Guard

Smart contracts security assessment

Final report

Tariff: Standard

Tradescrow

March 2022

0xguard.com

hello@0xguard.com

Contents

Introduction

The report has been prepared for the Tradescrow team.

The audited code is available at @tradescrow/contracts Github repository and was audited after commit e9be54e. A recheck has been done after commit c19630e. Users must check if they are interacting with the same contract as was audited.

The audited contract is a secure exchanger for ERC721, ERC1155, ERC20 tokens and native coins. The contract takes a commission for creating swap offers.

ERC721, ERC1155, ERC20 interfaces is implemented with the use of OpenZeppelin libraries, which is considered the best practice.

Name	Tradescrow
Audit date	2022-03-18 - 2022-03-18
Language	Solidity
Platform	Harmony

Contracts checked

Name	Address
Tradescrow.sol	

Procedure

We perform our audit according to the following procedure:

Automated analysis

- Scanning the project's smart contracts with several publicly available automated Solidity analysis tools

- Manual verification (reject or confirm) all the issues found by the tools

Manual audit

- Manually analyse smart contracts for security vulnerabilities
- Smart contracts' logic check

Known vulnerabilities checked

Title	Check result
Unencrypted Private Data On-Chain	passed
Code With No Effects	passed
Message call with hardcoded gas amount	passed
Typographical Error	passed
DoS With Block Gas Limit	passed
Presence of unused variables	passed
Incorrect Inheritance Order	passed
Requirement Violation	passed
Weak Sources of Randomness from Chain Attributes	passed
Shadowing State Variables	passed
Incorrect Constructor Name	passed
Block values as a proxy for time	passed
Authorization through tx.origin	passed
DoS with Failed Call	passed
Delegatecall to Untrusted Callee	passed
Use of Deprecated Solidity Functions	passed

Assert Violation	passed
State Variable Default Visibility	passed
Reentrancy	passed
Unprotected SELFDESTRUCT Instruction	passed
Unprotected Ether Withdrawal	passed
Unchecked Call Return Value	passed
Floating Pragma	Not passed
Outdated Compiler Version	passed
Integer Overflow and Underflow	passed
Function Default Visibility	passed

Classification of issue severity

High severity — High severity issues can cause a significant or full loss of funds, change of contract ownership, major interference with contract logic. Such issues require immediate attention.

Medium severity — Medium severity issues do not pose an immediate risk, but can be detrimental to the client's reputation if exploited. Medium severity issues may lead to a contract failure and can be fixed by modifying the contract state or redeployment. Such issues require attention.

Low severity — Low severity issues do not cause significant destruction to the contract's functionality. Such issues are recommended to be taken into consideration.

Issues

High severity issues

1. Owner can pause the contract (FIXED) (Tradescrow.sol)

Contract may be paused by the owner, thereby freezing the user tokens sent for exchange.

```
function cancelSwap(uint256 swapId) external nonReentrant whenNotPaused {
  ...
}
```

Recommendation: Use a multisig wallet and put it behind a Timelock contract by giving it owner rights. After this the severity of the issue may be lowered.

Medium severity issues

1. Tokens with fees on transfers are not supported (FIXED) (Tradescrow.sol)

If a token with a transfer fee is added to the swap, after confirming the exchange, the contract will send more tokens than it received.

Recommendation: Check actual amount of deposited tokens by checking balance before and after token transfers in the proposeSwap() and initiateSwap() functions.

Low severity issues

1. The rest of ETH on the contract (FIXED) (Tradescrow.sol)

The function withdrawFees() leaves 1e18 native for gas. The situation where the contract will pay for gas is impossible.

Conclusion

Tradescrow Tradescrow.sol contract was audited. 1 high, 1 medium, 1 low severity issues were found.

Update: all issues were fixed in the update.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability)set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without 0xGuard prior written consent.

