OFFSIDE LABS

DLMM Token-2022

Smart Contract Security Assessment

March 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 4 security audits of *DLMM* smart contracts, the first audit began in June 19, 2024, and the last audit concluded in March 12, 2025.

Project Overview

Meteora's Dynamic Liquidity Market Maker (DLMM) gives LPs access to dynamic fees and precise liquidity concentration all in real-time. DLMM is designed with features that provide more flexibility for LPs and help LPs earn as much fees as possible with their capital.

DLMM organizes the liquidity of an asset pair into discrete price bins. Token reserves deposited in a liquidity bin are made available for exchange at the price defined for that particular bin. Swaps that happen within the price bin itself would not suffer from slippage. The market for the asset pair is established by aggregating all the discrete liquidity bins.

This update introduces support for SPL-Token-2022, leveraging V2 instructions to seamlessly integrate SPL-Token-2022 features and extensions. This enhancement enables users to interact with advanced token functionalities, while maintaining compatibility with existing operations.

Audit Scope

The assessment scope contains mainly the smart contracts of the *lb_clmm* program for the *DLMM* project.

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

- DLMM:
  - Branch: staging
  - PR-286:
    - Commit Hash: c2d45937facf14cc9ffb748f11fe5340a70fc12f
    - Codebase Link: PR-286
  - PR-290:
    - Commit Hash: 3baa5ffe8840e7320b1ecad5bf833bba76eb8da4
    - Codebase Link: PR-290
  - PR-345:
    - Commit Hash: 4cc5b1c6e8e942632f05aefad72bbb8fc169510e
    - Codebase Link: PR-345
  - PR-353 to PR-355:
    - Commit Hash: d58175d265bce1c3825306e72b908fa9a00d0e89
    - Codebase Link:
      - PR-353
      - PR-354
      - PR-355

We listed the files we have audited below:

- DLMM:
  - programs/lb_clmm/src/*.rs

Findings

The security audit revealed:

- 1 critical issues
- 1 high issues
- 2 medium issues
- 2 low issues
- 4 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	Loss of User Balance When Enabling Host Fee	Critical	Fixed
02	Incorrect Program ID Used to Find ATA	High	Fixed
03	Not Support All Types of Customized Pairs in remove_liquidity2	Medium	Fixed
04	Missing Check When Depositing Into Active Bin in add_liquidity2	Medium	Fixed
05	Inconsistency in Transfer Fee Responsibility When Collecting Host Fees	Low	Fixed
06	Total Transfer Fees May Increase	Low	Acknowledged
07	Client Should Verify the Metadata Pointer Extension	Informational	Fixed
08	calculate_pre_fee_amount Is Inconsistent with the Actual Pre-Fee Amount	Informational	Fixed
09	Redundant Memo Transfer	Informational	Fixed
10	SPL Token With freeze_authority Will Pass validate_mint Check	Informational	Acknowledged

4 Key Findings and Recommendations

4.1 Loss of User Balance When Enabling Host Fee

Severity: Critical	Status: Fixed
Target: Smart Contract	Category: Math Error

Description

In `internal_handle_swap2`, if `token_mint_in` is a mint with a `TransferFee`, `lb_pair` will ignore the Token2022 Transfer Fee portion of `amount_in` during the swap. After the swap operation is performed on the bins involved, the Token2022 Transfer Fee is reconsidered before triggering `transfer_checked`.

The `amount_in` in `internal_handle_swap2` can be broken down into six parts, explained as follows:

