1inch Cross-chain Swap

Security Audit Report

March 26, 2024

Contents

1 Introduction

1.1 1inch Cross-chain Swap

`1inch Cross-chain Swap` introduces a two-party atomic swap mechanism, optimized for EVM-compatible chains with an option to execute a swap by a single party as an ordinary limit order. This reduces centralization, improves efficiency and security over existing bridging methods and enables direct transactions between users and market makers (resolvers) across chains, thereby encouraging broader web3 adoption.

1.2 Source Code

The following source code was reviewed during the audit:

- https://github.com/1inch/cross-chain-swap.git
- Commit ID: 5158cb8

And this is the final version representing all fixes implemented for the issues identified in the audit:

- https://github.com/1inch/cross-chain-swap.git
- Commit ID: b17cdb7

2 | Overall Assessment

This report has been compiled to identify issues and vulnerabilities within the `1inch Cross-chain Swap` project Throughout this audit, we identified a total of 3 issues spanning various severity levels. By employing auxiliary tool techniques to supplement our thorough manual code review, we have discovered the following findings.

Severity	Count	Acknowledged	Won't Do	Addressed
Critical	-	-	-	-
High	-	-	-	-
Medium	-	-	-	-
Low	2	-	-	2
Informational	1	-	-	1
Undetermined	-	-	-	-

3 | Vulnerability Summary

3.1 Overview

Click on an issue to jump to it, or scroll down to see them all.

~~L-1~~ Enhanced Active Period Check for EscrowDst::publicWithdraw()

~~L-2~~ Possible Overflow in TimelocksLib::get()

~~I-1~~ Meaningful Events for Key Operations

3.2 Security Level Reference

In web3 smart contract audits, vulnerabilities are typically classified into different severity levels based on the potential impact they can have on the security and functionality of the contract. Here are the definitions for critical-severity, high-severity, medium-severity, and low-severity vulnerabilities:

Severity	Description
C-X (Critical)	A severe security flaw with immediate and significant negative consequences. It poses high risks, such as unauthorized access, financial losses, or complete disruption of functionality. Requires immediate attention and remediation.
H-X (High)	Significant security issues that can lead to substantial risks. Although not as severe as critical vulnerabilities, they can still result in unauthorized access, manipulation of contract state, or financial losses. Prompt remediation is necessary.
M-X (Medium)	Moderately impactful security weaknesses that require attention and remediation. They may lead to limited unauthorized access, minor financial losses, or potential disruptions to functionality.
L-X (Low)	Minor security issues with limited impact. While they may not pose significant risks, it is still recommended to address them to maintain a robust and secure smart contract.
I-X (Informational)	Warnings and things to keep in mind when operating the protocol. No immediate action required.
U-X (Undetermined)	Identified security flaw requiring further investigation. Severity and impact need to be determined. Additional assessment and analysis are necessary.

3.3 Vulnerability Details

[L-1] Enhanced Active Period Check for EscrowDst::publicWithdraw()

Target	Category	IMPACT	LIKELIHOOD	STATUS
EscrowDst.sol	Business Logic	Low	Low	Addressed

In the `1inch Cross-chain Swap` protocol, the `EscrowDst` contract is utilized to initially lock the maker required token and amount and then unlock them with verification of the secret presented by 1inch relayer. According to the function annotations, the `publicWithdraw()` function allows anyone to unlock funds on behalf of the maker during the time period from `DstPublicWithdrawal` to `DstCancellation`. However, upon thorough examination, we notice an inconsistency between the implementation (line 55) and the comments (line 49).

EscrowDst::publicWithdraw()

```
46  /**
47  * @notice See {IEscrow-publicWithdraw}.
48  * @dev The function works on the time intervals highlighted with capital letters
        :
49  * ---- contract deployed --/-- finality --/-- private withdrawal --/-- PUBLIC
        WITHDRAWAL --/-- private cancellation ----
50  */
51  function publicWithdraw(bytes32 secret, Immutables calldata immutables)
52      external
53      onlyValidImmutables(immutables)
54      onlyValidSecret(secret, immutables)
55      onlyAfter(immutables.timelocks.get(TimelocksLib.Stage.DstPublicWithdrawal))
56  {
57      _uniTransfer(immutables.token.get(), immutables.maker.get(), immutables.
            amount);
58      _ethTransfer(msg.sender, immutables.safetyDeposit);
59  }
```

Remediation Improve the implementation of the publicWithdraw() function according to the comments.

