OFFSIDE LABS

Dynamic Bonding Curve

Smart Contract Security Assessment

April 2025

Prepared for:

Meteora

Prepared by:

Offside Labs

Ronny Xing

Sirius Xie

Contents

1 About Offside Labs

Offside Labs stands as a pre-eminent security research team, comprising highly skilled hackers with top - tier talent from both academia and industry.

The team demonstrates extensive and diverse expertise in modern software systems, which encompasses yet are not restricted to *browsers*, *operating systems*, *IoT devices*, and *hypervisors*. Offside Labs is at the forefront of innovative domains such as *cryptocurrencies* and *blockchain technologies*. The team achieved notable accomplishments including the successful execution of remote jailbreaks on devices like the **iPhone** and **PlayStation 4**, as well as the identification and resolution of critical vulnerabilities within the **Tron Network**.

Offside Labs actively involves in and keeps contributing to the security community. The team was the winner and co-organizer for the *DEFCON CTF*, the most renowned CTF competition in Web2. The team also triumphed in the **Paradigm CTF 2023** in Web3. Meanwhile, the team has been conducting responsible disclosure of numerous vulnerabilities to leading technology companies, including *Apple*, *Google*, and *Microsoft*, safeguarding digital assets with an estimated value exceeding **$300 million**.

During the transition to Web3, Offside Labs has attained remarkable success. The team has earned over **$9 million** in bug bounties, and **three** of its innovative techniques were acknowledged as being among the **top 10 blockchain hacking techniques of 2022** by the Web3 security community.

2 Executive Summary

Introduction

Offside Labs completed a security audit of *Dynamic Bonding Curve* smart contracts, starting on March 25, 2025, and concluding on April 27, 2025.

Project Overview

The Dynamic Bonding Curve (DBC) is a permissionless launch pool protocol that enables users to create and launch tokens with customizable bonding curves directly. The program allows partners to configure key parameters such as quote tokens, token graduation curves, and fees. It supports transfers on both SPL Token and Token-2022 standards and introduces Dynamic Fees. Tokens can graduate to supported automated market makers (currently Meteora DAMM v1 and v2), enabling launchers to claim fees from locked liquidity pools.

Audit Scope

The assessment scope contains mainly the smart contracts of the dynamic-bonding-curve program for the *Dynamic Bonding Curve* project.

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

- Dynamic Bonding Curve
  - Codebase Link: `https://github.com/MeteoraAg/dynamic-bonding-curve`
  - Branch: remove_quote_constraints
    - Commit Hash: 11483e02250fa6686e1f9d7b092aff3c1dce1251
  - PR-21:
    - Commit Hash: b5823372081041bc55a1a2cb2147d040be084e9e
    - Codebase Link: `PR-21`
  - PR-23:
    - Commit Hash: fa45aa14c7cd2b0f7a10303f16a3a0c38ed6fdc6
    - Codebase Link: `PR-23`
  - PR-26:
    - Commit Hash: 04a5716eca53de2ad89f769d83a98df0bdcf6a56
    - Codebase Link: `PR-26`
  - PR-28:
    - Commit Hash: 81f35ce39584200ba474d17d058e5bddcb9109b9
    - Codebase Link: `PR-28`
  - PR-29:
    - Commit Hash: 8861e052f5e1df13d3dd8777b2d1110017741745
    - Codebase Link: `PR-29`
  - PR-32:
    - Commit Hash: f63c0b3410e7dacbf6743d699f9c92621bb3c65f
    - Codebase Link: `PR-32`