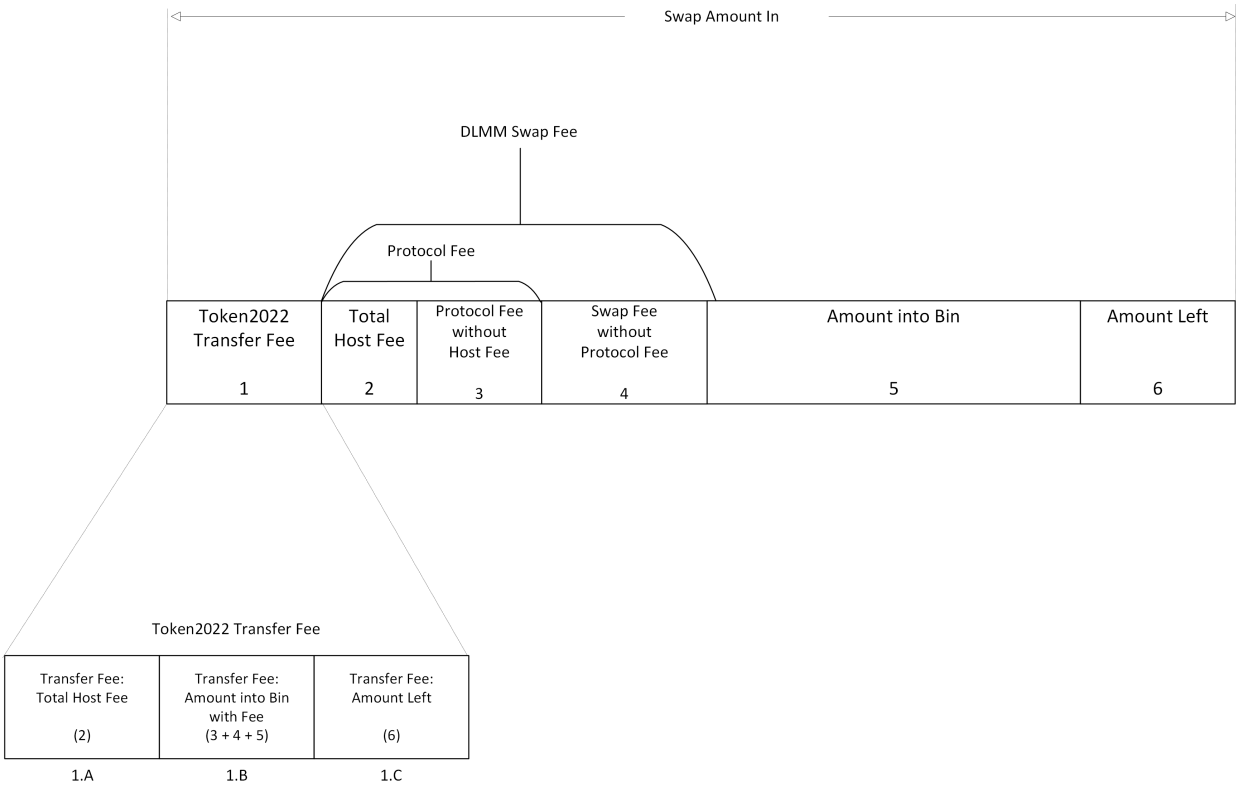


Figure 1: Transfer Fees

Part 1: Token2022 transfer fee: `lb_pair` removes this portion from `amount_in` before the swap, then uses the `amount_in` minus the Token2022 transfer fee as the actual input value for the swap.

Part 2: During the swap, a host fee is charged at a fixed rate of 20% from the protocol fee.

Part 3: The remaining protocol fee after the host fee is deducted, representing the protocol's actual revenue.

Part 4: The remaining fee collected from the user's account during the swap after deducting the protocol fee.

Part 5: The amount of `token_mint_in` actually added to the pool as liquidity by the user after the swap.

Part 6: The excess amount of `token_mint_in` in `amount_in` after the swap.

Further breaking down the Token2022 transfer fee, it can be divided into three parts:

Part 1.A (transfer fee of `Part 2`): The transfer fee required when transferring the host fee from the user account.

Part 1.B (transfer fee of `Part 3 + Part 4 + Part 5`): The transfer fee required when transferring from the user account to the bin in `lb_pair`.

Part 1.C (transfer fee of `Part 6`): The transfer fee corresponding to the amount left, which is not charged as the transfer did not actually occur.

This is the code snippet from `internal_handle_swap2` regarding the calculation of the transfer amount:
programs/lb_clmm/src/instructions/v2/swap2.rs#L546-L563

From the calculation results, the final `transfer_fee_included_host_fee` includes Part 2, Part 1.A, Part 1.C. This composition, besides the host fee itself, also includes other non-host fee-related Token2022 transfer fees: `Part 1.C`.

