OFFSIDE LABS

Solana Cross-Chain

Smart Contract Security Assessment

July 2025

Prepared for:

1inch

Prepared by:

Offside Labs

Gongyu Shi

Tim Li

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 *1inch Solana Cross-Chain* smart contracts, starting on June 16th, 2025, and concluding on July 10th, 2025.

Project Overview

The 1inch Solana Cross-Chain project implements the `1inchFusion+` , an atomic cross-chain swaps protocol, that enables seamless token exchanges between Solana and EVM chains. By utilizing both Hashlock and Timelock mechanisms, along with a Dutch auction model, the protocol achieves a compact and efficient design.

Users can create an order on the source chain and deposit the tokens to be swapped. Once a resolver accepts the order, the user only needs to share the secret to unlock the tokens. All remaining steps are handled by the resolvers, and finally the output tokens are transferred to the user's account on the destination chain.

Audit Scope

The assessment scope contains mainly the smart contracts of the cross-chain-escrow-src, cross-chain-escrow-dst and whitelist programs for the *1inch Solana Cross-Chain* project.

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

- 1inch Solana Cross-Chain
  - Codebase: `https://github.com/1inch/solana-crosschain-protocol/`
  - Commit Hash: 06ef2533ab5dd451a6d61bda677d54e6e628e21e

We listed the files we have audited below:

- 1inch Solana Cross-Chain
  - common/src/escrow.rs
  - common/src/utils.rs
  - programs/cross-chain-escrow-dst/src/lib.rs
  - programs/cross-chain-escrow-src/src/auction.rs
  - programs/cross-chain-escrow-src/src/lib.rs
  - programs/cross-chain-escrow-src/src/merkle_tree.rs
  - programs/whitelist/src/lib.rs

Findings

The security audit revealed:

- 0 critical issue
- 2 high issues

- 3 medium issues
- 2 low issues
- 6 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	Asset Loss of Maker Due to Incorrect Amount Recorded in the Escrow Account	High	Fixed
02	Rent Extraction Arbitrage via Forced ATA Creation in Cancel IXs	High	Acknowledged
03	Malicious Resolver Can Steal Maker's Tokens Due to Missing Mint Validation	Medium	Fixed
04	Incorrect dst_amount Calculation May Cause Swap Failure	Medium	Fixed
05	Missing Validation for Parameters When Creating the Order and Escrow	Medium	Acknowledged
06	Order Re-Creation May Lead to Confusion	Low	Fixed
07	Resolver Reward Loss Due to Forced ATA Creation in Withdraw IXs	Low	Acknowledged
08	Unnecessary CPI When Transfer Amount Is Zero	Informational	Fixed
09	Unnecessary Canonical Bump Calculation During Resolver Deregistration	Informational	Fixed
10	Missing Access Control for Escrow Creation on the Destination Chain	Informational	Acknowledged
11	Missing sync_native CPI During Escrow Creation on the Destination Chain	Informational	Acknowledged
12	Recipient ATA Not Required When Token Is Native	Informational	Acknowledged
13	Unchecked reward_limit in the cancel_order_by_resolver IX	Informational	Acknowledged

4 Key Findings and Recommendations

4.1 Asset Loss of Maker Due to Incorrect Amount Recorded in the Escrow Account

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

Description

During the creation of an escrow account via the `create_escrow` IX, the `amount` parameter is used to derive the escrow PDA.

```
620 #[instruction(amount: u64, ...]
621 pub struct CreateEscrow<'info> {
622     ...
623     #[account(
624         init,
625         payer = taker,
626         space = constants::DISCRIMINATOR_BYTES + EscrowSrc::INIT_SPACE,
627         seeds = [
628             ...
629             amount.to_be_bytes().as_ref(),
630             ...
631         ],
632         bump,
633     )]
634     escrow: Box<Account<'info, EscrowSrc>>,
```

programs/cross-chain-escrow-src/src/lib.rs#L620-L634

However, within this instruction, the `escrow.amount` field is incorrectly set to `order.amount` instead of the `amount` parameter passed in.

```
203         ctx.accounts.escrow.set_inner(EscrowSrc {
204             ...
205             amount: order.amount,
206             ...
207         });
```