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts 0xGuard to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Slither output

```
Reentrancy in Tradescrow.acceptSwap(uint256) (Tradescrow.sol#182-200):
        External calls:
        - safeMultipleTransfersFrom(address(this),_swaps[swapId].initiator.addr,_swaps[s
wapId].target) (Tradescrow.sol#189)
                - returndata = address(token).functionCall(data,SafeERC20: low-level
call failed) (@openzeppelin\contracts\token\ERC20\utils\SafeERC20.sol#93)
                - (success,returndata) = target.call{value: value}(data) (@openzeppelin
\contracts\utils\Address.sol#132)
                - IERC721(offer.nfts[i].addr).safeTransferFrom(from,to,offer.nfts[i].id,)
(Tradescrow.sol#322)
                - IERC1155(offer.nfts[i].addr).safeTransferFrom(from,to,offer.nfts[i].id,o
ffer.nfts[i].amount,) (Tradescrow.sol#324)
                -
IERC20(offer.coins[i_scope_0].addr).safeTransfer(to,offer.coins[i_scope_0].amount)
(Tradescrow.sol#329)
                - IERC20(offer.coins[i_scope_0].addr).safeTransferFrom(from,to,offer.coins
[i_scope_0].amount) (Tradescrow.sol#331)
        - safeMultipleTransfersFrom(address(this),_swaps[swapId].target.addr,_swaps[swap
Id].initiator) (Tradescrow.sol#192)
                - returndata = address(token).functionCall(data,SafeERC20: low-level
call failed) (@openzeppelin\contracts\token\ERC20\utils\SafeERC20.sol#93)
                - (success,returndata) = target.call{value: value}(data) (@openzeppelin
\contracts\utils\Address.sol#132)
                - IERC721(offer.nfts[i].addr).safeTransferFrom(from,to,offer.nfts[i].id,)
(Tradescrow.sol#322)
                - IERC1155(offer.nfts[i].addr).safeTransferFrom(from,to,offer.nfts[i].id,o
ffer.nfts[i].amount,) (Tradescrow.sol#324)
                -
IERC20(offer.coins[i_scope_0].addr).safeTransfer(to,offer.coins[i_scope_0].amount)
(Tradescrow.sol#329)
                - IERC20(offer.coins[i_scope_0].addr).safeTransferFrom(from,to,offer.coins
[i_scope_0].amount) (Tradescrow.sol#331)
        External calls sending eth:
        - safeMultipleTransfersFrom(address(this),_swaps[swapId].initiator.addr,_swaps[s
wapId].target) (Tradescrow.sol#189)
                - (success,returndata) = target.call{value: value}(data) (@openzeppelin
\contracts\utils\Address.sol#132)
```

```
        - safeMultipleTransfersFrom(address(this),_swaps[swapId].target.addr,_swaps[swap
Id].initiator) (Tradescrow.sol#192)
                - (success,returndata) = target.call{value: value}(data) (@openzeppelin
\contracts\utils\Address.sol#132)
        - transferNative(_swaps[swapId].initiator,_swaps[swapId].target)
(Tradescrow.sol#194)
                - to.addr.transfer(native) (Tradescrow.sol#341)
        - transferNative(_swaps[swapId].target,_swaps[swapId].initiator)
(Tradescrow.sol#195)
                - to.addr.transfer(native) (Tradescrow.sol#341)
        State variables written after the call(s):
        - delete _swaps[swapId] (Tradescrow.sol#199)
Reentrancy in Tradescrow.cancelSwap(uint256) (Tradescrow.sol#208-232):
        External calls:
        - safeMultipleTransfersFrom(address(this),_swaps[swapId].initiator.addr,_swaps[s
wapId].initiator) (Tradescrow.sol#217)
                - returndata = address(token).functionCall(data,SafeERC20: low-level
call failed) (@openzeppelin\contracts\token\ERC20\utils\SafeERC20.sol#93)
                - (success,returndata) = target.call{value: value}(data) (@openzeppelin
\contracts\utils\Address.sol#132)
                - IERC721(offer.nfts[i].addr).safeTransferFrom(from,to,offer.nfts[i].id,)
(Tradescrow.sol#322)
                - IERC1155(offer.nfts[i].addr).safeTransferFrom(from,to,offer.nfts[i].id,o
ffer.nfts[i].amount,) (Tradescrow.sol#324)
                -
IERC20(offer.coins[i_scope_0].addr).safeTransfer(to,offer.coins[i_scope_0].amount)
(Tradescrow.sol#329)
                - IERC20(offer.coins[i_scope_0].addr).safeTransferFrom(from,to,offer.coins
[i_scope_0].amount) (Tradescrow.sol#331)
        - safeMultipleTransfersFrom(address(this),_swaps[swapId].target.addr,_swaps[swap
Id].target) (Tradescrow.sol#221)
                - returndata = address(token).functionCall(data,SafeERC20: low-level
call failed) (@openzeppelin\contracts\token\ERC20\utils\SafeERC20.sol#93)
                - (success,returndata) = target.call{value: value}(data) (@openzeppelin
\contracts\utils\Address.sol#132)
                - IERC721(offer.nfts[i].addr).safeTransferFrom(from,to,offer.nfts[i].id,)
(Tradescrow.sol#322)
                - IERC1155(offer.nfts[i].addr).safeTransferFrom(from,to,offer.nfts[i].id,o
ffer.nfts[i].amount,) (Tradescrow.sol#324)
                -
IERC20(offer.coins[i_scope_0].addr).safeTransfer(to,offer.coins[i_scope_0].amount)
(Tradescrow.sol#329)
```

```
                - IERC20(offer.coins[i_scope_0].addr).safeTransferFrom(from,to,offer.coins
[i_scope_0].amount) (Tradescrow.sol#331)
        External calls sending eth:
        - safeMultipleTransfersFrom(address(this),_swaps[swapId].initiator.addr,_swaps[s
wapId].initiator) (Tradescrow.sol#217)
                - (success,returndata) = target.call{value: value}(data) (@openzeppelin
\contracts\utils\Address.sol#132)
        - transferNative(_swaps[swapId].initiator,_swaps[swapId].initiator)
(Tradescrow.sol#218)
                - to.addr.transfer(native) (Tradescrow.sol#341)
        - safeMultipleTransfersFrom(address(this),_swaps[swapId].target.addr,_swaps[swap
Id].target) (Tradescrow.sol#221)
                - (success,returndata) = target.call{value: value}(data) (@openzeppelin
\contracts\utils\Address.sol#132)
        - transferNative(_swaps[swapId].target,_swaps[swapId].target)
