SHERLOCK

Security Review For 1inch

Collaborative Audit Prepared For:	**1inch**
Lead Security Expert(s):	**bin2chen** **krikolkk**
Date Audited:	**June 16 - June 20, 2025**
Final Commit:	**5595c32**

Introduction

TBA

Scope

Repository: 1inch/solana-crosschain-protocol

Audited Commit: 06ef2533ab5dd451a6d61bda677d54e6e628e21e

Final Commit: 5595c32680e5bc527300a2354aeb9f79733f0d1e

Files:

- common/src/constants.rs
- common/src/error.rs
- common/src/escrow.rs
- common/src/lib.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/error.rs
- programs/whitelist/src/lib.rs

Final Commit Hash

5595c32680e5bc527300a2354aeb9f79733f0d1e

Findings

Each issue has an assigned severity:

- Medium issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
- High issues are directly exploitable security vulnerabilities that need to be fixed.
- Low/Info issues are non-exploitable, informational findings that do not pose a security risk or impact the system's integrity. These issues are typically cosmetic or related to compliance requirements, and are not considered a priority for remediation.

Issues Found

High	Medium	Low/Info
1	0	2

Issues Not Fixed and Not Acknowledged

High	Medium	Low/Info
0	0	0

Issue H-1: incorrect value for escrow.amount

Source: https://github.com/sherlock-audit/2025-06-1inch-june-16th/issues/12

Summary

when `create_escrow()` , incorrectly setting `escrow.amount = order.amount`

Vulnerability Detail

when `create_escrow()` , we will save `escrow`

```
    pub fn create_escrow(
        ctx: Context<CreateEscrow>,
        amount: u64,
        dutch_auction_data: AuctionData,
        merkle_proof: Option<MerkleProof>,
    ) -> Result<()> {
...
        ctx.accounts.escrow.set_inner(EscrowSrc {
            order_hash: order.order_hash,
            hashlock,
            maker: order.creator,
            taker: ctx.accounts.taker.key(),
            token: order.token,
@>          amount: order.amount,
            safety_deposit: order.safety_deposit,
            withdrawal_start,
            public_withdrawal_start,
            cancellation_start,
            public_cancellation_start,
            rescue_start: order.rescue_start,
            asset_is_native: order.asset_is_native,
            dst_amount: get_dst_amount(order.dst_amount, &dutch_auction_data)?,
        });
```

Currently set `escrow.admount = order.amount` When `allow_multiple_fills = true`, the actual quantity should be the input parameter `amount`, not `order.amount`.

`CreateEscrow` uses the amount of the input parameter.

```
#[derive(Accounts)]
#[instruction(amount: u64, dutch_auction_data: AuctionData, merkle_proof:
↪  Option<MerkleProof>)]
pub struct CreateEscrow<'info> {
...
   #[account(
        init,
```

```
        payer = taker,
        space = constants::DISCRIMINATOR_BYTES + EscrowSrc::INIT_SPACE,
        seeds = [
            "escrow".as_bytes(),
            order.order_hash.as_ref(),
            &get_escrow_hashlock(
                order.hashlock,
                merkle_proof.clone()
            ),
            order.creator.as_ref(),
            taker.key().as_ref(),
            mint.key().as_ref(),
@>          amount.to_be_bytes().as_ref(),
            order.safety_deposit.to_be_bytes().as_ref(),
            order.rescue_start.to_be_bytes().as_ref(),
        ],
        bump,
    )]
    escrow: Box<Account<'info, EscrowSrc>>,
```

Subsequent pda addresses calculated elsewhere may not be correct Example. (Withdraw use escrow.amount = order.amount)

```
pub struct Withdraw<'info> {
    #[account(
        mut,
        seeds = [
            "escrow".as_bytes(),
            escrow.order_hash.as_ref(),
            escrow.hashlock.as_ref(),
            escrow.maker.as_ref(),
            escrow.taker.as_ref(),
            mint.key().as_ref(),
@>          escrow.amount.to_be_bytes().as_ref(),
            escrow.safety_deposit.to_be_bytes().as_ref(),
            escrow.rescue_start.to_be_bytes().as_ref(),
        ],
        bump,
    )]
    escrow: Box<Account<'info, EscrowSrc>>,
```

When cancel() refunds, it also gets the `escrow.amount()= order.amount`

Suppose:

order.amount = 100 , order.allow_multiple_fills = true alice call `(create_escrow = 50)`

1. CreateEscrow's seeds = [”escrow”+....50....]
2. Withdraw's seeds = [”escrow”+....100...]. => error

3. `cancel()` -> `close_and_withdraw_native_ata()` -> refunds `escrow.amount() == 100`

Impact

Operations such as `Withdraw/Cancel` cannot be performed.