- PR-34
  - Commit Hash: f57c116157129687cbe37fed15c87f577bef6c79
  - Codebase Link: PR-34
- PR-40
  - Commit Hash: 56ba1668578891c42e928b32ba6d87430003fb8d
  - Codebase Link: PR-40
- PR-52
  - Commit Hash: 821ffb3504048a0282f5e9f55db1e22d92a8638d
  - Codebase Link: PR-52
- PR-53
  - Commit Hash: 186922825468d079018edddb4ff00323ee1e02e6
  - Codebase Link: PR-53
- PR-56
  - Commit Hash: c36419ec34cd8078b520032e10920c62b21f33cb
  - Codebase Link: PR-56
- PR-57
  - Commit Hash: 2c01ccc523be591f8649d99198afd913b5f662c9
  - Codebase Link: PR-57
- PR-58
  - Commit Hash: 8434c2cd7cd6e3d373a99b16178cf51bf7455ab3
  - Codebase Link: PR-58
- PR-59
  - Commit Hash: 264f044d81fde3fc6636c1dc194654cada3f39c1
  - Codebase Link: PR-59
- PR-60
  - Commit Hash: e76e7a64cce36043f223d7da456a2989fa5b5eb9
  - Codebase Link: PR-60
- PR-61
  - Commit Hash: d41276a26138f5eb8cb59b3b4f4459c361ce0773
  - Codebase Link: PR-61
- PR-63
  - Commit Hash: a6eb5863405fd06b97eb6b22ef51249c48028025
  - Codebase Link: PR-63
- PR-64
  - Commit Hash: bb510d30c1fe414dfd596e193692c42b7d18843a
  - Codebase Link: PR-64
- PR-65
  - Commit Hash: fa4642dc623a68b9facabd8f78b071a2b9671487
  - Codebase Link: PR-65
- PR-66
  - Commit Hash: 9ade9c1cdf71df66442a9b505e3702d267d5d844
  - Codebase Link: PR-66

We listed the files we have audited below:

- Dynamic Bonding Curve
    - programs/dynamic-bonding-curve/src/**/*.rs

Findings

The security audit revealed:

- 1 critical issues
- 0 high issue
- 5 medium issues
- 2 low issues
- 7 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	Unclaimed Fees May Be Burned in migrate_meteora_damm	Critical	Fixed
02	Large Final Swap May Prevent base_mint from Graduating	Medium	Fixed
03	Missing Length Check on curve Parameter in create_config	Medium	Fixed
04	Volatility Could Be Largely Affected by Absolute Price Changes	Medium	Fixed
05	Possible DoS on Migration or Claim Fee When Migrating to DAMM v2	Medium	Fixed
06	create_config IX Fails When BaseFeeConfig.period_frequency > 0	Medium	Fixed
07	Reduced MAX_CURVE_POINT May Prevent Migration for Existing Virtual Pools	Low	Fixed
08	Impact of Fee Collection Mode	Low	Acknowledged
09	Redundant IXs	Informational	Fixed
10	CU Optimization	Informational	Fixed
11	Inconsistent Rounding Direction When Calculating Base Token Amount	Informational	Acknowledged
12	Missing Validation on pool_fees.protocol_trade_fee_numerator	Informational	Acknowledged
13	Missing Memo Support for Token-2022 Transfers in transfer_from_pool	Informational	Acknowledged
14	Rounding Direction When Calculating Base Token Amount	Informational	Acknowledged
15	Pools With Specific Config Can Not Be Migrated	Informational	Acknowledged

4 Key Findings and Recommendations

4.1 Unclaimed Fees May Be Burned in migrate_ meteora_damm

Severity: Critical	**Status: Fixed**
Target: Smart Contract	**Category: Logic Error**

Description

In the `migrate_meteora_damm` IX, when the `base_mint` is migrated to the dynamic AMM, any excess `base_mint` in the `base_vault` is burned.

```
    // burn the rest of token in pool authority
    let left_base_token =
    ↪   ctx.accounts.base_vault.amount.safe_sub(base_reserve)?;
    if left_base_token > 0 {
        let seeds = pool_authority_seeds!(ctx.bumps.pool_authority);
        anchor_spl::token::burn(
            CpiContext::new_with_signer(...),
            left_base_token,
        )?;
    }
```

programs/virtual-curve/src/instructions/migration/meteora_damm/migrate_-meteora_damm_initialize_pool.rs#L260-L276

However, during the burn process, the unclaimed protocol fee and partner fee are not accounted for, which may result in these fees being burned along with the excess base_mint.

Impact

As a result, `migrate_meteora_damm` IX directly causes the unclaimed protocol fee and partner fee in `base_vault` to be burned, leading to a loss for the fee claimer. Additionally, any claim attempts made after migration will fail, as the required fees can no longer be withdrawn from `base_vault`, causing `claim_protocol_fee` and `claim_trading_fee` IXs to fail.

Recommendation

It is recommended to adjust the calculation of `left_base_token` to properly exclude the portion of `base_mint` allocated for the protocol fee and partner fee before performing the burn.