(Tradescrow.sol#222)
                - to.addr.transfer(native) (Tradescrow.sol#341)
        State variables written after the call(s):
        - delete _swaps[swapId] (Tradescrow.sol#230)
Reentrancy in Tradescrow.initiateSwap(uint256,Tradescrow.Offer)
(Tradescrow.sol#140-171):
        External calls:
        - safeMultipleTransfersFrom(address(msg.sender),address(this),offer)
(Tradescrow.sol#148-152)
                - returndata = address(token).functionCall(data,SafeERC20: low-level
call failed) (@openzeppelin\contracts\token\ERC20\utils\SafeERC20.sol#93)
                - (success,returndata) = target.call{value: value}(data) (@openzeppelin
\contracts\utils\Address.sol#132)
                - IERC721(offer.nfts[i].addr).safeTransferFrom(from,to,offer.nfts[i].id,)
(Tradescrow.sol#322)
                - IERC1155(offer.nfts[i].addr).safeTransferFrom(from,to,offer.nfts[i].id,o
ffer.nfts[i].amount,) (Tradescrow.sol#324)
                -
IERC20(offer.coins[i_scope_0].addr).safeTransfer(to,offer.coins[i_scope_0].amount)
(Tradescrow.sol#329)
                - IERC20(offer.coins[i_scope_0].addr).safeTransferFrom(from,to,offer.coins
[i_scope_0].amount) (Tradescrow.sol#331)
        External calls sending eth:
        - safeMultipleTransfersFrom(address(msg.sender),address(this),offer)
(Tradescrow.sol#148-152)
                - (success,returndata) = target.call{value: value}(data) (@openzeppelin
```

```
\contracts\utils\Address.sol#132)
        State variables written after the call(s):
        - _swaps[swapId].target.nfts.push(offer.nfts[i]) (Tradescrow.sol#155)
        - _swaps[swapId].target.coins.push(offer.coins[i_scope_0]) (Tradescrow.sol#158)
        - _swaps[swapId].target.native = msg.value - fee (Tradescrow.sol#161)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities

Tradescrow.withdrawFees(address) (Tradescrow.sol#261-267) contains a tautology or
contradiction:
        - require(bool,string)(address(this).balance - _native - 1000000000000000000 >=
0,Tradescrow: No available fees) (Tradescrow.sol#264)
Tradescrow.isNotEmpty(Tradescrow.Offer) (Tradescrow.sol#297-303) contains a tautology or
contradiction:
        - offer.nfts.length != 0 || offer.coins.length != 0 || offer.native >= 0
(Tradescrow.sol#299)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#tautology-or-
contradiction

Reentrancy in Tradescrow.acceptSwap(uint256) (Tradescrow.sol#182-200):
        External calls:
        - safeMultipleTransfersFrom(address(this),_swaps[swapId].initiator.addr,_swaps[s
wapId].target) (Tradescrow.sol#189)
                - returndata = address(token).functionCall(data,SafeERC20: low-level
call failed) (@openzeppelin\contracts\token\ERC20\utils\SafeERC20.sol#93)
                - (success,returndata) = target.call{value: value}(data) (@openzeppelin
\contracts\utils\Address.sol#132)
                - IERC721(offer.nfts[i].addr).safeTransferFrom(from,to,offer.nfts[i].id,)
(Tradescrow.sol#322)
                - IERC1155(offer.nfts[i].addr).safeTransferFrom(from,to,offer.nfts[i].id,o
ffer.nfts[i].amount,) (Tradescrow.sol#324)
                -
IERC20(offer.coins[i_scope_0].addr).safeTransfer(to,offer.coins[i_scope_0].amount)
(Tradescrow.sol#329)
                - IERC20(offer.coins[i_scope_0].addr).safeTransferFrom(from,to,offer.coins
[i_scope_0].amount) (Tradescrow.sol#331)
        - safeMultipleTransfersFrom(address(this),_swaps[swapId].target.addr,_swaps[swap
Id].initiator) (Tradescrow.sol#192)
                - returndata = address(token).functionCall(data,SafeERC20: low-level
call failed) (@openzeppelin\contracts\token\ERC20\utils\SafeERC20.sol#93)
                - (success,returndata) = target.call{value: value}(data) (@openzeppelin
```

```
\contracts\utils\Address.sol#132)
                - IERC721(offer.nfts[i].addr).safeTransferFrom(from,to,offer.nfts[i].id,)
(Tradescrow.sol#322)
                - IERC1155(offer.nfts[i].addr).safeTransferFrom(from,to,offer.nfts[i].id,o
ffer.nfts[i].amount,) (Tradescrow.sol#324)
                -
IERC20(offer.coins[i_scope_0].addr).safeTransfer(to,offer.coins[i_scope_0].amount)
(Tradescrow.sol#329)
                - IERC20(offer.coins[i_scope_0].addr).safeTransferFrom(from,to,offer.coins
[i_scope_0].amount) (Tradescrow.sol#331)
        External calls sending eth:
        - safeMultipleTransfersFrom(address(this),_swaps[swapId].initiator.addr,_swaps[s
wapId].target) (Tradescrow.sol#189)
                - (success,returndata) = target.call{value: value}(data) (@openzeppelin
\contracts\utils\Address.sol#132)
        - safeMultipleTransfersFrom(address(this),_swaps[swapId].target.addr,_swaps[swap
Id].initiator) (Tradescrow.sol#192)
                - (success,returndata) = target.call{value: value}(data) (@openzeppelin
\contracts\utils\Address.sol#132)
        - transferNative(_swaps[swapId].initiator,_swaps[swapId].target)
(Tradescrow.sol#194)
                - to.addr.transfer(native) (Tradescrow.sol#341)
        State variables written after the call(s):
        - transferNative(_swaps[swapId].initiator,_swaps[swapId].target)
(Tradescrow.sol#194)
                - _native -= from.native (Tradescrow.sol#338)
Reentrancy in Tradescrow.acceptSwap(uint256) (Tradescrow.sol#182-200):
        External calls:
        - safeMultipleTransfersFrom(address(this),_swaps[swapId].initiator.addr,_swaps[s
wapId].target) (Tradescrow.sol#189)
                - returndata = address(token).functionCall(data,SafeERC20: low-level
call failed) (@openzeppelin\contracts\token\ERC20\utils\SafeERC20.sol#93)
                - (success,returndata) = target.call{value: value}(data) (@openzeppelin