Impact

The host fee receiver can benefit from this issue to collect more host fees.

In `swap_exact_out2`, if the user sets a large slippage, the `amount_left` increases, thereby increasing the value of Part 1.C. The host fee receiver can directly take Part 1.C from the reserved Token2022 Transfer fee in Step 1, which do not belong to them. This portion of the token is actually transferred from the extra `amount_in` of the swap user.

In `swap2`, `amount_left` is always 0, so there is no Part 1.C Transfer Fee. However, when calculating the `transfer_fee_included_host_fee`, due to precision issues, the host fee receiver still receives more host fees than expected.

Proof Of Concept

1. Create a lb_pair with a mint pair (BTC, USDC). BTC is a token with Token2022 TransferFee Extension (3% Fee Basis Points).
2. Add Liquidity (1 BTC, 500 USDC) for lb_pair.
3. Invoke swapExactOut2 with a large slippage input (1 BTC => 1 USDC).
4. Transfer Host Fee Receiver gets larger host fee than the protocol expected.

Recommendation

The calculation method for `transfer_fee_included_host_fee` needs to be revised to exclude the initially reserved but unused Token2022 Transfer fee.

Mitigation Review Log

Meteora Team: PR-303

Offside Labs: In the current fixed version, `Part 1.C` is correctly subtracted from the transfer fee, but in certain cases, it causes the host fee receiver to get a reduced host fee. In fact, the host fee will be used to cover the transfer fee that should be paid by the user. This issue is detailed in the "***Inconsistency in Transfer Fee Responsibility When Collecting Host Fees***" of this report.

Meteora Team: A new transfer fee calculation has been implemented in PR-355.

Offside Labs: Fixed.

4.2 Incorrect Program ID Used to Find ATA

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

Description

The `remove_liquidity2` instruction incorrectly uses the mint address as the token program id to look up the ATA.

```
24 let dest_token_x = anchor_spl::associated_token::
25     get_associated_token_address_with_program_id(
26         &owner,
27         &self.token_x_mint.key(),
28         &self.token_x_mint.key(),
29     );
```

programs/lb_clmm/src/instructions/v2/remove_liquidity2.rs#L24-L29

Impact

This could prevent users from withdrawing token-2022 related liquidity.

Recommendation

Use `token_x_program` and `token_y_program` as the token program id.

Mitigation Review Log

Offside Labs: Fixed in PR-351

4.3 Not Support All Types of Customized Pairs in remove_liquidity2

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

Description

In the `remove_liquidity2` IX, the current slot of the IX is used to determine whether the liquidity in the position can be removed.

```
67      require!(
68          !position.is_liquidity_locked(Clock::get()?.slot),
69          LBError::LiquidityLocked
70      );
```

programs/lb_clmm/src/instructions/v2/remove_liquidity2.rs#L67-L70

However, it does not account for `customized_permissionless_pair` with `activation_type` set to `ActivationType::Timestamp` .

Impact

This may result in such positions being unable to remove liquidity as expected.

Recommendation

It is recommended to perform a check similar to that in the `remove_liquidity` IX.

```
135 let pair_type_access_validator =
    ↪   get_lb_pair_type_access_validator(&lb_pair)?;
136 // check liquidity is locked
137 require!(!position.is_liquidity_locked(
138     pair_type_access_validator.get_current_point()),
139     LBError::LiquidityLocked
140 );
141 require!(pair_type_access_validator.validate_remove_liquidity_access(
142     total_amount_x != 0)?,
143     LBError::LiquidityLocked
144 );
```

programs/lb_clmm/src/instructions/withdraw/remove_liquidity.rs#L135-L144

Mitigation Review Log

Offside Labs: Fixed in PR-349 .