Mitigation Review Log

Fixed in the commit 687af178903b78985a79572c32e1d714995ec9bc.

4.2 Large Final Swap May Prevent base_mint from Graduating

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

Description

In the last swap before the curve completes, if a user swaps out a large amount of `base_mint`, it may leave the remaining `base_mint` balance in `base_vault` lower than `PoolConfig.migration_base_threshold`.

Impact

In this case, while the curve can still complete, the `migrate_meteora_damm` IX will fail during the creation of the dynamic AMM pool due to insufficient `base_mint` in `base_vault`, preventing it from successfully graduating.

Recommendation

It is recommended to impose a restriction on the token amount in the final swap before curve completion to mitigate this issue.

Mitigation Review Log

Fixed in the commit 687af178903b78985a79572c32e1d714995ec9bc and d3674b100f26ec3da900930e71fb48222d719d5b.

4.3 Missing Length Check on curve Parameter in create_config

Severity: Medium	**Status: Fixed**
Target: Smart Contract	**Category: Data Validation**

Description

In `create_config`, the curve points are passed in through `ConfigParameters.curve`. After validating `ConfigParameters.curve`, these points are stored in `PoolConfig.curve` via `PoolConfig::init`.

`PoolConfig::init` limits the number of curve points to a maximum of `MAX_CURVE_POINT` (20) and automatically ignores any points beyond the 20th entry in `ConfigParameters.curve`.

```
        let curve_length = curve.len();
        for i in 0..MAX_CURVE_POINT {
            if i < curve_length {
                self.curve[i] =
                 ↪  curve[i].to_liquidity_distribution_config();
            } else {
                self.curve[i] = LiquidityDistributionConfig {
                    sqrt_price: MAX_SQRT_PRICE, // set max
                    liquidity: 0,
                }
            }
        }
```

programs/virtual-curve/src/state/config.rs#L282-L292

However, some other fields in `PoolConfig`, such as `PoolConfig.swap_base_amount` and `PoolConfig.migration_base_amount`, are derived from `ConfigParameters.curve`.

Impact

Since `PoolConfig` cannot be modified after creation, an invalid curve may result in a scenario where the pool never reaches completion, preventing `base_mint` from graduating.

Recommendation

It is recommended to impose a restriction on `ConfigParameters.curve` to ensure its length does not exceed `MAX_CURVE_POINT`.

Mitigation Review Log

Fixed in the commit 687af178903b78985a79572c32e1d714995ec9bc.

4.4 Volatility Could Be Largely Affected by Absolute Price Changes

Severity: Medium	Status: Fixed
Target: Smart Contract	Category: Math

Description

There is an issue with the current dynamic fee calculation formula.

Generally, accumulation is based on the delta of bin IDs, meaning that regardless of the current price, volatility is always accumulated in units of bin step bps. However, in the virtual curve, the `get_delta_bin_id` function treats bin ID movement as linear rather than exponential.

```
    // we approximate (1+bin_step)^bin_id = 1 + bin_step * bin_id
...
        let delta_id = if sqrt_price_a > sqrt_price_b {
            sqrt_price_a
                .safe_sub(sqrt_price_b)?
                .safe_div(bin_step_u128)?
        } else {
            sqrt_price_b
                .safe_sub(sqrt_price_a)?
                .safe_div(bin_step_u128)?
        };
        Ok(delta_id.safe_mul(2)?) // mul 2 because we are using sqrt price
```

programs/virtual-curve/src/state/fee.rs#L211-L227

As a result, volatility is more influenced by absolute price differences rather than proportion changes in bps.

Impact

This causes volatility to be measured in absolute price changes, leading to inflated volatility at higher prices and minimal volatility at lower prices. When `sqrt_price < bin_step`, volatility will always be zero.

Recommendation

It is recommended to improve the algorithm in `get_delta_bin_id` by replacing `Px - Py` with `Px / Py`. This adjustment will help mitigate the impact of absolute price changes on volatility.

$$P_x = (1+b)^x$$
$$P_y = (1+b)^y$$
$$\frac{P_x}{P_y} = (1+b)^{x-y} \approx 1 + b * (x-y)$$

Mitigation Review Log

Fixed in the commit b5823372081041bc55a1a2cb2147d040be084e9e.