\contracts\utils\Address.sol#132)
                - IERC721(offer.nfts[i].addr).safeTransferFrom(from,to,offer.nfts[i].id,)
(Tradescrow.sol#322)
                - IERC1155(offer.nfts[i].addr).safeTransferFrom(from,to,offer.nfts[i].id,o
ffer.nfts[i].amount,) (Tradescrow.sol#324)
                -
IERC20(offer.coins[i_scope_0].addr).safeTransfer(to,offer.coins[i_scope_0].amount)
(Tradescrow.sol#329)
```

```
                - IERC20(offer.coins[i_scope_0].addr).safeTransferFrom(from,to,offer.coins
[i_scope_0].amount) (Tradescrow.sol#331)
        - safeMultipleTransfersFrom(address(this),_swaps[swapId].target.addr,_swaps[swap
Id].initiator) (Tradescrow.sol#192)
                - returndata = address(token).functionCall(data,SafeERC20: low-level
call failed) (@openzeppelin\contracts\token\ERC20\utils\SafeERC20.sol#93)
                - (success,returndata) = target.call{value: value}(data) (@openzeppelin
\contracts\utils\Address.sol#132)
                - IERC721(offer.nfts[i].addr).safeTransferFrom(from,to,offer.nfts[i].id,)
(Tradescrow.sol#322)
                - IERC1155(offer.nfts[i].addr).safeTransferFrom(from,to,offer.nfts[i].id,o
ffer.nfts[i].amount,) (Tradescrow.sol#324)
                -
IERC20(offer.coins[i_scope_0].addr).safeTransfer(to,offer.coins[i_scope_0].amount)
(Tradescrow.sol#329)
                - IERC20(offer.coins[i_scope_0].addr).safeTransferFrom(from,to,offer.coins
[i_scope_0].amount) (Tradescrow.sol#331)
        External calls sending eth:
        - safeMultipleTransfersFrom(address(this),_swaps[swapId].initiator.addr,_swaps[s
wapId].target) (Tradescrow.sol#189)
                - (success,returndata) = target.call{value: value}(data) (@openzeppelin
\contracts\utils\Address.sol#132)
        - safeMultipleTransfersFrom(address(this),_swaps[swapId].target.addr,_swaps[swap
Id].initiator) (Tradescrow.sol#192)
                - (success,returndata) = target.call{value: value}(data) (@openzeppelin
\contracts\utils\Address.sol#132)
        - transferNative(_swaps[swapId].initiator,_swaps[swapId].target)
(Tradescrow.sol#194)
                - to.addr.transfer(native) (Tradescrow.sol#341)
        - transferNative(_swaps[swapId].target,_swaps[swapId].initiator)
(Tradescrow.sol#195)
                - to.addr.transfer(native) (Tradescrow.sol#341)
        State variables written after the call(s):
        - transferNative(_swaps[swapId].target,_swaps[swapId].initiator)
(Tradescrow.sol#195)
                - _native -= from.native (Tradescrow.sol#338)
Reentrancy in Tradescrow.cancelSwap(uint256) (Tradescrow.sol#208-232):
        External calls:
        - safeMultipleTransfersFrom(address(this),_swaps[swapId].initiator.addr,_swaps[s
wapId].initiator) (Tradescrow.sol#217)
                - returndata = address(token).functionCall(data,SafeERC20: low-level
```

```
call failed) (@openzeppelin\contracts\token\ERC20\utils\SafeERC20.sol#93)
                - (success,returndata) = target.call{value: value}(data) (@openzeppelin
\contracts\utils\Address.sol#132)
                - IERC721(offer.nfts[i].addr).safeTransferFrom(from,to,offer.nfts[i].id,)
(Tradescrow.sol#322)
                - IERC1155(offer.nfts[i].addr).safeTransferFrom(from,to,offer.nfts[i].id,o
ffer.nfts[i].amount,) (Tradescrow.sol#324)
                -
IERC20(offer.coins[i_scope_0].addr).safeTransfer(to,offer.coins[i_scope_0].amount)
(Tradescrow.sol#329)
                - IERC20(offer.coins[i_scope_0].addr).safeTransferFrom(from,to,offer.coins
[i_scope_0].amount) (Tradescrow.sol#331)
        External calls sending eth:
        - safeMultipleTransfersFrom(address(this),_swaps[swapId].initiator.addr,_swaps[s
wapId].initiator) (Tradescrow.sol#217)
                - (success,returndata) = target.call{value: value}(data) (@openzeppelin
\contracts\utils\Address.sol#132)
        - transferNative(_swaps[swapId].initiator,_swaps[swapId].initiator)
(Tradescrow.sol#218)
                - to.addr.transfer(native) (Tradescrow.sol#341)
        State variables written after the call(s):
        - transferNative(_swaps[swapId].initiator,_swaps[swapId].initiator)
(Tradescrow.sol#218)
                - _native -= from.native (Tradescrow.sol#338)
Reentrancy in Tradescrow.cancelSwap(uint256) (Tradescrow.sol#208-232):
        External calls:
        - safeMultipleTransfersFrom(address(this),_swaps[swapId].target.addr,_swaps[swap
Id].target) (Tradescrow.sol#221)
                - returndata = address(token).functionCall(data,SafeERC20: low-level
call failed) (@openzeppelin\contracts\token\ERC20\utils\SafeERC20.sol#93)
                - (success,returndata) = target.call{value: value}(data) (@openzeppelin
\contracts\utils\Address.sol#132)
                - IERC721(offer.nfts[i].addr).safeTransferFrom(from,to,offer.nfts[i].id,)
(Tradescrow.sol#322)
                - IERC1155(offer.nfts[i].addr).safeTransferFrom(from,to,offer.nfts[i].id,o
ffer.nfts[i].amount,) (Tradescrow.sol#324)
                -
IERC20(offer.coins[i_scope_0].addr).safeTransfer(to,offer.coins[i_scope_0].amount)
(Tradescrow.sol#329)
                - IERC20(offer.coins[i_scope_0].addr).safeTransferFrom(from,to,offer.coins
[i_scope_0].amount) (Tradescrow.sol#331)
```

```
        External calls sending eth:
        - safeMultipleTransfersFrom(address(this),_swaps[swapId].target.addr,_swaps[swap
Id].target) (Tradescrow.sol#221)
                - (success,returndata) = target.call{value: value}(data) (@openzeppelin
\contracts\utils\Address.sol#132)
        - transferNative(_swaps[swapId].target,_swaps[swapId].target)
(Tradescrow.sol#222)
                - to.addr.transfer(native) (Tradescrow.sol#341)
        State variables written after the call(s):
