1inch Fee Charging Update (2025-04) Security Audit

：1inch Fee Charging Update (2025-04)

May 14, 2025

Revision 1.0

ChainLight@Theori

Table of Contents

Executive Summary

Beginning on April 24, 2025, ChainLight of Theori conducted a three-day security audit of 1inch's fee charging update.

Key areas of focus included:

- **Surplus Fee Mechanism:** A significant portion of our effort was dedicated to analyzing the newly introduced surplus fee logic in `SimpleSettlement.sol`.
- **Validation of Refactored Code:** The audit verified the correct refactoring of `FeeTaker._postInteraction()` into a new internal function, `_getFeeAmounts()`, ensuring its logical integrity was maintained.
- **Review of Ancillary Changes:** Other changes, such as the removal of a fee basis point sanity check in `AmountGetterWithFee.sol`, were reviewed for security impacts.

Summary of Findings

The audit revealed a total of one issue, categorized by severity as follows:

- **Medium:** 1 issue

Audit Overview

Scope

Name	1inch Fee Charging Update (2025-04) Security Audit

Target / Version	• Git Repository (`1inch/fusion-protocol`): commit `c0197a5b4110e89a7a832fff8e95927e41564e17`
	  ◦ `contracts/SimpleSettlement.sol`
	• Git Repository (`1inch/limit-order-protocol`): commit `cf8e50eb24c01c7b276a30a5877840df35e66e67`
	  ◦ `contracts/extensions/FeeTaker.sol`
	  ◦ `contracts/extensions/AmountGetterWithFee.sol`
	  ◦ `contracts/extensions/AmountGetterBase.sol`

Application Type	Smart contracts
Lang. / Platforms	Smart contracts [Solidity]

Code Revision

N/A

Severity Categories

Severity	Description
Critical	The attack cost is low (not requiring much time or effort to succeed in the actual attack), and the vulnerability causes a high-impact issue. (e.g., Effect on service availability, Attacker taking financial gain)
High	An attacker can succeed in an attack which clearly causes problems in the service's operation. Even when the attack cost is high, the severity of the issue is considered "high" if the impact of the attack is remarkably high.
Medium	An attacker may perform an unintended action in the service, and the action may impact service operation. However, there are some restrictions for the actual attack to succeed.
Low	An attacker can perform an unintended action in the service, but the action does not cause significant impact or the success rate of the attack is remarkably low.
Informational	Any informational findings that do not directly impact the user or the protocol.
Note	Neutral information about the target that is not directly related to the project's safety and security.

Status Categories

Status	Description
Reported	ChainLight reported the issue to the client.
WIP	The client is working on the patch.
Patched	The client fully resolved the issue by patching the root cause.
Mitigated	The client resolved the issue by reducing the risk to an acceptable level by introducing mitigations.
Acknowledged	The client acknowledged the potential risk, but they will resolve it later.
Won't Fix	The client acknowledged the potential risk, but they decided to accept the risk.

Finding Breakdown by Severity

Category	Count	Findings
Critical	0	• N/A
High	0	• N/A
Medium	1	• `1INCH-FEECHARGING2504-001`
Low	0	• N/A
Informational	0	• N/A
Note	0	• N/A

Findings

Summary

#	ID	Title	Severity	Status
1	`1INCH-FEECHARGING2504-001`	Surplus Fee Bypass via `estimatedTakingAmount` Manipulation	Medium	Mitigated

#1 `1INCH-FEECHARGING2504-001` Surplus Fee Bypass via `estimatedTakingAmount` Manipulation

ID	Summary	Severity
`1INCH-FEECHARGING2504-001`	The surplus fee calculation in `SimpleSettlement._getFeeAmounts()` can be manipulated by providing an excessively large `estimatedTakingAmount`, allowing users to bypass surplus fees.	Medium

Description

The `SimpleSettlement._getFeeAmounts()` function calculates surplus fees by comparing actual taking amounts against the `estimatedTakingAmount` provided at the time of order creation. The smart contract itself cannot independently determine if this amount is appropriate, as it represents the expected amount based on the observed state when the user created the order.

If a user submits an extremely large value for the `estimatedTakingAmount`, such as `type(uint256).max`, this input causes the calculated surplus to be zero, irrespective of the actual amount ultimately taken, thereby negating any surplus fee.

Impact

Medium

Users can leverage this issue to evade surplus fees, leading to lower revenue from fees for the protocol.

Recommendation

Since this manipulation should be detectable through off-chain monitoring, whitelisted resolvers should take action against users (addresses) found to be intentionally manipulating the surplus fee calculation.

Remediation

Mitigated

Appropriate backend validations have been ensured to check if the `estimatedTakingAmount` for an order remains within a reasonable and expected range.

Revision History

Version	Date	Description
1.0	May 14, 2025	Initial version

Theori, Inc. (“We”) is acting solely for the client and is not responsible to any other party. Deliverables are valid for and should be used solely in connection with the purpose for which they were prepared as set out in our engagement agreement. You should not refer to or use our name or advice for any other purpose. The information (where appropriate) has not been verified. No representation or warranty is given as to accuracy, completeness or correctness of information in the Deliverables, any document, or any other information made available. Deliverables are for the internal use of the client and may not be used or relied upon by any person or entity other than the client. Deliverables are confidential and are not to be provided, without our authorization (preferably written), to entities or representatives of entities (including employees) that are not the client, including affiliates or representatives of affiliates of the client.