4.4 Missing Check When Depositing Into Active Bin in add_liquidity2

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

Description

In the `add_liquidity` IX, if a quote token is being deposited into an active bin, the program checks whether this action is allowed for the pair.

```
288 if bin_id == active_id && amount_y > 0 {
289 let pair_type_access_validator =
    ↪ get_lb_pair_type_access_validator(&lb_pair)?;
290 require!(
291 pair_type_access_validator.validate_deposit_quote_token_in_active_bin(),
292     LBError::PoolDisabled
293     );
294 }
```

programs/lb_clmm/src/instructions/deposit/add_liquidity.rs#L288-L294

However, in the `add_liquidity2` IX, this check is not performed.

Impact

If the pair currently does not allow to deposit quote token, a malicious user could bypass this restriction using the `add_liquidity2` IX.

Recommendation

It is recommended to add the `LbPairTypeActionAccess.validate_deposit_quote_token_in_active_bin` check to the `add_liquidity2` IX.

Mitigation Review Log

Offside Labs: Fixed in PR-349 .

4.5 Inconsistency in Transfer Fee Responsibility When Collecting Host Fees

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

Description

In the `swap_exact_out` process, when the Transfer Fee `Part 1.B` has reached the `max_fee` (Part 1 should also be `max_fee`), in order to satisfy the equation `Part 1 == Part 1.A + Part 1.B + Part 1.C` , both Part 1.A and Part 1.C must be 0. However, when amount_left > 0, the fix code calculates a fee `(Part 1.C)` within `transfer_fee_included_amount_left` that is greater than 0. This fee is then effectively deducted from the host fee, leading to the host fee receiver getting less than the expected amount. In other words, the host fee receiver paid the transfer fee on behalf of the user.

Recommendation

To simplify the transfer amount calculation of the host fee, ignore the inverse fees calculation associated with `amount_left` and host fee.

Mitigation Review Log

Meteora Team: A new transfer fee calculation has been implemented in PR-355.

Offside Labs: Fixed.

4.6 Total Transfer Fees May Increase

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

Description

Before this change, PR-355, if swap in token amount does not reach the max transfer fee, the transfer fee of the host fee transfer is borne by the user. After this change, the transfer fee of the host fee transfer will now be shouldered by the host themselves (meaning they receive less host fee). Additionally, the overall transfer fee has increased. Because the total number of tokens being transferred could increase.

Impact

Under the current implementation, both users and hosts may need to pay more transfer fees for swaping the same amount of tokens.

Proof Of Concept

Token A has a transfer fee of 1%**.** A user wants to swap 100,000 Token A for Token B. and the host fee is 1,000 Token A based on the swap in amount.

- Before the Change:
    1. The user transfers 98,990 Token A to the pool.
    2. The user sends 1,010 Token A as the host fee.
    3. After the 1% transfer fee, the host receives **999 Token A**.
    4. The total transfer fee is **1,001 Token A.**
- After the Change:
    1. The user transfers 100,000 Token A to the pool.
    2. The pool sends 1,000 Token A as the host fee.
    3. After the 1% transfer fee, the host receives **990 Token A**.
    4. The total transfer fee is **1,010 Token A.**

Recommendation

For swap exact in, the original implementation can be followed to separately calculate the token amounts (including transfer fees) to be sent to the pool and host. This ensures the total tokens transferred are minimized each time, thereby reducing the transfer fees.

Mitigation Review Log

Meteora Team: We are updating to allow a flexible fee collection mode.

4.7 Informational and Undetermined Issues

Client Should Verify the Metadata Pointer Extension

Severity: Informational	Status: Fixed
Target: Client	Category: Data Validation

The `validate_mint` function uses whitelist validation for extensions to ensure that the program only accepts token-2022 that has `MetadataPointer` or `TokenMetadata` enabled.

If the client needs to use the Metadata pointed to by MetadataPointer for front-end display or branching conditions, please ensure that the MetadataPointer of the mint and that the mint field in the corresponding `TokenMetadata` account both point to each other, to avoid phony mints claiming to be tokens with special properties, e.g. stablecoins.

calculate_pre_fee_amount Is Inconsistent with the Actual Pre-Fee Amount

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

`calculate_pre_fee_amount` calculates the initial amount before fees with `post_fee_amount` and `TransferFee.transfer_fee_basis_points` .