        - transferNative(_swaps[swapId].target,_swaps[swapId].target)
(Tradescrow.sol#222)
                - _native -= from.native (Tradescrow.sol#338)
Reentrancy in Tradescrow.initiateSwap(uint256,Tradescrow.Offer)
(Tradescrow.sol#140-171):
        External calls:
        - safeMultipleTransfersFrom(address(msg.sender),address(this),offer)
(Tradescrow.sol#148-152)
                - returndata = address(token).functionCall(data,SafeERC20: low-level
call failed) (@openzeppelin\contracts\token\ERC20\utils\SafeERC20.sol#93)
                - (success,returndata) = target.call{value: value}(data) (@openzeppelin
\contracts\utils\Address.sol#132)
                - IERC721(offer.nfts[i].addr).safeTransferFrom(from,to,offer.nfts[i].id,)
(Tradescrow.sol#322)
                - IERC1155(offer.nfts[i].addr).safeTransferFrom(from,to,offer.nfts[i].id,o
ffer.nfts[i].amount,) (Tradescrow.sol#324)
                -
IERC20(offer.coins[i_scope_0].addr).safeTransfer(to,offer.coins[i_scope_0].amount)
(Tradescrow.sol#329)
                - IERC20(offer.coins[i_scope_0].addr).safeTransferFrom(from,to,offer.coins
[i_scope_0].amount) (Tradescrow.sol#331)
        External calls sending eth:
        - safeMultipleTransfersFrom(address(msg.sender),address(this),offer)
(Tradescrow.sol#148-152)
                - (success,returndata) = target.call{value: value}(data) (@openzeppelin
\contracts\utils\Address.sol#132)
        State variables written after the call(s):
        - _native += _swaps[swapId].target.native (Tradescrow.sol#162)
Reentrancy in Tradescrow.proposeSwap(address,Tradescrow.Offer) (Tradescrow.sol#100-126):
        External calls:
        - safeMultipleTransfersFrom(address(msg.sender),address(this),offer)
(Tradescrow.sol#105)
```

```
                - returndata = address(token).functionCall(data,SafeERC20: low-level
call failed) (@openzeppelin\contracts\token\ERC20\utils\SafeERC20.sol#93)
                - (success,returndata) = target.call{value: value}(data) (@openzeppelin
\contracts\utils\Address.sol#132)
                - IERC721(offer.nfts[i].addr).safeTransferFrom(from,to,offer.nfts[i].id,)
(Tradescrow.sol#322)
                - IERC1155(offer.nfts[i].addr).safeTransferFrom(from,to,offer.nfts[i].id,o
ffer.nfts[i].amount,) (Tradescrow.sol#324)
                -
IERC20(offer.coins[i_scope_0].addr).safeTransfer(to,offer.coins[i_scope_0].amount)
(Tradescrow.sol#329)
                - IERC20(offer.coins[i_scope_0].addr).safeTransferFrom(from,to,offer.coins
[i_scope_0].amount) (Tradescrow.sol#331)
        External calls sending eth:
        - safeMultipleTransfersFrom(address(msg.sender),address(this),offer)
(Tradescrow.sol#105)
                - (success,returndata) = target.call{value: value}(data) (@openzeppelin
\contracts\utils\Address.sol#132)
        State variables written after the call(s):
        - _native += swap.initiator.native (Tradescrow.sol#119)
        - swap.open = TRUEINT (Tradescrow.sol#109)
        - swap.initiator.addr = address(msg.sender) (Tradescrow.sol#110)
        - swap.initiator.nfts.push(offer.nfts[i]) (Tradescrow.sol#112)
        - swap.initiator.coins.push(offer.coins[i_scope_0]) (Tradescrow.sol#115)
        - swap.initiator.native = msg.value - fee (Tradescrow.sol#118)
        - swap.target.addr = target (Tradescrow.sol#121)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-2

Reentrancy in Tradescrow.acceptSwap(uint256) (Tradescrow.sol#182-200):
        External calls:
        - safeMultipleTransfersFrom(address(this),_swaps[swapId].initiator.addr,_swaps[s
wapId].target) (Tradescrow.sol#189)
                - returndata = address(token).functionCall(data,SafeERC20: low-level
call failed) (@openzeppelin\contracts\token\ERC20\utils\SafeERC20.sol#93)
                - (success,returndata) = target.call{value: value}(data) (@openzeppelin
\contracts\utils\Address.sol#132)
                - IERC721(offer.nfts[i].addr).safeTransferFrom(from,to,offer.nfts[i].id,)
(Tradescrow.sol#322)
                - IERC1155(offer.nfts[i].addr).safeTransferFrom(from,to,offer.nfts[i].id,o
ffer.nfts[i].amount,) (Tradescrow.sol#324)
```

```
                -
IERC20(offer.coins[i_scope_0].addr).safeTransfer(to,offer.coins[i_scope_0].amount)
(Tradescrow.sol#329)
                - IERC20(offer.coins[i_scope_0].addr).safeTransferFrom(from,to,offer.coins
[i_scope_0].amount) (Tradescrow.sol#331)
        - safeMultipleTransfersFrom(address(this),_swaps[swapId].target.addr,_swaps[swap
Id].initiator) (Tradescrow.sol#192)
                - returndata = address(token).functionCall(data,SafeERC20: low-level
call failed) (@openzeppelin\contracts\token\ERC20\utils\SafeERC20.sol#93)
                - (success,returndata) = target.call{value: value}(data) (@openzeppelin
\contracts\utils\Address.sol#132)
                - IERC721(offer.nfts[i].addr).safeTransferFrom(from,to,offer.nfts[i].id,)
(Tradescrow.sol#322)
                - IERC1155(offer.nfts[i].addr).safeTransferFrom(from,to,offer.nfts[i].id,o
ffer.nfts[i].amount,) (Tradescrow.sol#324)
                -
IERC20(offer.coins[i_scope_0].addr).safeTransfer(to,offer.coins[i_scope_0].amount)
(Tradescrow.sol#329)
                - IERC20(offer.coins[i_scope_0].addr).safeTransferFrom(from,to,offer.coins
[i_scope_0].amount) (Tradescrow.sol#331)
        External calls sending eth:
        - safeMultipleTransfersFrom(address(this),_swaps[swapId].initiator.addr,_swaps[s
wapId].target) (Tradescrow.sol#189)
                - (success,returndata) = target.call{value: value}(data) (@openzeppelin
\contracts\utils\Address.sol#132)
        - safeMultipleTransfersFrom(address(this),_swaps[swapId].target.addr,_swaps[swap
