1inch Exchange

Security Assessment

November 16th, 2020

By :

Angelos Apostolidis @ CertiK

angelos.apostolidis@certik.org

Camden Smallwood @ CertiK

camden.smallwood@certik.org

Alex Papageorgiou @ CertiK

alex.papageorgiou@certik.org

Disclaimer

What is a CertiK report?

- A document describing in detail an in depth analysis of a particular piece(s) of source code provided to CertiK by a Client.
- An organized collection of testing results, analysis and inferences made about the structure, implementation and overall best practices of a particular piece of source code.
- Representation that a Client of CertiK has indeed completed a round of auditing with the intention to increase the quality of the company/product's IT infrastructure and or source code.

Overview

Project Summary

Project Name	1inch Exchange
Description	The audit is based on a subset of the 1nch exchange core contracts.
Platform	Ethereum; Solidity
Codebase	GitHub Repository
Commits	1. aa1d1c54546f38b912a24722134ab0c2ae94860d 2. 37a41846c8587be5eb6e41a11c40226eb412073b 3. 5a3ca0af129105b926744ff3804e45699697dedb

Audit Summary

Delivery Date	Nov. 16, 2020
Method of Audit	Static Analysis, Manual Review
Consultants Engaged	3
Timeline	Oct. 24, 2020 - Nov. 16 2020

Vulnerability Summary

Total Issues	8
Total Medium	1
Total Informational	7

Executive Summary

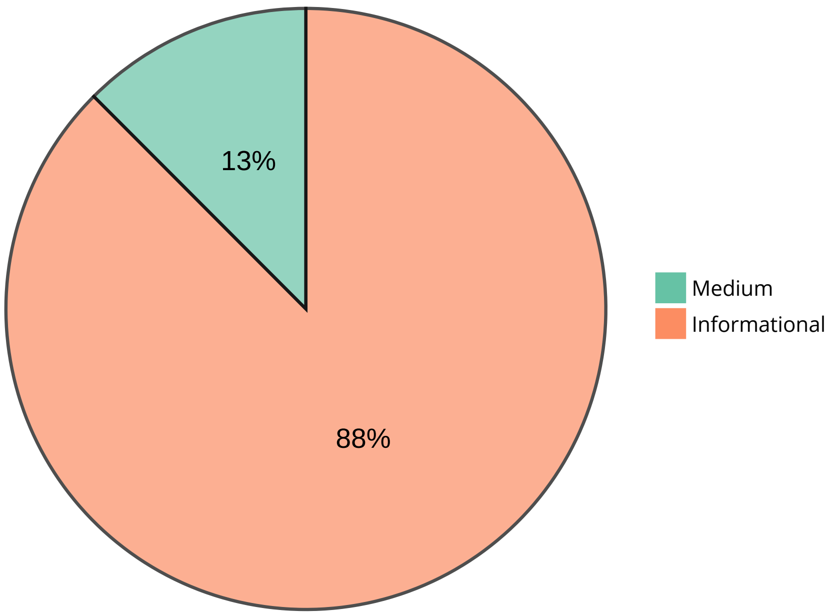
We were approached by 1nch to conduct an audit of a subset from their exchange core contracts, 1inch Exchange. Our audit was able to pinpoint some sections where the codebase could be improved optimization-wise, however only a single medium vulnerability was pinpointed that was clarified by the developers to be desired functionality and should not result in an exploitable attack vector.

Files In Scope

ID	Contract	Location
GDC	GasDiscountCalculator	contracts/GasDiscountCalculator.sol
OIE	OneInchExchange	contracts/OneInchExchange.sol
OIF	OneInchFlags	contracts/OneInchFlags.sol
RRP	RevertReasonParser	contracts/helpers/RevertReasonParser.sol
UERC	UniERC20	contracts/helpers/UniERC20.sol

Findings

Finding Summary



ID	Title	Type	Severity	Resolved
OIN-01	Unlocked Solidity compiler versions	Language Specific	Informational	(!)
OIN-02	Non-optimal flags storage type	Optimization	Informational	(!)
OIN-03	Non-optimal extra ETH comparison logic	Logic	Informational	✓
OIN-04	Usage of if-revert pattern over require	Implementation	Informational	(!)✓
OIN-05	Redundant function calls in `swap` implementation	Performance	Informational	✓
OIN-06	Empty reason for non-Chi tokens on out-of-gas exception	Logic	Informational	✓
OIN-07	Pausable contract not externally unpausable	Implementation	Medium	(!)✓
OIN-08	Unnecessary `else` statement in `uniBalanceOf` implementation	Coding Style	Informational	(!)

OIN-01: Unlocked Solidity compiler versions

Type	Severity	Location
Language Specific	Informational	GasDiscountCalculator.sol L3, OneInchExchange.sol L3, OneInchFlags.sol L3, RevertReasonParser.sol L3, UniERC20.sol L3

Description:

An unlocked compiler version in the source code of the contract permits the user to compile it at or above a particular version. This, in turn, leads to differences in the generated bytecode between compilations due to differing compiler version numbers.This can lead to an ambiguity when debugging as compiler specific bugs may occur in the codebase that would be hard to identify over a span of multiple compiler versions rather than a specific one.