programs/cross-chain-escrow-src/src/lib.rs#L203-L207

Impact

If an order allows multiple fills, the `amount` in the passed-in parameters may be less than `order.amount`, resulting in a mismatch between the "amount" used to derive the PDA and the "amount" recorded in the PDA account.

```
712     #[account(
713         mut,
714         seeds = [
715             ...
716             escrow.amount.to_be_bytes().as_ref(),
717                         ...
718         ],
719         bump,
720     )]
721     escrow: Box<Account<'info, EscrowSrc>>,
```

programs/cross-chain-escrow-src/src/lib.rs#L712-L721

This misalignment causes a permanent DoS for all other IXs (`withdraw` , `public_withdraw` , `cancel_escrow` and `public_cancel_escrow`) that utilize the escrow account, since the constraint above uses the incorrect "amount".

If the swap is successful, the taker will be unable to claim the assets in the escrow immediately, as any withdraw operations will fail, and they can only retrieve them through the rescue operation.

If the swap fails, the maker will be unable to refund their assets, as any cancel operations may fail, resulting in a direct loss for the maker.

Recommendation

Set `escrow.amount` to `amount` in the parameters instead of `order.amount` during initialization.

Mitigation Review Log

Fixed in the commit 6e0ba0911da36d56ea62967897e1e981a7ec33d9.

4.2 Rent Extraction Arbitrage via Forced ATA Creation in Cancel IXs

Severity: High	Status: Acknowledged
Target: Smart Contract	Category: Protocol Design

Description

The `cancel_escrow` , `public_cancel_escrow` and `cancel_order_by_resolver` IXs on the source chain can help to refund the assets stored in the escrow or order ATA when the swap if failed. If the token is not native, then the ATA of the maker is also required by these IXs. If this account does not exist, the one who is going to execute these IXs is forced to create it for the maker.

```
984     #[account(
985         mut,
986         associated_token::mint = mint,
987         associated_token::authority = creator,
988         associated_token::token_program = token_program
989     )]
990     // Optional if the token is native
991     creator_ata: Option<Box<InterfaceAccount<'info, TokenAccount>>>,
```

programs/cross-chain-escrow-src/src/lib.rs#L984-L991

Impact

For the `cancel_escrow` case, a malicious maker can create a legitimate order for a non-native token, then close their corresponding ATA to reclaim the rent. Once a taker engages with the order, the maker may intentionally refuse to share the correct secret or provide an invalid one, causing the swap to fail, and the escrow will be canceled.

In order to claim the paid rent of `escrow` and `escrow_ata`, the taker is likely to compromise by creating the ATA for the maker. A similar issue exists in the `public_cancel_escrow` scenario. In short, a malicious maker can repeatedly execute this strategy with no risk, arbitraging the ATA rent each time. The taker loses the rent, which is eventually claimed by the maker.

In the `cancel_order_by_resolver` case, the resolver is required to cover the rent for the maker's ATA, which may not be fully reimbursed by the `cancellation_premium`. This significantly reduces the resolver's incentive to assist in canceling orders.

Recommendation

A possible fix is to introduce a deposit mechanism. When creating an order (e.g., one that is split into 4 parts, resulting in up to 5 fills), the maker must provide a deposit equal to the rent required for 5 ATAs.

As each escrow is created, one-fifth of the deposit is transferred into the escrow. During withdrawal or cancellation operations, the maker's ATA must be provided and validated. If the ATA exists, the corresponding portion of the deposit is refunded to the maker. Otherwise, the deposit is used to create the required ATA.

Mitigation Review Log

1inch Team: We find it highly unlikely that a maker would intentionally delete their own ATA and block receiving funds. Resolver should take this loss into account.

4.3 Malicious Resolver Can Steal Maker's Tokens Due to Missing Mint Validation

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

Description

In the `create_escrow`, `cancel_order_by_resolver`, `cancel_escrow` and `public_cancel_escrow` IXs, the passed-in `mint` account is not validated, it may differ from the expected token mint recorded in `order.token` or `escrow.token`.

```
637     /// CHECK: check is not necessary as token is only used as a
        ↪ constraint to creator_ata and order
638     mint: Box<InterfaceAccount<'info, Mint>>,
639
640     /// Account to store order details
641     #[account(
642         mut,
643         seeds = [
644             ...
645             order.token.key().as_ref(),
646             ...
647         ],
648         bump,
649     )]
650     order: Box<Account<'info, Order>>,
```

programs/cross-chain-escrow-src/src/lib.rs#L637-L650

Impact

In the `create_escrow` case, assuming the order supports multiple fills and the previous issue has been fixed (otherwise any multiple fills cannot work correctly), a malicious resolver can exploit it by creating two escrows.