4.5 Possible DoS on Migration or Claim Fee When Migrating to DAMM v2

Severity: Medium	**Status: Fixed**
Target: Smart Contract	**Category: Precision**

Description

In the `migration_damm_v2` IX, when both the creator and the partner hold shares, liquidity is added to the cp-amm in two separate steps.

Take the case where `creator_liquidity >` `partner_liquidity` as an example:

- **Step 1**: During `initialize_pool` , a position is first created for the creator and liquidity is added. When the cp-amm calculates `token_a_amount_1` and `token_b_amount_1` based on the provided liquidity, it uses **`rounding up`** .
- **Step 2**: After creating a position for the partner, liquidity is added via `add_liquidity` . Similarly, `token_a_amount_2` and `token_b_amount_2` are computed using **`rounding up`** .

Since both steps round up, it may result in `token_a_amount_1 + token_a_amount_2 > migration_base_threshold` or `token_b_amount_1 + token_b_amount_2 > migration_quote_threshold` .

Impact

When minting the base mint during `PoolConfig.get_initial_base_supply` , a buffer is reserved. However, for the quote mint, it's possible that insufficient quote amount causes a DoS.

If the protocol fee and trading fee have already been fully claimed prior to the migration, and the remaining quote amount is exactly the `migration_quote_threshold` , which is insufficient to complete both `initialize_pool` and `add_liquidity` , the `migration_damm_v2` instruction will fail.

If the fees haven't been claimed yet, the migration may succeed, but future `claim_protocol_fee` or `claim_trading_fee` instructions could fail due to the quote vault not having enough tokens to cover the recorded fee amounts in the pool.

Proof of Concept

Below is an example with problematic values:

```
partner_lp_percentage: 32 partner_locked_lp_percentage: 0
creator_lp_percentage: 68 creator_locked_lp_percentage: 0
migration_sqrt_price: 4295048016000000000000
migration_base_threshold: 9223026 migration_quote_threshold: 5000000000
...
[initialize_pool]: token base amount: 6271658 token quote amount:
  ↪ 3400000001
[add_liquidity]: token base amount: 2951369 token quote amount:
  ↪ 1600000000
```

```
total token base amount: 9223027 token quote amount: 5000000001

[!!!base!!!] actual base amount cost: 9223027 > threshold 9223026
[!!!quote!!!] actual quote amount cost: 5000000001 > threshold 5000000000
```

It shows that in the end, both the actual base and quote amounts used are **`1 unit greater`** than the respective thresholds.

Recommendation

It is suggested to recalculate the `total_liquidity` to be passed into `add_liquidity` during `create_second_position` based on `migration_quote_threshold - delta quote token amount after pool creation` .

Although this may cause the liquidity distribution between the creator and the partner to deviate slightly from the preset configuration in `config` , it ensures that the `add_liquidity` instruction can succeed and that subsequent fee claims will function as expected.

Liquidity reduced due to precision loss can be prioritized for deduction from `locked_liquidity` , if `locked_liquidity` is greater than zero.

Mitigation Review Log

Fixed in the commit 28c62ebe7d20fa73d70230ff0c0628aa6239c065.

4.6 create_config IX Fails When BaseFeeConfig.period_frequency > 0

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

Description

In `BaseFeeConfig.get_base_fee_numerator` , the latest `base_fee_numerator` is calculated.

In PR-40, the original logic that compared `current_point` and `activation_point` was removed: `PR-40` .

This introduces a new assumption: `current_point > activation_point` , but this assumption does not always hold—for example, in `BaseFeeConfig.get_min_base_fee_numerator` , where the input parameter can have `current_point < activation_point`
.

```rust
    pub fn get_min_base_fee_numerator(&self) -> Result<u64> {
        // trick to force current_point < activation_point (in order to
        ↪  get the lowest fee)
        self.get_base_fee_numerator(0, 1)
    }
```

programs/virtual-curve/src/state/config.rs#L153-L156

In such a case, when `BaseFeeConfig.period_frequency > 0`, `get_base_fee_numerator` will trigger a `safe_sub` error and return early.

Impact

If `create_config` IX is invoked with `BaseFeeParameters.period_frequency > 0`, it will fail due to the error in `BaseFeeConfig.get_min_base_fee_numerator`.

Recommendation

It is recommended to restore the original comparison logic.