Recommendation:

We advise that the compiler version is instead locked at version `0.6.12` which is referenced by the `truffle-config.json` file of the repository.

Alleviation:

The 1inch team decided to stick to the current compiler specification convention they utilize.

OIN-02: Non-optimal flags storage type

Type	Severity	Location
Optimization	Informational	OneInchExchange.sol L31

Description:

The `flags` variable could be reduced in size to be tight-packed with the address declaration that follows it by setting its size to `uint96` permitting 96 different flags. This would reduce the total storage cost of the `SwapDescription` to 7 from 8 if we do not factor in the `permit` variable.

Recommendation:

While it can be stored in a smaller data type, we advise that `uint96` is used to ensure that the full space of the storage slot is utilized and that the unpacking operations are simpler with regards to unpadding the value.

The value never remains at rest and as such a gas optimization should be observed if this exhibit is applied as the tight packing happens off-chain and only the unpacking operation needs to be conducted on-chain.

Alleviation:

After conducting gas tests in tandem with 1inch, we deduced that this exhibit would instead cause a minor gas increase instead of reduction and as such, the 1inch team decided to not apply this exhibit.

OIN-03: Non-optimal extra ETH comparison logic

Type	Severity	Location
Logic	Informational	OneInchExchange.sol L64

Description:

The `isExtraEthAllowed` flag alludes to a "permittance" rather than an enforcement, which the `require` condition of the linked line does.

Recommendation:

We advise that the greater-than (`>`) operator is changed to a greater-than-or-equal (`>=`) operator.

Alleviation:

The 1inch team instead decided to rename the flag to an enforcement rather than permittance properly representing the conditional being imposed.

OIN-04: Usage of if-revert pattern over require

Type	Severity	Location
Implementation	Informational	OneInchExchange.sol L94-L96

Description:

The `swap` function in the `OneInchExchange` contract contains the usage of the if-revert pattern over using a require clause, which may be deprecated in the future:

```
if (!desc.flags.isDiscountChi()) {
  revert(RevertReasonParser.parse(reason, "Swap failed: "));
}
```

Recommendation:

The if-revert pattern should be updated to require clause:

```
require(
  desc.flags.isDiscountChi(),
  RevertReasonParser.parse(reason, "Swap failed: ")
);
```

Alleviation:

The `if-revert` pattern is applied here to optimize gas cost by not executing the `parse` function regardless of whether the condition would fail. As such, the code segment should remain as is.

OIN-05: Redundant function calls in `swap` implementation

Type	Severity	Location
Performance	Informational	OneInchExchange.sol L62, L64, L68, L76, L89, L94

Description:

The `swap` function in the `OneInchExchange` contract makes multiple calls to the `isETH` function from the `UniERC20` contract and the `isPartialFill` and `isDiscountChi` functions in the `OneInchFlags` library, which is inefficient compared to storing their results in intermediate local variables and referencing those instead.

Recommendation:

Consider storing the result of each function call in an intermediate local variable and using those in place of any duplicate function calls in order to save on the overall cost of gas:

```
bool isETH = desc.srcToken.isETH();
bool isPartialFill = desc.flags.isPartialFill();
bool isDiscountChi = desc.flags.isDiscountChi();
```

Alleviation:

The 1inch team heeded our recommendations and applied some more intermediate variables to further optimize the gas cost of the function.

OIN-06: Empty reason for non-Chi tokens on out-of-gas exception

Type	Severity	Location
Logic	Informational	OneInchExchange.sol L95

Description:

Whenever an out-of-gas exception occurs and the user is not using a Chi token as a substitute for gas, the `reason` variable will be empty (0x).

Recommendation:

This case should be handled with a special error message and included in the `RevertReasonParser` library.

Alleviation:

The `RevertReasonParser` library was updated to conform to the Solidity `0.8.X` error handling style, returning `"Unknown()"` if the error is not parse-able as is the case when running out of gas.

OIN-07: Pausable contract not externally unpausable

Type	Severity	Location
Implementation	Medium	OneInchExchange.sol L156-L158

Description:

The `_unpause` function in the `Pausable` contract is not exposed, meaning that an accidental invocation of the `pause` function is irreversible.

Recommendation:

We advise that a similar function is coded that exports the `_unpause` function.

Alleviation:

The 1inch team responded with the following transcript:

> We want to pause the contract in case we find the vulnerability but we don’t want to have the ability to unpause it to prevent users’ funds loss.

OIN-08: Unnecessary `else` statement in `uniBalanceOf` implementation

Type	Severity	Location
Coding Style	Informational	UniERC20.sol L21-L27

Description:

The `uniBalanceOf` function can skip the `else`-block implementation and directly return the specific value after the `if`-condition is not met.

Recommendation:

We advise the team to change the function as seen below:

```
function uniBalanceOf(IERC20 token, address account) internal view
returns (uint256) {
  if (isETH(token)) {
    return account.balance;
  }
  return token.balanceOf(account);
}
```

Alleviation:

The 1inch team took note of this exhibit however decided to stick to the `if-else` style.