Id].initiator) (Tradescrow.sol#192)
                - (success,returndata) = target.call{value: value}(data) (@openzeppelin
\contracts\utils\Address.sol#132)
        - transferNative(_swaps[swapId].initiator,_swaps[swapId].target)
(Tradescrow.sol#194)
                - to.addr.transfer(native) (Tradescrow.sol#341)
        - transferNative(_swaps[swapId].target,_swaps[swapId].initiator)
(Tradescrow.sol#195)
                - to.addr.transfer(native) (Tradescrow.sol#341)
        Event emitted after the call(s):
        - SwapExecuted(_swaps[swapId].initiator.addr,_swaps[swapId].target.addr,swapId)
(Tradescrow.sol#197)
Reentrancy in Tradescrow.cancelSwap(uint256) (Tradescrow.sol#208-232):
        External calls:
        - safeMultipleTransfersFrom(address(this),_swaps[swapId].initiator.addr,_swaps[s
```

```
wapId].initiator) (Tradescrow.sol#217)
                - returndata = address(token).functionCall(data,SafeERC20: low-level
call failed) (@openzeppelin\contracts\token\ERC20\utils\SafeERC20.sol#93)
                - (success,returndata) = target.call{value: value}(data) (@openzeppelin
\contracts\utils\Address.sol#132)
                - IERC721(offer.nfts[i].addr).safeTransferFrom(from,to,offer.nfts[i].id,)
(Tradescrow.sol#322)
                - IERC1155(offer.nfts[i].addr).safeTransferFrom(from,to,offer.nfts[i].id,o
ffer.nfts[i].amount,) (Tradescrow.sol#324)
                -
IERC20(offer.coins[i_scope_0].addr).safeTransfer(to,offer.coins[i_scope_0].amount)
(Tradescrow.sol#329)
                - IERC20(offer.coins[i_scope_0].addr).safeTransferFrom(from,to,offer.coins
[i_scope_0].amount) (Tradescrow.sol#331)
        - safeMultipleTransfersFrom(address(this),_swaps[swapId].target.addr,_swaps[swap
Id].target) (Tradescrow.sol#221)
                - returndata = address(token).functionCall(data,SafeERC20: low-level
call failed) (@openzeppelin\contracts\token\ERC20\utils\SafeERC20.sol#93)
                - (success,returndata) = target.call{value: value}(data) (@openzeppelin
\contracts\utils\Address.sol#132)
                - IERC721(offer.nfts[i].addr).safeTransferFrom(from,to,offer.nfts[i].id,)
(Tradescrow.sol#322)
                - IERC1155(offer.nfts[i].addr).safeTransferFrom(from,to,offer.nfts[i].id,o
ffer.nfts[i].amount,) (Tradescrow.sol#324)
                -
IERC20(offer.coins[i_scope_0].addr).safeTransfer(to,offer.coins[i_scope_0].amount)
(Tradescrow.sol#329)
                - IERC20(offer.coins[i_scope_0].addr).safeTransferFrom(from,to,offer.coins
[i_scope_0].amount) (Tradescrow.sol#331)
        External calls sending eth:
        - safeMultipleTransfersFrom(address(this),_swaps[swapId].initiator.addr,_swaps[s
wapId].initiator) (Tradescrow.sol#217)
                - (success,returndata) = target.call{value: value}(data) (@openzeppelin
\contracts\utils\Address.sol#132)
        - transferNative(_swaps[swapId].initiator,_swaps[swapId].initiator)
(Tradescrow.sol#218)
                - to.addr.transfer(native) (Tradescrow.sol#341)
        - safeMultipleTransfersFrom(address(this),_swaps[swapId].target.addr,_swaps[swap
Id].target) (Tradescrow.sol#221)
                - (success,returndata) = target.call{value: value}(data) (@openzeppelin
\contracts\utils\Address.sol#132)
```

```
        - transferNative(_swaps[swapId].target,_swaps[swapId].target)
(Tradescrow.sol#222)
                - to.addr.transfer(native) (Tradescrow.sol#341)
        Event emitted after the call(s):
        - SwapCancelled(msg.sender,swapId) (Tradescrow.sol#226)
        - SwapClosed(swapId) (Tradescrow.sol#229)
Reentrancy in Tradescrow.initiateSwap(uint256,Tradescrow.Offer)
(Tradescrow.sol#140-171):
        External calls:
        - safeMultipleTransfersFrom(address(msg.sender),address(this),offer)
(Tradescrow.sol#148-152)
                - returndata = address(token).functionCall(data,SafeERC20: low-level
call failed) (@openzeppelin\contracts\token\ERC20\utils\SafeERC20.sol#93)
                - (success,returndata) = target.call{value: value}(data) (@openzeppelin
\contracts\utils\Address.sol#132)
                - IERC721(offer.nfts[i].addr).safeTransferFrom(from,to,offer.nfts[i].id,)
(Tradescrow.sol#322)
                - IERC1155(offer.nfts[i].addr).safeTransferFrom(from,to,offer.nfts[i].id,o
ffer.nfts[i].amount,) (Tradescrow.sol#324)
                -
IERC20(offer.coins[i_scope_0].addr).safeTransfer(to,offer.coins[i_scope_0].amount)
(Tradescrow.sol#329)
                - IERC20(offer.coins[i_scope_0].addr).safeTransferFrom(from,to,offer.coins
[i_scope_0].amount) (Tradescrow.sol#331)
        External calls sending eth:
        - safeMultipleTransfersFrom(address(msg.sender),address(this),offer)
(Tradescrow.sol#148-152)
                - (success,returndata) = target.call{value: value}(data) (@openzeppelin
\contracts\utils\Address.sol#132)
        Event emitted after the call(s):
        - SwapInitiated(msg.sender,_swaps[swapId].initiator.addr,swapId,offer)
(Tradescrow.sol#165-170)
Reentrancy in Tradescrow.proposeSwap(address,Tradescrow.Offer) (Tradescrow.sol#100-126):
        External calls:
        - safeMultipleTransfersFrom(address(msg.sender),address(this),offer)
(Tradescrow.sol#105)
                - returndata = address(token).functionCall(data,SafeERC20: low-level
call failed) (@openzeppelin\contracts\token\ERC20\utils\SafeERC20.sol#93)
                - (success,returndata) = target.call{value: value}(data) (@openzeppelin
\contracts\utils\Address.sol#132)
                - IERC721(offer.nfts[i].addr).safeTransferFrom(from,to,offer.nfts[i].id,)
(Tradescrow.sol#322)
```

```
                - IERC1155(offer.nfts[i].addr).safeTransferFrom(from,to,offer.nfts[i].id,o