[L-2] Possible Overflow in TimelocksLib::get()

Target	Category	IMPACT	LIKELIHOOD	STATUS
TimelocksLib.sol	Numeric Errors	Low	Low	Addressed

The TimelocksLib contract is designed to compactly store multiple timelocks in a uint256 variable. Especially, the get() function is utilized to retrieve the `stage` specified timelock. Upon thorough examination, we observe an overflow risk in the `(data + (data >> bitShift))& _TIMELOCK_MASK` (line 76) statement.

TimelocksLib::get()

```
72 function get(Timelocks timelocks, Stage stage) internal pure returns (uint256) {
73     unchecked {
74         uint256 data = Timelocks.unwrap(timelocks);
75         uint256 bitShift = uint256(stage) << 5;
76         return (data + (data >> bitShift)) & _TIMELOCK_MASK;
77     }
78 }
```

Remediation Mitigate the potential overflow risk in the get() function.

[I-1] Meaningful Events for Key Operations

Target	Category	IMPACT	LIKELIHOOD	STATUS
Multiple Contracts	Coding Practices	N/A	N/A	Addressed

The `event` feature is vital for capturing runtime dynamics in a contract. Upon emission, `events` store transaction arguments in logs, supplying external analytics and reporting tools with crucial information. They play a pivotal role in scenarios like modifying system-wide parameters or handling token operations.

However, in our examination of protocol dynamics, we observed that certain key operations lack meaningful events to document their changes. We highlight the representative routines below.

EscrowFactory::createDstEscrow()

```
115 function createDstEscrow(IEscrow.Immutables calldata dstImmutables, uint256
        srcCancellationTimestamp) external payable {
116     address token = dstImmutables.token.get();
117     uint256 nativeAmount = dstImmutables.safetyDeposit;
118     if (token == address(0)) {
119         nativeAmount += dstImmutables.amount;
```

```
120         }
121         if (msg.value != nativeAmount) revert InsufficientEscrowBalance();
123         IEscrow.Immutables memory immutables = dstImmutables;
124         immutables.timelocks = immutables.timelocks.setDeployedAt(block.timestamp);
125         // Check that the escrow cancellation will start not later than the
                cancellation time on the source chain.
126         if (immutables.timelocks.get(TimelocksLib.Stage.DstCancellation) >
                srcCancellationTimestamp) revert InvalidCreationTime();
128         bytes32 salt = immutables.hashMem();
129         address escrow = ESCROW_DST_IMPLEMENTATION.cloneDeterministic(salt, msg.value
                );
130         if (token != address(0)) {
131             IERC20(token).safeTransferFrom(msg.sender, escrow, immutables.amount);
132         }
133     }
```

Remediation Ensure the proper emission of meaningful events containing accurate information to promptly reflect state changes.

4 Appendix

4.1 About AstraSec

AstraSec is a blockchain security company that serves to provide high-quality auditing services for blockchain-based protocols. With a team of blockchain specialists, AstraSec maintains a strong commitment to excellence and client satisfaction. The audit team members have extensive audit experience for various famous DeFi projects. AstraSec's comprehensive approach and deep blockchain understanding make it a trusted partner for the clients.

4.2 Disclaimer

The information provided in this audit report is for reference only and does not constitute any legal, financial, or investment advice. Any views, suggestions, or conclusions in the audit report are based on the limited information and conditions obtained during the audit process and may be subject to unknown risks and uncertainties. While we make every effort to ensure the accuracy and completeness of the audit report, we are not responsible for any errors or omissions in the report.

We recommend users to carefully consider the information in the audit report based on their own independent judgment and professional advice before making any decisions. We are not responsible for the consequences of the use of the audit report, including but not limited to any losses or damages resulting from reliance on the audit report.

This audit report is for reference only and should not be considered a substitute for legal documents or contracts.

4.3 Contact

Name	AstraSec Team
Phone	+86 176 2267 4194
Email	contact@astrasec.ai
Twitter	https://twitter.com/AstraSecAI