Code Snippet

https://github.com/sherlock-audit/2025-06-1inch-june-16th/blob/b191eeb85bc9c3dfac25aaf6c3085d455c5ff0aa/solana-crosschain-protocol/programs/cross-chain-escrow-src/src/lib.rs#L209

Tool Used

Manual Review

Recommendation

```
    pub fn create_escrow(
...
        ctx.accounts.escrow.set_inner(EscrowSrc {
            order_hash: order.order_hash,
            hashlock,
            maker: order.creator,
            taker: ctx.accounts.taker.key(),
            token: order.token,
-           amount: order.amount,
+           amount: amount,
```

Discussion

byshape

Fixed in https://github.com/1inch/solana-crosschain-protocol/commit/6e0ba0911da36d56ea62967897e1e981a7ec33d9

bin2chen66

Fixed This PR has been changed to use `params.amount`

Issue L-1: dst_amount should use the percentage of order amount

Source: https://github.com/sherlock-audit/2025-06-1inch-june-16th/issues/13

Vulnerability Detail

when `create_escrow()` , we calculate `dst_amount`, the code is as follows:

```
    pub fn create_escrow(
...

        ctx.accounts.escrow.set_inner(EscrowSrc {
            order_hash: order.order_hash,
            hashlock,
            maker: order.creator,
            taker: ctx.accounts.taker.key(),
            token: order.token,
            amount: order.amount,
            safety_deposit: order.safety_deposit,
            withdrawal_start,
            public_withdrawal_start,
            cancellation_start,
            public_cancellation_start,
            rescue_start: order.rescue_start,
            asset_is_native: order.asset_is_native,
@>          dst_amount: get_dst_amount(order.dst_amount, &dutch_auction_data)?,
        });
```

Currently the whole `order.dst_amount` is used but when `allow_multiple_fills = true` , it may be created as multiple `escrow` It is recommended to use the actual percentage of the order amount.

Impact

dst_amount may be large

Code Snippet

https://github.com/sherlock-audit/2025-06-1inch-june-16th/blob/b191eeb85bc9c3dfac25aaf6c3085d455c5ff0aa/solana-crosschain-protocol/programs/cross-chain-escrow-src/src/lib.rs#L217

Tool Used

Manual Review

Recommendation

`get_dst_amount(amount * order.dst_amount / oder.amount,...)`

Discussion

byshape

Fixed in https://github.com/1inch/solana-crosschain-protocol/commit/5595c32680e5bc527300a2354aeb9f79733f0d1e

bin2chen66

Fixed This PR has been changed to use `order.dst_amount * params.amount / oder.amount`

Issue L-2: suggests to restrict mint == escrow.token

Source: https://github.com/sherlock-audit/2025-06-1inch-june-16th/issues/14

Vulnerability Detail

when `cancel_escrow()` , `taker` could specify `mint`

```
pub struct CancelEscrow<'info> {
...
@>  mint: Box<InterfaceAccount<'info, Mint>>,
    #[account(
        mut,
        seeds = [
            "escrow".as_bytes(),
            escrow.order_hash.as_ref(),
            escrow.hashlock.as_ref(),
            escrow.maker.as_ref(),
            taker.key().as_ref(),
@>          escrow.token.key().as_ref(),
            escrow.amount.to_be_bytes().as_ref(),
            escrow.safety_deposit.to_be_bytes().as_ref(),
            escrow.rescue_start.to_be_bytes().as_ref(),
        ],
        bump,
    )]
    escrow: Box<Account<'info, EscrowSrc>>,
    #[account(
        mut,
        associated_token::mint = mint,
        associated_token::authority = escrow,
        associated_token::token_program = token_program
    )]
```

However, there is currently no restriction on `mint == escrow.token`. a malicious user could specify the wrong mint, leaving `escrow.token` in the original `escrow_ata`.

Note: a similar problem exists in the `cancel_order_by_resolver`, `create_escrow` and `public_cancel_escrow` instructions.

Impact

using the wrong mint

Code Snippet

https://github.com/sherlock-audit/2025-06-1inch-june-16th/blob/b191eeb85bc9c3dfac25aaf6c3085d455c5ff0aa/solana-crosschain-protocol/programs/cross-chain-escrow-src/src/lib.rs#L293

Tool Used

Manual Review

Recommendation

```
    pub fn cancel_escrow(ctx: Context<CancelEscrow>) -> Result<()> {
...
+       require!(&ctx.accounts.mint.key() == &ctx.accounts.escrow.token.key() ,
↪  EscrowError::InvalidAccount);
```

Discussion

SevenSwen

It seems that using `ctx.accounts.mint.key()` in the seed derivation instead of `ctx.accounts.escrow.token.key()` should be sufficient. That way, we wouldn't need to use the additional constraint you're suggesting. What do you think about this approach?

bin2chen66

Yes. This is a good way.

byshape

Fixed in https://github.com/1inch/solana-crosschain-protocol/commit/882fbad8ff1378cda8c959428ed32e5a3fb6618f

bin2chen66

Fixed

This PR has been changed: create_escrow() -> CreateEscrow.order.seeds[..,mint,..] cancel_escrow() -> CancelEscrow.escrow.seeds[..,mint,..] public_cancel_escrow () -> PublicCancelEscrow.escrow.seeds[..,mint,..] cancel_order_by_resolver () -> CancelOrderbyResolver.order.seeds[..,mint,..]

Disclaimers

Sherlock does not provide guarantees nor warranties relating to the security of the project.

Usage of all smart contract software is at the respective users' sole risk and is the users' responsibility.