ffer.nfts[i].amount,) (Tradescrow.sol#324)
                -
IERC20(offer.coins[i_scope_0].addr).safeTransfer(to,offer.coins[i_scope_0].amount)
(Tradescrow.sol#329)
                - IERC20(offer.coins[i_scope_0].addr).safeTransferFrom(from,to,offer.coins
[i_scope_0].amount) (Tradescrow.sol#331)
        External calls sending eth:
        - safeMultipleTransfersFrom(address(msg.sender),address(this),offer)
(Tradescrow.sol#105)
                - (success,returndata) = target.call{value: value}(data) (@openzeppelin
\contracts\utils\Address.sol#132)
        Event emitted after the call(s):
        - SwapProposed(msg.sender,target,_swapsCounter.current(),offer)
(Tradescrow.sol#123)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-3

Address.isContract(address) (@openzeppelin\contracts\utils\Address.sol#27-37) uses
assembly
        - INLINE ASM (@openzeppelin\contracts\utils\Address.sol#33-35)
Address.verifyCallResult(bool,bytes,string) (@openzeppelin\contracts\utils
\Address.sol#196-216) uses assembly
        - INLINE ASM (@openzeppelin\contracts\utils\Address.sol#208-211)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['^0.8.0', '^0.8.9']
        - ^0.8.0 (@openzeppelin\contracts\access\Ownable.sol#4)
        - ^0.8.0 (@openzeppelin\contracts\security\Pausable.sol#4)
        - ^0.8.0 (@openzeppelin\contracts\security\ReentrancyGuard.sol#4)
        - ^0.8.0 (@openzeppelin\contracts\token\ERC1155\IERC1155.sol#4)
        - ^0.8.0 (@openzeppelin\contracts\token\ERC1155\IERC1155Receiver.sol#4)
        - ^0.8.0 (@openzeppelin\contracts\token\ERC1155\utils\ERC1155Holder.sol#4)
        - ^0.8.0 (@openzeppelin\contracts\token\ERC1155\utils\ERC1155Receiver.sol#4)
        - ^0.8.0 (@openzeppelin\contracts\token\ERC20\IERC20.sol#4)
        - ^0.8.0 (@openzeppelin\contracts\token\ERC20\utils\SafeERC20.sol#4)
        - ^0.8.0 (@openzeppelin\contracts\token\ERC721\IERC721.sol#4)
        - ^0.8.0 (@openzeppelin\contracts\token\ERC721\IERC721Receiver.sol#4)
        - ^0.8.0 (@openzeppelin\contracts\token\ERC721\utils\ERC721Holder.sol#4)
        - ^0.8.0 (@openzeppelin\contracts\utils\Address.sol#4)
```

```
        - ^0.8.0 (@openzeppelin\contracts\utils\Context.sol#4)
        - ^0.8.0 (@openzeppelin\contracts\utils\Counters.sol#4)
        - ^0.8.0 (@openzeppelin\contracts\utils\introspection\ERC165.sol#4)
        - ^0.8.0 (@openzeppelin\contracts\utils\introspection\IERC165.sol#4)
        - ^0.8.9 (Tradescrow.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-
pragma-directives-are-used

Address.functionCall(address,bytes) (@openzeppelin\contracts\utils\Address.sol#80-82)
is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (@openzeppelin\contracts\utils
\Address.sol#109-115) is never used and should be removed
Address.functionDelegateCall(address,bytes) (@openzeppelin\contracts\utils
\Address.sol#169-171) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (@openzeppelin\contracts\utils
\Address.sol#179-188) is never used and should be removed
Address.functionStaticCall(address,bytes) (@openzeppelin\contracts\utils
\Address.sol#142-144) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (@openzeppelin\contracts\utils
\Address.sol#152-161) is never used and should be removed
Address.sendValue(address,uint256) (@openzeppelin\contracts\utils\Address.sol#55-60) is
never used and should be removed
Context._msgData() (@openzeppelin\contracts\utils\Context.sol#21-23) is never used and
should be removed
Counters.decrement(Counters.Counter) (@openzeppelin\contracts\utils\Counters.sol#32-38)
is never used and should be removed
Counters.reset(Counters.Counter) (@openzeppelin\contracts\utils\Counters.sol#40-42) is
never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (@openzeppelin\contracts\token\ERC20\utils
\SafeERC20.sol#45-58) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (@openzeppelin\contracts\token
\ERC20\utils\SafeERC20.sol#69-80) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (@openzeppelin\contracts\token
\ERC20\utils\SafeERC20.sol#60-67) is never used and should be removed
Tradescrow.requireEmpty(Tradescrow.Offer) (Tradescrow.sol#285-289) is never used and
should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (@openzeppelin\contracts\access\Ownable.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin\contracts\security\Pausable.sol#4) allows old
versions
```

```
Pragma version^0.8.0 (@openzeppelin\contracts\security\ReentrancyGuard.sol#4) allows
old versions
Pragma version^0.8.0 (@openzeppelin\contracts\token\ERC1155\IERC1155.sol#4) allows old
versions
Pragma version^0.8.0 (@openzeppelin\contracts\token\ERC1155\IERC1155Receiver.sol#4)
allows old versions
Pragma version^0.8.0 (@openzeppelin\contracts\token\ERC1155\utils\ERC1155Holder.sol#4)
allows old versions
Pragma version^0.8.0 (@openzeppelin\contracts\token\ERC1155\utils
\ERC1155Receiver.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin\contracts\token\ERC20\IERC20.sol#4) allows old
versions
Pragma version^0.8.0 (@openzeppelin\contracts\token\ERC20\utils\SafeERC20.sol#4) allows
old versions
Pragma version^0.8.0 (@openzeppelin\contracts\token\ERC721\IERC721.sol#4) allows old
versions
Pragma version^0.8.0 (@openzeppelin\contracts\token\ERC721\IERC721Receiver.sol#4)
allows old versions
Pragma version^0.8.0 (@openzeppelin\contracts\token\ERC721\utils\ERC721Holder.sol#4)
allows old versions
Pragma version^0.8.0 (@openzeppelin\contracts\utils\Address.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin\contracts\utils\Context.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin\contracts\utils\Counters.sol#4) allows old versions
