transaction, which allowed the instruction to appear to work correctly.

Impact

When a user calls the `claim` IX alone without creating the token account in the same transaction, the transaction will fail. This prevents successful claims in real usage scenarios where the token account already exists.

Recommendation

Add the `mut` attribute to the `owner_base_token` account in the `claim` IX, ensuring that the token account is properly marked as writable in both the program and the IDL.

Mitigation Review Log

Fixed in commit 0fc57919541fc120ebe552c98abd71cf93d4e8ec.

4.5 Possible DoS Due to Missing vest_duration Validation

Severity: Low	Status: Fixed
Target: Smart Contract	Category: Data Validation

Description

In the `initialize_presale` IX, `LockedVestingArgs.vest_duration == 0` is allowed.

In the `refresh_escrow` and `claim` IXs, `calculate_dripped_amount_for_user` directly uses `LockedVestingArgs.vest_duration` as the divisor in its calculation.

```
55      let dripped_total_sold_token = u128::from(vested_amount)
56          .safe_mul(elapsed_seconds.into())?
57          .safe_div(vest_duration.into())?;
```

programs/presale/src/math/claim_math.rs#L55-L57

If `vest_duration == 0`, this will result in a division-by-zero error and cause the IX to fail.

Impact

A `presale.vest_duration` of 0 may prevent users from being able to `claim` and `close_escrow`.

Recommendation

It is recommended to handle the case where `presale.vest_duration == 0` separately in the calculation, or to improve the checks on presale configuration parameters in `LockedVestingArgs.validate`.

Mitigation Review Log

Fixed in commit 9db7341ea7c60ba7630c71caaadbd4a343b753b0.

4.6 immediately_release_bps Cannot be Set to MAX_FEE_BASIS_POINTS in intialize_presale IX

Severity: Low	Status: Fixed
Target: Smart Contract	Category: Logic Error

Description

In the `initialize_presale` IX, `LockedVestingArgs.validate` checks the value of `LockedVestingArgs.immediately_release_bps`.

```rust
    if self.immediately_release_bps == MAX_FEE_BASIS_POINTS {
        require!(
            self.lock_duration == 0 && self.vest_duration == 0,
            PresaleError::InvalidLockVestingInfo
        );
    }

    // Portion of token is immediately release, another part must be
    ↪ vested else it's have same effect as immediate release
    if self.immediately_release_bps > 0 {
        require!(
            self.lock_duration > 0 || self.vest_duration > 0,
            PresaleError::InvalidLockVestingInfo
        );
    }
```

programs/presale/src/instructions/initialize_presale/params.rs#L201-L214

However, when `immediately_release_bps == MAX_FEE_BASIS_POINTS`, execution enters both of the above `if` branches, and the requirements for `lock_duration` and `vest_duration` in those branches conflict with each other.

Impact

This issue causes the `initialize_presale` IX to fail with `PresaleError::InvalidLockVestingInfo`, making it impossible to support releasing all of the base mint immediately.

Recommendation

It is recommended that in `LockedVestingArgs.validate`, within the first `if` branch related to `immediately_release_bps`, the function return immediately once the condition is satisfied.

Mitigation Review Log

Fixed in commit 9db7341ea7c60ba7630c71caaadbd4a343b753b0.

4.7 Users May Be Unable to Withdraw Remaining Quote in FixedPrice Mode

Severity: Low	Status: Fixed
Target: Smart Contract	Category: Logic Error

Description

In FixedPrice mode, if either the global value or a given PresaleRegistry's `total_deposit` stays below the price of one unit of base (`q_price`), the computed `sold_amount` will be floored to 0.

```rust
fn calculate_token_bought(q_price: u128, amount: u64) -> Result<u128> {
    let q_amount = u128::from(amount).safe_shl(SCALE_OFFSET)?;
    let token_bought = q_amount.safe_div(q_price)?;

    Ok(token_bought)
}
```

programs/presale/src/presale_mode_handler/fixed_price_presale.rs#L4-L9

With multiple presale registries, the presale may still reach the completed stage by hitting `presale_minimum_cap` .

Impact

In that case, the following issues arise:

- Users cannot receive any base from that PresaleRegistry because the calculated claim amount is 0.
- In FixedPrice mode, users are not allowed to invoke the `withdraw_remaining_quote` IX on a Completed presale, so they cannot withdraw the quote mint they deposited.
- After Completed, the creator can withdraw all quote (up to `presale_maximum_cap`) via the `creator_withdraw` IX.

Recommendation

For FixedPrice presale initialization, require that `buyer_minimum_deposit_cap` corresponds to a base amount > 0. Also, after any deposit/withdraw, when `escrow.total_deposit` remains > 0, ensure the final escrow balance is sufficient to claim at least 1 unit of base.

Mitigation Review Log

Fixed in commit 8a86e8a62be83ec6028437d96771b59a3b7ba6b8.

4.8 Missing Buyer Cap Check in withdraw IX

Severity: Low	Status: Fixed
Target: Smart Contract	Category: Logic Error

Description

In the deposit IX, here is the validation on post `escrow.total_deposit` .

```
64      require!(
65          escrow.total_deposit >=
    ↪ presale_registry.buyer_minimum_deposit_cap
66              && escrow.total_deposit <=
    ↪ presale_registry.buyer_maximum_deposit_cap,
67          PresaleError::DepositAmountOutOfCap
68      );
```

programs/presale/src/instructions/process_deposit.rs#L64-L68

However, in the withdraw IX, it allows withdrawing any amount within the range `(0, escrow.total_deposit]` , and after updating `escrow.total_deposit` , there is no validation on `escrow.total_deposit` the same as in the deposit IX.