Mitigation Review Log

Fixed in the commit 264f044d81fde3fc6636c1dc194654cada3f39c1 and a6eb5863405fd06b97eb6b22ef51249c48028025.

4.7 Reduced MAX_CURVE_POINT May Prevent Migration for Existing Virtual Pools

Severity: Low	**Status: Fixed**
Target: Smart Contract	**Category: Code QA**

Description

In PR-52, `MAX_CURVE_POINT` was reduced from 20 to 16. In both `get_swap_amount_from_quote_to_base` and `get_swap_amount_from_base_to_quote`, `MAX_CURVE_POINT` is used as the upper bound when looping through the curve points to compute the swap amount.

For virtual pools created before this upgrade, the old value of `MAX_CURVE_POINT` (20) was used during initialization. As a result, the reduction of this value may prevent swap functions from accessing curve points with indices in the range [16, 20).

Impact

If a virtual pool created before this upgrade requires the price to move into that range ([16, 20)) in order to reach its migration threshold, it could become impossible to migrate successfully. This change might prevent such virtual pools—originally able to complete migration—from ever doing so due to the reduced `MAX_CURVE_POINT` .

Recommendation

To ensure backward compatibility with older virtual pools, it's recommended to modify the loop upper bound in `get_swap_amount_from_quote_to_base` and `get_swap_amount_from_base_to_quote` , for example, by using `config.curve.len()` instead.

Mitigation Review Log

Fixed in the commit e76e7a64cce36043f223d7da456a2989fa5b5eb9.

4.8 Impact of Fee Collection Mode

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

Description

In the virtual curve, there are two fee collection modes: `OnlyB` and `BothToken` . The main difference between them lies in whether the fee is deducted from the input or the output when swapping quote for base.

In `CollectFeeMode::OnlyB` mode, the fee is first deducted from the input quote mint before the swap is executed. In `CollectFeeMode::BothToken` mode, the fee is deducted from the swap output base mint.

These two different fee collection modes, even with the same fee rate, do not produce entirely equivalent results.

Impact

A quote to base swap with same input amount and same start price can get more ouput base tokens in `OnlyB` mode than in `BothToken` mode. Due to differing average swap price and output token amount for users.

For example, assuming at the current curve, 900 quote tokens can be swaped to 900 base tokens and, and at the next curve, 100 quote tokens can be swapped to 10 base tokens. The fee is 10% and there are 1000 quote tokens waiting for swaping.

Under `BothToken` mode, 1000 quote tokens are swapped to 910 base tokens, and finally with 819 base tokens as output and 91 base tokens as fee. Under `OnlyB` mode, the swapping result is 900 base tokens as output and 100 quote tokens as fee.

Recommendation

It will be better to collect swap fees on input amounts in normal cases. There are two benefits:

1. swap fees will be excluded from the swap, so swap fees will not impact the final price and slippage.
2. even under `OnlyB` mode, users can get a same base to quote swap result as the `BothToken` mode. And if we dont want that, we also can decide the price which the quote swap fees are swapped at.

4.9 Informational and Undetermined Issues

Redundant IXs

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

In the module `virtual_curve` , the following two sets of IXs are functionally and semantically equivalent.

- `partner_claim_lp_from_meteora_dynamic_amm` and `migrate_meteora_damm_partner_claim_lp_token`
- `creator_claim_lp_from_meteora_dynamic_amm` and `migrate_meteora_damm_creator_claim_lp_token`

CU Optimization

Severity: Informational	Status: Fixed
Target: Smart Contract	Category: CU

In the `create_config` IX, `swap_base_amount_buffer` , `minimum_base_supply_with_buffer` , and `minimum_base_supply_without_buffer` are only used when creating a fixed supply config. Therefore, they can be calculated inside the if block after parsing `token_supply` , which would help save compute units when creating a dynamic supply config.

Inconsistent Rounding Direction When Calculating Base Token Amount

Severity: Informational	Status: Acknowledged
Target: Smart Contract	Category: Math

In `get_base_token_for_swap` , the base mint amount that can be swapped out using the quote is calculated. However, when calling `get_delta_amount_base_unsigned` , `Rounding::Up` is used. In a swap, it is recommended to use `Rounding::Down` for the output mint amount, as it is more appropriate. This ensures consistency with the rounding direction in `get_swap_result_from_quote_to_base` .