When creating the first one, the `mint` is set to a token mint fully controlled by themselves (so that they can create the corresponding `order_ata` and mint some tokens to it) and `amount` is set to `order.amount - 1`. After the execution, the `order.remaining_amount` will be reduced to 1. Then the malicious resolver creates the second escrow where the `mint` is set to the real mint, and the `amount` is set to 1. Due to the logic in the following code, all the real tokens in the real `order_ata` will be transferred to the second escrow, but the recorded amount of the escrow is 1. Finally, the resolver only needs to handle the swap for 1 token and may claim all the tokens in that order.

```
186         if order.remaining_amount == amount {
187             // Transfer amount may be increased due to external transfers
188             amount_to_transfer = ctx.accounts.order_ata.amount;
189         }
```

programs/cross-chain-escrow-src/src/lib.rs#L186-L189

All other cases are related to the cancelation of orders or escrows, and the malicious resolver can provide a fake `mint` with the similar method mentioned above. Then, the real `order` or `escrow` accounts will be closed, while the real `order_ata` or `escrow_ata` remains active. As a result, all the assets in these ATAs can never be refunded to the maker, what's worse, the malicious resolver may illegitimately seize them via the `rescue_funds_for_escrow` or `rescue_funds_for_order` IXs.

Recommendation

It is recommended to validate the `mint` account in these instructions.

Mitigation Review Log

Fixed in the commit 882fbad8ff1378cda8c959428ed32e5a3fb6618f.

4.4 Incorrect dst_amount Calculation May Cause Swap Failure

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

Description

In the `create_escrow` IX on the source chain, the expected amount of tokens to be deposited in the destination escrow (`dst_amount`) is derived based on `order.dst_amount` and the auction parameters.

```
217     dst_amount: get_dst_amount(order.dst_amount, &dutch_auction_data)?,
```

programs/cross-chain-escrow-src/src/lib.rs#L217-L217

However, since the order may not be fully fulfilled, the actual `dst_amount` should be calculated proportionally according to the fulfilled portion of the order.

Impact

If the order allows multiple fills and the current escrow does not fulfill the order entirely, the `dst_amount` field is incorrectly set. In such cases, the value of `dst_amount` does not correspond to the actual amount of tokens deposited in the destination escrow, which can lead to a swap failure.

Recommendation

It is recommended to calculate `dst_amount` proportionally based on the ratio between `escrow.amount` and `order.amount`.

Mitigation Review Log

Fixed in the commit 5595c32680e5bc527300a2354aeb9f79733f0d1e.

4.5 Missing Validation for Parameters When Creating the Order and Escrow

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

Description

In the `create` IX on both the source and destination chains, numerous parameters are passed in. However, most of them are either not validated or are subject to insufficient checks.

1. The `finality_duration`, `withdrawal_duration`, `public_withdrawal_duration` and `cancellation_duration` are not validated.
2. The `safety_deposit` is checked to be greater than zero and does not exceed the rent required for an escrow account.
3. The `max_cancellation_premium` is checked to ensure it does not exceed the lamport balance of the order account.

Impact

When creating an order on the source chain, several parameters may be intentionally or unintentionally misconfigured, leading to unexpected or undesirable behavior:

1. Short Durations: If the user sets a very short `withdraw_duration`, the taker may be unable to claim the tokens in time, even if the secret has been revealed.
2. Low `safety_deposit` (e.g., 1): If the `safety_deposit` is set to a negligible value such as 1, resolvers may have no financial incentive to help to perform `public_withdraw` or `public_cancel` operations.
3. Low `max_cancellation_premium`: If `max_cancellation_premium` is set to 0, a DoS condition may occur when invoking `cancel_order_by_resolver`, due to the check `order.max_cancellation_premium > 0`.

Similar impacts can be observed when the taker creates the escrow on the destination chain.

Recommendation

It is recommended to implement additional checks to ensure that all duration parameters fall within a reasonable range, and to enforce appropriate lower bounds for both `safety_deposit` and `max_cancellation_premium`.

Mitigation Review Log

1inch Team: Backend performs validation on all orders before forwarding them to resolvers.