However, when `transfer_fee_basis_points` is `ONE_IN_BASIS_POINTS` , it returns 0 as the initial amount. This may introduce inconsistency between the contract's internal accounting and transfers if `post_fee_amount` is greater than 0.

```
416 } else if transfer_fee_basis_points == ONE_IN_BASIS_POINTS
417     || post_fee_amount == 0 {
418     Some(0)
419 }
```

programs/lb_clmm/src/utils/token2022.rs#L416-L418

Although the current program ensures that the call chain using `calculate_pre_fee_amount` within `calculate_transfer_fee_included_amount` excludes the possibility of returning 0, it is recommended to change the return value for `transfer_fee_basis_points == ONE_IN_BASIS_POINTS` to `maximum_fee.checked_add(post_fee_amount)` , considering future code that might use `calculate_pre_fee_amount` .

Mitigation Review Log

Meteora Team: PR-303

Offside Labs: Fixed.

Redundant Memo Transfer

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

During `perform_swap` , transferring the user input tokens to `token_vault_in` with the `transfer_from_user2` call doesn't need to include `memo_transfer_context` : programs/lb_clmm/src/instructions/v2/swap2.rs#L171

Since `token_vault_in` is the transfer destination in DLMM and does not enable the Memo Transfer Extension, It's better to set the Memo Transfer Context to `None` .

SPL Token With freeze_authority Will Pass validate_mint Check

Severity: Informational	Status: Acknowledged
Target: Smart Contract	Category: Data Validation

In `validate_mint` , if a mint's owner is the SPL Token, it automatically passes validation and returns.

However, a Token-2022 mint with `freeze_authority` but without a corresponding token badge is flagged as Unsupported. For SPL Tokens, even those with `freeze_authority`, the check will still pass them as valid if they don't have token badges.

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.

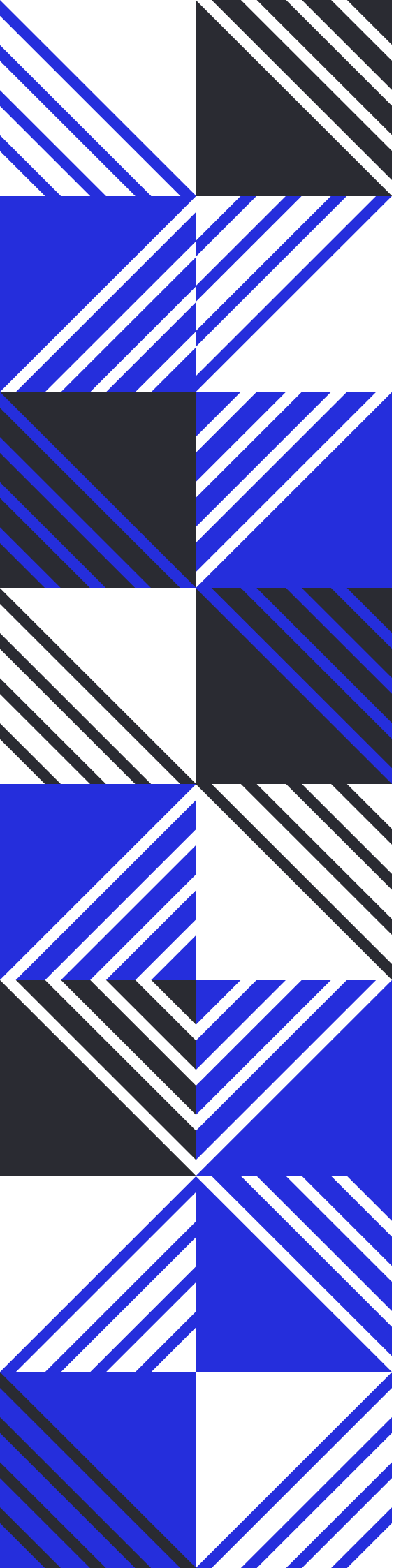
Offside Labs disclaims any liability for losses or damages resulting from the use of this report or from any future security breaches. The team strongly recommends that clients undertake multiple independent audits and implement a public bug bounty program to enhance the security of their smart contracts.

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