1inch Token Plugins Security Review

Pashov Audit Group

Conducted by: pashov

September 8th, 2023

Contents

1. About pashov

Krum Pashov, or **pashov**, is an independent smart contract security researcher. Having found numerous security vulnerabilities in various protocols, he does his best to contribute to the blockchain ecosystem and its protocols by putting time and effort into security research & reviews. Check his previous work <u>here</u> or reach out on Twitter <u>@pashovkrum</u>.

2. Disclaimer

A smart contract security review can never verify the complete absence of vulnerabilities. This is a time, resource and expertise bound effort where I try to find as many vulnerabilities as possible. I can not guarantee 100% security after the review or even if the review will find any problems with your smart contracts. Subsequent security reviews, bug bounty programs and on-chain monitoring are strongly recommended.

3. Introduction

A time-boxed security review of the **1inch Token Plugins** protocol was done by **pashov**, with a focus on the security aspects of the application's smart contracts implementation.

4. About 1inch Token Plugins

Token Plugins are an ERC20 system that enhances normal ERC20 tokens' functionality by externally tracking balances of users who have opted-in for a specific plugin. A basic example of a use case is if you'd like to track token shares without actually transferring tokens to a smart contract. Another example is farming/staking - a user that holds a token that supports plugins doesn't have to transfer the token from his wallet to the farming/staking contract - he just has to opt-in to the plugin (calling `addPlugin` method). This way, modularity is achieved, and users can participate in different side use-cases for a token they hold, without it leaving their wallet.

A plugin is implemented by inheriting from the `Plugin` contract, then specifying the target `token` and a `updateBalances` behavior. The `ERC20Plugins` contract is one that a token should inherit to support adding and removing external plugins, or another option is to use it in a combination with an ERC20 wrapper contract.

Previous security audit reports

Observations

The `_updateBalances` method does a low-level `call` to a `plugin` address that is user-controlled. It also has an `assembly` block marked with a `"memory-safe"` annotation. When an `_updateBalances` external call in `ERC20Plugins` fails, the transaction is not reverted, unless the failure was due to insufficient gas.

Privileged Roles & Actors

There are no privileged roles in the design of the plugins, only the normal user who can add and remove his plugins. The user can also make the `ERC20Plugins` contract call his own `Plugin` contract.

5. Risk Classification

Severity	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

5.1. Impact

- High - leads to a significant material loss of assets in the protocol or significantly harms a group of users.
- Medium - only a small amount of funds can be lost (such as leakage of value) or a core functionality of the protocol is affected.
- Low - can lead to any kind of unexpected behavior with some of the protocol's functionalities that's not so critical.

5.2. Likelihood

- High - attack path is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount of funds that can be stolen or lost.
- Medium - only a conditionally incentivized attack vector, but still relatively likely.
- Low - has too many or too unlikely assumptions or requires a significant stake by the attacker with little or no incentive.

5.3. Action required for severity levels

- Critical - Must fix as soon as possible (if already deployed)
- High - Must fix (before deployment if not already deployed)
- Medium - Should fix
- Low - Could fix

6. Security Assessment Summary

review commit hash - **585cad4891c3de7bd8692d366740ae72595fdefc**

No fixes implemented.

Scope

The following smart contracts were in scope of the audit:

- `interfaces/**`
- `libs/ReentrancyGuard`
- `Plugin`
- `ERC20Plugins`

7. Executive Summary

Over the course of the security review, pashov engaged with 1inch Token Plugins to review 1inch Token Plugins. In this period of time a total of **2** issues were uncovered.

Protocol Summary

Protocol Name	1inch Token Plugins
Date	September 8th, 2023

Findings Count

Severity	Amount
Low	2
Total Findings	2

Summary of Findings

ID	Title	Severity	Status
[L-01]	Using an immutable value as a gas limit might be dangerous	Low	Resolved
[L-02]	It's possible to use a flawed compiler version	Low	Resolved

8. Findings

8.1. Low Findings

[L-01] Using an `immutable` value as a gas limit might be dangerous

The `pluginCallGasLimit` is a value that is set in the constructor of `ERC20Plugins` and is `immutable`. This value is used to cap the gas allowed for an external low-level call to use. The problem with this is that previous Ethereum hardforks have changed gas costs of commonly used opcodes (for example with `EIP-150`) - this happening again can result in the `pluginCallGasLimit` value being insufficient to execute the operations needed by the contract. Consider making the value mutable or removing it altogether.

[L-02] It's possible to use a flawed compiler version

Solidity version 0.8.13 & 0.8.14 have a security vulnerability related to assembly blocks that write to memory, which are present in `ERC20Plugins` contract. The issue is fixed in version 0.8.15 and is explained <u>here</u>. While the problem is with the legacy optimizer (not the yul IR pipeline that you are using), it is still correct to enforce latest Solidity version (0.8.21) or at least the one enforced in other popular library code as OpenZeppelin (0.8.19). Applies for the whole codebase.

```
-pragma solidity ^0.8.0;
+pragma solidity ^0.8.19;
```