4.6 Order Re-Creation May Lead to Confusion

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

Description

Once the user creates the `order` account, they may close it via the `cancel_order` IX. However, it is still possible for the user to recreate the same account at a later time using the `create` IX, as long as all parameters used to derive the PDA remain unchanged.

Impact

During the recreation of an order account, certain parameters may be modified, such as all durations, `dutch_auction_data_hash`, `max_cancellation_premium` and `parts_amount`. This may cause confusion for resolvers, especially if the original order was partially fulfilled, as they may assume that these parameters associated with the recreated order are the same as those of the previously closed one.

Recommendation

It is recommended to implement an off-chain check to prevent reuse of the same `order_hash`, or alternatively compute the `order_hash` on-chain using all relevant input parameters to ensure uniqueness.

Mitigation Review Log

Fixed in the commit 957245423aa4c77693547e15c7d5589301306b50.

4.7 Resolver Reward Loss Due to Forced ATA Creation in Withdraw IXs

Severity: Low	Status: Acknowledged
Target: Smart Contract	Category: Protocol Design

Description

The `public_withdraw` IXs on both the source and destination chains require the ATA of the taker or maker, respectively, when the token involved is non-native. If the corresponding ATA does not exist, the resolver who helps to invoke these IXs may be forced to pay the rent for the ATA creation.

Impact

This results in a reduction of the expected reward received by the resolver. Furthermore, if the `safety_deposit` is insufficient to cover the rent of an ATA, there may be no incentive for any resolver to execute the withdrawal, as doing so would lead to a net loss. This behavior deviates from the intended protocol design.

Recommendation

It is recommended to validate that the `safety_deposit` is sufficient to cover the rent for one ATA.

Specifically, for the source chain case, it is also recommended to check whether the required ATA exists. If it does not, the rent from the escrow ATA (which is intended to be closed after the withdrawal) will not be refunded to the taker. Instead, it will be used to create the taker's ATA. This method ensures that the resolver can always receive the full `safety_deposit`.

Mitigation Review Log

1inch Team: We believe there is no strong incentive for takers or makers to delete their ATAs.

4.8 Informational and Undetermined Issues

Unnecessary CPI When Transfer Amount Is Zero

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

In the `cancel_order_by_resolver` IX, the `maker_amount` is calculated and it is the amount of lamports that will be refund to the maker. If it is zero, then there is no need to perform a transfer. It is suggested to add a check inside the `uni_transfer`.

```
464         let maker_amount =
              ↪ ctx.accounts.order_ata.to_account_info().lamports()
465             - std::cmp::min(cancellation_premium, reward_limit);
466         ...
467         uni_transfer(
468             &UniTransferParams::NativeTransfer {
469                 from: ctx.accounts.resolver.to_account_info(),
470                 to: ctx.accounts.creator.to_account_info(),
471                 amount: maker_amount,
472                 program: ctx.accounts.system_program.clone(),
473             },
474             None,
475         )?;
```

programs/cross-chain-escrow-src/src/lib.rs#L464-L475

Fixed in the commit ec78cee25b216d15c5dc75e2762dc6d20608b660.

Unnecessary Canonical Bump Calculation During Resolver Deregistration

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

In the `deregister` IX, a constraint is applied to the `resolver_access` , it will find the canonical bump and use it to perform the check. But this is not necessary, the corresponding bump has already been recorded in this account as `resolver_access.bump` .

```
100     #[account(
101         mut,
102         close = authority,
103         seeds = [RESOLVER_ACCESS_SEED, user.key().as_ref()],
104         bump,
105     )]
106     pub resolver_access: Account<'info, ResolverAccess>,
```

programs/whitelist/src/lib.rs#L100-L106

Fixed in the commit 57ec9206c8e9a560f5159c9a1658a6cc9f6e8a96.

Missing Access Control for Escrow Creation on the Destination Chain

Severity: Informational	Status: Acknowledged
Target: Smart Contract	Category: Access Control

Currently, anyone can create escrows on the destination chain, which is inconsistent with the design specification. Although this does not lead to any issues now, we recommend restricting this operation to legitimate resolvers only.