Pragma version^0.8.0 (@openzeppelin\contracts\utils\introspection\ERC165.sol#4) allows
old versions
Pragma version^0.8.0 (@openzeppelin\contracts\utils\introspection\IERC165.sol#4) allows
old versions
Pragma version^0.8.9 (Tradescrow.sol#3) necessitates a version too recent to be
trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.11 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-
versions-of-solidity

Low level call in Address.sendValue(address,uint256) (@openzeppelin\contracts\utils
\Address.sol#55-60):
        - (success) = recipient.call{value: amount}() (@openzeppelin\contracts\utils
\Address.sol#58)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string)
(@openzeppelin\contracts\utils\Address.sol#123-134):
        - (success,returndata) = target.call{value: value}(data) (@openzeppelin
\contracts\utils\Address.sol#132)
```

```
Low level call in Address.functionStaticCall(address,bytes,string) (@openzeppelin
\contracts\utils\Address.sol#152-161):
        - (success,returndata) = target.staticcall(data) (@openzeppelin\contracts\utils
\Address.sol#159)
Low level call in Address.functionDelegateCall(address,bytes,string) (@openzeppelin
\contracts\utils\Address.sol#179-188):
        - (success,returndata) = target.delegatecall(data) (@openzeppelin\contracts
\utils\Address.sol#186)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-
calls

Reentrancy in Tradescrow.acceptSwap(uint256) (Tradescrow.sol#182-200):
        External calls:
        - transferNative(_swaps[swapId].initiator,_swaps[swapId].target)
(Tradescrow.sol#194)
                - to.addr.transfer(native) (Tradescrow.sol#341)
        - transferNative(_swaps[swapId].target,_swaps[swapId].initiator)
(Tradescrow.sol#195)
                - to.addr.transfer(native) (Tradescrow.sol#341)
        External calls sending eth:
        - safeMultipleTransfersFrom(address(this),_swaps[swapId].initiator.addr,_swaps[s
wapId].target) (Tradescrow.sol#189)
                - (success,returndata) = target.call{value: value}(data) (@openzeppelin
\contracts\utils\Address.sol#132)
        - safeMultipleTransfersFrom(address(this),_swaps[swapId].target.addr,_swaps[swap
Id].initiator) (Tradescrow.sol#192)
                - (success,returndata) = target.call{value: value}(data) (@openzeppelin
\contracts\utils\Address.sol#132)
        - transferNative(_swaps[swapId].initiator,_swaps[swapId].target)
(Tradescrow.sol#194)
                - to.addr.transfer(native) (Tradescrow.sol#341)
        - transferNative(_swaps[swapId].target,_swaps[swapId].initiator)
(Tradescrow.sol#195)
                - to.addr.transfer(native) (Tradescrow.sol#341)
        State variables written after the call(s):
        - transferNative(_swaps[swapId].target,_swaps[swapId].initiator)
(Tradescrow.sol#195)
                - _native -= from.native (Tradescrow.sol#338)
        - delete _swaps[swapId] (Tradescrow.sol#199)
        Event emitted after the call(s):
        - SwapExecuted(_swaps[swapId].initiator.addr,_swaps[swapId].target.addr,swapId)
(Tradescrow.sol#197)
```

```
Reentrancy in Tradescrow.cancelSwap(uint256) (Tradescrow.sol#208-232):
        External calls:
        - transferNative(_swaps[swapId].initiator,_swaps[swapId].initiator)
(Tradescrow.sol#218)
                - to.addr.transfer(native) (Tradescrow.sol#341)
        - transferNative(_swaps[swapId].target,_swaps[swapId].target)
(Tradescrow.sol#222)
                - to.addr.transfer(native) (Tradescrow.sol#341)
        External calls sending eth:
        - safeMultipleTransfersFrom(address(this),_swaps[swapId].initiator.addr,_swaps[s
wapId].initiator) (Tradescrow.sol#217)
                - (success,returndata) = target.call{value: value}(data) (@openzeppelin
\contracts\utils\Address.sol#132)
        - transferNative(_swaps[swapId].initiator,_swaps[swapId].initiator)
(Tradescrow.sol#218)
                - to.addr.transfer(native) (Tradescrow.sol#341)
        - safeMultipleTransfersFrom(address(this),_swaps[swapId].target.addr,_swaps[swap
Id].target) (Tradescrow.sol#221)
                - (success,returndata) = target.call{value: value}(data) (@openzeppelin
\contracts\utils\Address.sol#132)
        - transferNative(_swaps[swapId].target,_swaps[swapId].target)
(Tradescrow.sol#222)
                - to.addr.transfer(native) (Tradescrow.sol#341)
        State variables written after the call(s):
        - delete _swaps[swapId] (Tradescrow.sol#230)
        Event emitted after the call(s):
        - SwapCancelled(msg.sender,swapId) (Tradescrow.sol#226)
        - SwapClosed(swapId) (Tradescrow.sol#229)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-4

renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (@openzeppelin\contracts\access
\Ownable.sol#54-56)
onERC1155Received(address,address,uint256,uint256,bytes) should be declared external:
        - ERC1155Holder.onERC1155Received(address,address,uint256,uint256,bytes)
(@openzeppelin\contracts\token\ERC1155\utils\ERC1155Holder.sol#12-20)
onERC1155BatchReceived(address,address,uint256[],uint256[],bytes) should be declared
external:
        -
ERC1155Holder.onERC1155BatchReceived(address,address,uint256[],uint256[],bytes)
```

```
(@openzeppelin\contracts\token\ERC1155\utils\ERC1155Holder.sol#22-30)
onERC721Received(address,address,uint256,bytes) should be declared external:
        - ERC721Holder.onERC721Received(address,address,uint256,bytes) (@openzeppelin
\contracts\token\ERC721\utils\ERC721Holder.sol#20-27)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-
function-that-could-be-declared-external
Tradescrow.sol analyzed (18 contracts with 77 detectors), 61 result(s) found
```

0x Guard