Missing Validation on pool_fees.protocol_trade_fee_numerator

Severity: Informational **Status: Acknowledged**

Target: Smart Contract **Category: Data Validation**

In `MigrateMeteoraDammCtx.validate_config_key` , `damm_config.pool_fees.trade_fee_numerator` is subject to certain restrictions. It is recommended to apply similar validation for `damm_config.pool_fees.protocol_trade_fee_numerator` to ensure parameter validity and constrain the protocol fees range.

Missing Memo Support for Token-2022 Transfers in transfer_from_pool

Severity: Informational **Status: Acknowledged**

Target: Smart Contract **Category: Token**

During a swap, both the base mint and quote mint could potentially be Token-2022 mints. In Token-2022, users can opt to enable the `Memo` extension for their token accounts, which enforces the presence of a separate Memo instruction before any transfer instruction. However, the `transfer_from_pool` function does not yet handle logic related to the Memo extension. We recommend adding support for Memo to avoid transfer failures caused by its enforcement.

Rounding Direction When Calculating Base Token Amount

Severity: Informational **Status: Acknowledged**

Target: Smart Contract **Category: Precision**

In `get_migration_base_token` , the base mint amount during migration is calculated. However, when the `migration_option` is `MigrationOption::DammV2` , the function `get_delta_amount_base_unsigned` is called using `Rounding::Up` . It is generally more appropriate to use `Rounding::Down` when calculating the output mint amount.

Pools With Specific Config Can Not Be Migrated

Severity: Informational **Status: Acknowledged**

Target: Smart Contract **Category: Logic Error**

For pools with the migration option of Dynamic AMM, if the `migration_quote_threshold` is `1` , the pool can’t be migrated to the dynamic AMM pool because of the precision loss when depositing the 1 quote token to the dynamic vault.

The pool can't be migrated when the quote reserve reaches the migration quote threshold.

Assuming there is a malformed curve config:

- Quote mint is USDC;
- `migration_quote_threshold` is 1;
- the curve[0] is {sqrt_price: `MAX_SQRT_PRICE / 2` , liquidity: `1<<64` };
- start price is `MIN_SQRT_PRICE`

The migration prameters will be as follows:

```
swap_base_amount_256: 18446744078004599632
swap_base_amount: 4294886577
migration_base_amount: 1
```

Because the LP rate of the USDC vault is greater than zero
`https://solscan.io/token/3RpEekjLE5cdcG15YcXJUpxSepemvq2FpmMcgo342BwC` .

When creating the dynamic amm pool, the lp token deposited will be zero, the creation will failed.

It is really a challenge to prevent these malformed curve configs to break the normal flow of the launch pool. It could be better if we can enforce reasonable parameter bounds. Should production genuinely require out-of-bound values for specific use cases, we would explicitly declare dedicated instructions or specific use case handlers, rather than allowing such exceptions to hide in undocumented corner cases.

5 Disclaimer

This report reflects the security status of the project as of the date of the audit. It is intended solely for informational purposes and should not be used as investment advice. Despite carrying out a comprehensive review and analysis of the relevant smart contracts, it is important to note that Offside Labs' services do not encompass an exhaustive security assessment. The primary objective of the audit is to identify potential security vulnerabilities to the best of the team's ability; however, this audit does not guarantee that the project is entirely immune to future risks.

Offside Labs disclaims any liability for losses or damages resulting from the use of this report or from any future security breaches. The team strongly recommends that clients undertake multiple independent audits and implement a public bug bounty program to enhance the security of their smart contracts.

The audit is limited to the specific areas defined in Offside Labs' engagement and does not cover all potential risks or vulnerabilities. Security is an ongoing process, regular audits and monitoring are advised.

Please note: Offside Labs is not responsible for security issues stemming from developer errors or misconfigurations during contract deployment and does not assume liability for centralized governance risks within the project. The team is not accountable for any impact on the project's security or availability due to significant damage to the underlying blockchain infrastructure.

By utilizing this report, the client acknowledges the inherent limitations of the audit process and agrees that the firm shall not be held liable for any incidents that may occur after the completion of this audit.

This report should be considered null and void in case of any alteration.

OFFSIDE LABS

https://offside.io/

https://github.com/offsidelabs

https://twitter.com/offside_labs