Missing sync_native CPI During Escrow Creation on the Destination Chain

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

In the `create` function, if the asset is native, after transferring tokens to the `escrow_ata`, the `sync_native` IX is invoked via CPI. However, this operation is missing on the destination chain. As a result, the `rescue_funds` IX on the destination chain will always fail (as the amount of the ATA is still 0) until a `sync_native` operation is explicitly performed.

```
98          if escrow_type == EscrowType::Src {
99              anchor_spl::token::sync_native(CpiContext::new(
100                 token_program.to_account_info(),
101                 anchor_spl::token::SyncNative {
102                     account: escrow_ata.to_account_info(),
103                 },
104             ))?;
105         }
```

common/src/escrow.rs#L98-L105

Recipient ATA Not Required When Token Is Native

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

```
160     if escrow.escrow_type() == EscrowType::Dst && escrow.asset_is_native()
        ↪ {
161         close_and_withdraw_native_ata(escrow, escrow_ata, recipient,
        ↪ token_program, seeds)?;
162     } else {
163         withdraw_and_close_token_ata(
164             ...
165             &recipient_ata
166                 .ok_or(EscrowError::MissingRecipientAta)?
167                 .to_account_info(),
168             ...
169         )?;
170     }
```

common/src/escrow.rs#L160-L170

In the `withdraw` function, if the escrow is created on the source chain, the recipient's ATA is still required to receive the tokens, even if the token is native.

It is recommended to make the `taker_ata` account optional in both the `public_withdraw` and `withdraw` IXs, and to remove the escrow type restriction from the `if` condition shown in the above code snippet.

Unchecked reward_limit in the cancel_order_by_resolver IX

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

```
160     let maker_amount = ctx.accounts.order_ata.to_account_info().lamports()
161         - std::cmp::min(cancellation_premium, reward_limit);
```

common/src/escrow.rs#L160-L161

In the `cancel_order_by_resolver` IX, the `reward_limit` parameter is used to determine the amount of lamports refunded to the maker. However, this value is not validated. A resolver can set it to an arbitrarily high value, ensuring that they always receive the full `cancellation_premium` amount as a reward.

It is recommended to add proper bounds checking for `reward_limit`.

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.

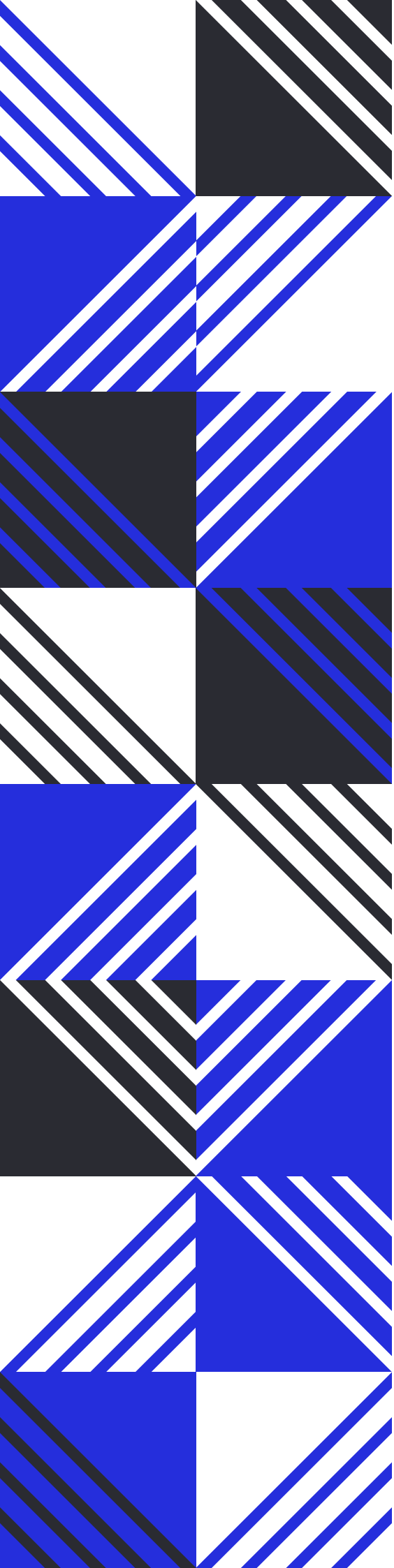
Offside Labs disclaims any liability for losses or damages resulting from the use of this report or from any future security breaches. The team strongly recommends that clients undertake multiple independent audits and implement a public bug bounty program to enhance the security of their smart contracts.

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