Impact

By using withdraw, a user can bypass the constraint `escrow.total_deposit >= presale_registry.buyer_minimum_deposit_cap` .

Recommendation

It is recommended that, before the end of the withdraw IX, the same check on `escrow.total_deposit` as in the deposit IX be added, with the additional allowance that `escrow.total_deposit` may reach zero.

Mitigation Review Log

Fixed in commit 72d47c272795f8e602a0c74461211aae4a4a526c.

4.9 Informational and Undetermined Issues

Possible DoS on Claim via Malicious Permissionless refresh_escrow IX

Severity: Informational	Status: Fixed
Target: Smart Contract	Category: Logic Error

The claim IX requires `escrow.last_refreshed_at == current_time` . The `escrow.last_refreshed_at` field is updated in the `refresh_escrow` IX, and `refresh_escrow` IX is permissionless. An attacker can preempt a user's claim by invoking `refresh_escrow` IX on the user's escrow in the slot before the user's claim, causing the user's standalone claim to fail. It's recommended to adjust the permission design for `refresh_escrow` .

Fixed in commit 99c7658d084a50d4a5cafc9338e3ee4ed940e664.

Overly Restrictive Stage Check in close_permissioned_server_metadata

Severity: Informational	Status: Fixed
Target: Smart Contract	Category: Code QA

In the `close_permissioned_server_metadata` IX, the presale is required to be in either not_start or ongoing stage. However, it is acceptable and should be allowed to close the metadata across all stages.

Fixed in commit e79648ed8d823e4f10311205ade366677cf646f9.

Reminder About Precision Residues

Severity: Informational	Status: Acknowledged
Target: Smart Contract	Category: Code QA

Because `claim_amount` and `remaining_quote` are both calculated by floor rounding according to the ratio of `escrow_total_deposit` to `total_deposit`, there will inevitably be precision residues left in the vault.The remaining amounts of quote and base tokens might approximately equal the number of participating escrows.

5 Disclaimer

This audit report is provided for informational purposes only and is not intended to be used as investment advice. While we strive to thoroughly review and analyze the smart contracts in question, we must clarify that our services do not encompass an exhaustive security examination. Our audit aims to identify potential security vulnerabilities to the best of our ability, but it does not serve as a guarantee that the smart contracts are completely free from security risks.

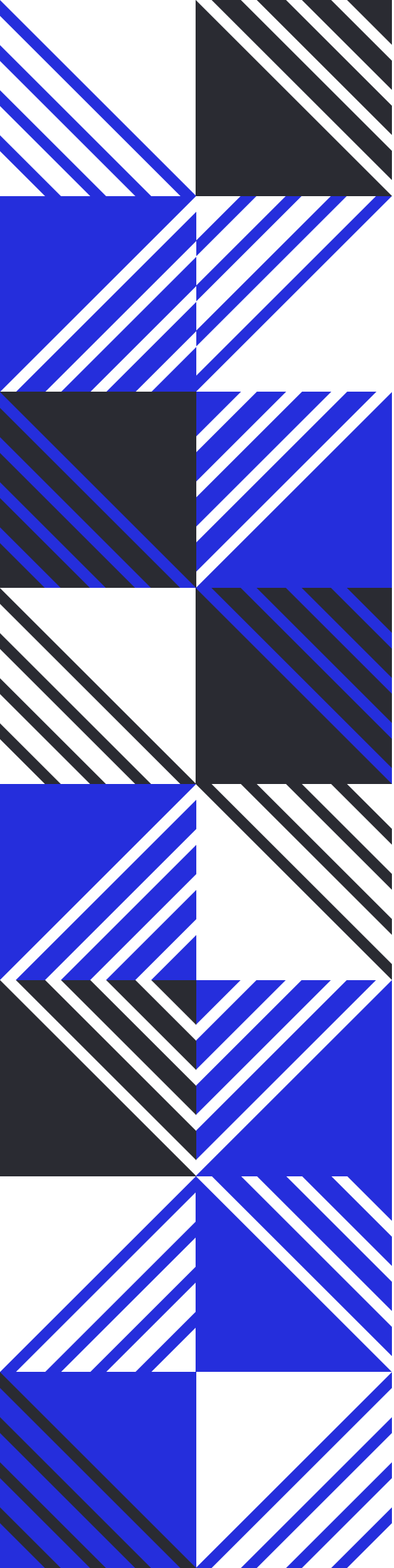
We expressly disclaim any liability for any losses or damages arising from the use of this report or from any security breaches that may occur in the future. We also recommend that our clients engage in multiple independent audits and establish a public bug bounty program as additional measures to bolster the security of their smart contracts.

It is important to note that the scope of our audit is limited to the areas outlined within our engagement and does not include every possible risk or vulnerability. Continuous security practices, including regular audits and monitoring, are essential for maintaining the security of smart contracts over time.

Please note: we are not liable for any security issues stemming from developer errors or misconfigurations at the time of contract deployment; we do not assume responsibility for any centralized governance risks within the project; we are not accountable for any impact on the project's security or availability due to significant damage to the underlying blockchain infrastructure.

By using this report, the client acknowledges the inherent limitations of the audit process and agrees that our firm shall not be held liable for any incidents that may occur subsequent to our engagement.

This report is considered null and void if the report (or any portion thereof) is altered in any manner.



OFFSIDE LABS

https://offside.io/

https://github.com/offsidelabs

https://twitter.com/offside_labs