OFFSIDE LABS

Solana Cross-Chain

v1.1

Smart Contract Security Assessment

September 2025

Prepared for:

1inch

Prepared by:

Offside Labs

Gongyu Shi

Siji Feng

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 v1.1* smart contracts, starting on August 28th, 2025, and concluding on August 29th, 2025.

Project Overview

The 1inch Solana Cross-Chain project 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.

Compared with the previous version v1.0, this version includes minor refactoring (e.g., refactoring of instruction parameters and adjustments to the timelock design).

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 v1.1* 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: 6d2ae3df93e59ea4911f6f6ca866639535392704

We listed the files we have audited below:

- 1inch Solana Cross-Chain v1.1
  - common/src/escrow.rs
  - common/src/timelocks.rs
  - programs/cross-chain-escrow-dst/src/lib.rs
  - programs/cross-chain-escrow-dst/src/utils.rs
  - programs/cross-chain-escrow-src/src/lib.rs
  - programs/cross-chain-escrow-src/src/utils.rs
  - programs/cross-chain-escrow-src/src/merkle_tree.rs
  - programs/whitelist/src/lib.rs

Findings

The security audit revealed:

- 0 critical issue
- 0 high issue
- 1 medium issue
- 0 low issue

- 0 informational issue

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	Missing Timelock Validation May Lead to DoS	Medium	Acknowledged

4 Key Findings and Recommendations

4.1 Missing Timelock Validation May Lead to DoS

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

Description

A timelock mechanism is introduced in the protocol. The program assumes that timelock stages follow a strict chronological order:

```
SrcWithdrawal < SrcPublicWithdrawal < SrcCancellation <
↪ SrcPublicCancellation
```

However, the `create` instruction on both the source and destination sides accepts a user-supplied parameter `timelocks: [u64; 4]` without validating the ordering of the stages.

Impact

A malicious user can configure invalid or nonsensical timelock values, leading to DoS in subsequent operations.

Examples include:

- Overflow: Setting a very large timelock value that causes integer overflow in the `get` method of `Timelocks`. This results in all withdrawal or cancellation attempts failing with an `ArithmeticOverflow` error.

```
40  pub fn get(self, stage: Stage) -> std::result::Result<u32,
    ↪ ProgramError> {
41      let shift = (stage as usize) * STAGE_BIT_SIZE;
42      let deployed_at = (self.0 >> DEPLOYED_AT_OFFSET).as_u32();
43      let delta = ((self.0 >> shift) & U256::from(u32::MAX)).as_u32();
44      let result = deployed_at
45          .checked_add(delta)
46          .ok_or(ProgramError::ArithmeticOverflow)?;
47      Ok(result)
48  }
```

common/src/timelocks.rs#L40-L48

- Logic violation: Configuring `SrcWithdrawal > SrcCancellation` ensures that withdrawals always fail due to the enforced range check.

```
280     pub fn withdraw(ctx: Context<Withdraw>, secret: [u8; 32]) ->
        ↪ Result<()> {
281         let now = get_current_timestamp()?;
282
283         let timelocks = Timelocks(U256(ctx.accounts.escrow.timelocks));
284         require!(
285             now >= timelocks.get(Stage::SrcWithdrawal)?
286                 && now < timelocks.get(Stage::SrcCancellation)?,
287             EscrowError::InvalidTime
288         );
```

programs/cross-chain-escrow-src/src/lib.rs#L280-L288

A malicious order with invalid timelocks on the source chain can prevent a taker, who has already prepared the escrow on the destination chain, from claiming their assets on the source chain in time.

Recommendation

Add explicit validation during `create` to enforce that the timelocks follow a strictly increasing order, and ensure that the differences between adjacent timelocks are within reasonable bounds.

Mitigation Review Log

1inch Team: Basically, the protocol itself doesn't validate the ordering stages. The validation is performed on the backend level.

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.

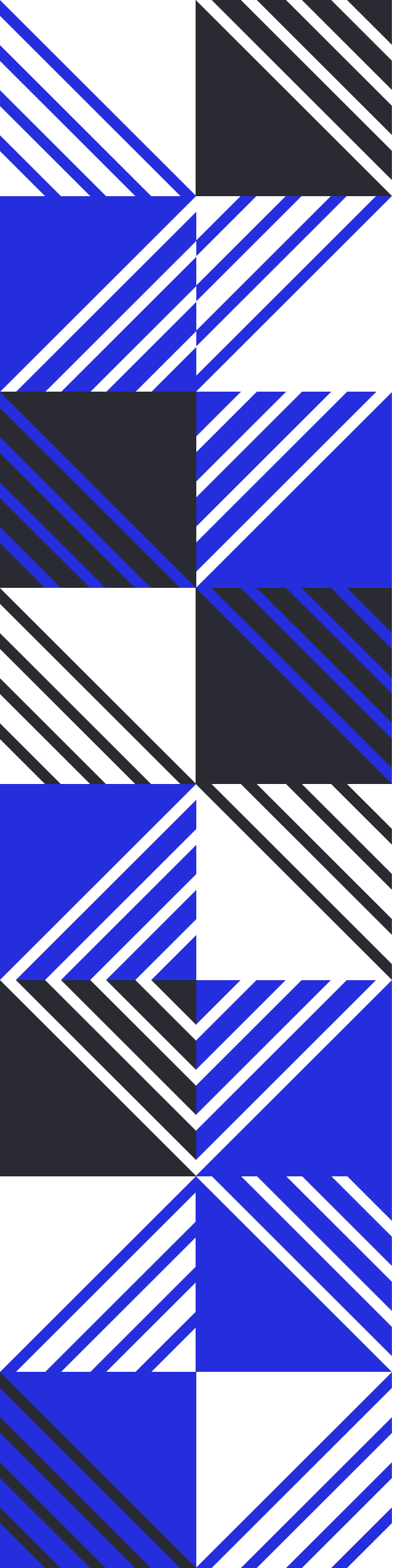
Offside Labs disclaims any liability for losses or damages resulting from the use of this report or from any future security breaches. The team strongly recommends that clients undertake multiple independent audits and implement a public bug bounty program to enhance the security of their smart contracts.

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