LimitSwap audit

28 March 2021, Igor Gulamov

Introduction

Igor Gulamov conducted the audit of 1inch LimitSwap smart contracts.

This review was performed by an independent reviewer under fixed rate.

Scope

Flatten version of `LimitSwap.sol`.

Issues

We found one critical no major issues.

Critical issue is fixed. We consider `LimitSwap.sol` from commit **025e3f78e21287cb37633e23f06726d0a907aa3c** as safe from the informational security point of view.

Critical

Everybody can transfer allowed assets

```
contract ERC20Proxy {
    using SafeERC20 for IERC20;

    // func_0000jYAHF(address,address,uint256,address) = transferFrom + 1 =
0x8d076e86
    function func_0000jYAHF(address from, address to, uint256 amount, IERC20
token) external {
        token.safeTransferFrom(from, to, amount);
    }
}


contract ERC721Proxy {
    using SafeERC20 for IERC20;

    // func_4002L9TKH(address,address,uint256,address) = transferFrom + 2 =
0x8d076e87
    function func_4002L9TKH(address from, address to, uint256 tokenId, IERC721
token) external {
        token.transferFrom(from, to, tokenId);
    }

    // func_2000nVqcj(address,address,uint256,address) == transferFrom + 3 =
0x8d076e88
    function func_2000nVqcj(address from, address to, uint256 tokenId, IERC721
token) external {
        token.safeTransferFrom(from, to, tokenId);
    }
```

```
}


contract ERC1155Proxy {
    using SafeERC20 for IERC20;

    // func_7000ksXmS(address,address,uint256,address,uint256) == transferFrom +
4 = 0x8d076e89
    function func_7000ksXmS(address from, address to, uint256 amount, IERC1155
token, uint256 tokenId) external {
        token.safeTransferFrom(from, to, tokenId, amount, "");
    }
}
```

Any asset, allowed to spend for this contract, will be stolen by bots. We recommend allowing calls only for whitelisted safe contracts.

Fixed at [025e3f78e21287cb37633e23f06726d0a907aa3c]

Warnings

Probably reentrancy on multiple points of `LimitSwap`

We recommend making `LimitSwap` non-reentrant because a lot of arbitrary calls bring a huge attack surface.

Pobably buffer overflow

```
library ArrayParser {
    function sliceBytes32(bytes memory data, uint256 start) internal pure
returns(bytes32 result) {
        assembly {
            result := mload(add(add(data, 0x20), start))
        }
    }

    function patchBytes32(bytes memory data, uint256 start, bytes32 value)
internal pure returns(bytes memory result) {
        assembly {
            mstore(add(add(data, 0x20), start), value)
        }
        return data;
    }
}
```

We recommend to verify, that length of the data is at least 32 bytes.

Fixed at [025e3f78e21287cb37633e23f06726d0a907aa3c]

Comments

Unnecessary SafeMath import

```
contract GetMakerAmountHelper {
    using SafeMath for uint256;
```

Severity Terms

Comment

Comment issues are generally subjective in nature, or potentially deal with topics like "best practices" or "readability". Comment issues in general will not indicate an actual problem or bug in code.

The maintainers should use their own judgment as to whether addressing these issues improves the codebase.

Warning

Warning issues are generally objective in nature but do not represent actual bugs or security problems.

These issues should be addressed unless there is a clear reason not to.

Major

Major issues will be things like bugs or security vulnerabilities. These issues may not be directly exploitable, or may require a certain condition to arise in order to be exploited.

Left unaddressed these issues are highly likely to cause problems with the operation of the contract or lead to a situation which allows the system to be exploited in some way.

Critical

Critical issues are directly exploitable bugs or security vulnerabilities.

Left unaddressed these issues are highly likely or guaranteed to cause major problems or potentially a full failure in the operations of the contract.