0x Guard

Smart contracts security assessment

Final report

Tariff: Top

Moe Raven

November 2023

0xguard.com

hello@0xguard.com

Contents

Introduction

Audited contract is distributed vaults for CargoX (CXO) token. Each token storing contract holds not more than 250000 CXO tokens at the moment of deposit. Any token excesses are subjected to fee up to 100% defined by the project owner at the moment of calling the `harvest` function.

Audited contract is deployed to the 0x215bdd355C92F3a5c05A5888804857Bac042CB66 address in the Polygon Network.

Name	Moe Raven
Audit date	2023-11-10 - 2023-11-14
Language	Solidity
Platform	Polygon Network

Contracts checked

Name	Address
HoldingCell	0xF15f373FAa10EDf99846254e5d082E7BAD7E3785
DynamicRelayerVault	0x215bdd355C92F3a5c05A5888804857Bac042CB66

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 analyze 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	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Outdated Compiler Version	passed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

No issues were found

Medium severity issues

No issues were found

Low severity issues

No issues were found

Conclusion

Moe Raven HoldingCell, DynamicRelayerVault contracts were audited. No severity issues were found.

Disclaimer

Slither output

```
INFO:Detectors:
Reentrancy in DynamicRelayerVault._deposit(uint256) (contracts/
DynamicRelayerVault.sol#1137-1152):
        External calls:
        - depositToken.safeTransferFrom(msg.sender,address(this),_amt) (contracts/
DynamicRelayerVault.sol#1140)
        - _spread(_amt) (contracts/DynamicRelayerVault.sol#1142)
                - returndata = address(token).functionCall(data,SafeERC20: low-level
call failed) (contracts/DynamicRelayerVault.sol#668)
                - depositToken.safeTransfer(address(currentCell),_amt) (contracts/
DynamicRelayerVault.sol#1165)
                - (success,returndata) = target.call{value: value}(data) (contracts/
DynamicRelayerVault.sol#211)
                - currentCell.deposit(_amt) (contracts/DynamicRelayerVault.sol#1166)
                - depositToken.safeTransfer(address(currentCell),canDeposit) (contracts/
DynamicRelayerVault.sol#1169)
                - currentCell.deposit(canDeposit) (contracts/
DynamicRelayerVault.sol#1170)
                - depositToken.safeTransfer(address(currentCell),rest) (contracts/
DynamicRelayerVault.sol#1181)
                - currentCell.deposit(rest) (contracts/DynamicRelayerVault.sol#1182)
        External calls sending eth:
        - _spread(_amt) (contracts/DynamicRelayerVault.sol#1142)
                - (success,returndata) = target.call{value: value}(data) (contracts/
DynamicRelayerVault.sol#211)
        State variables written after the call(s):
        - balance += _amt (contracts/DynamicRelayerVault.sol#1143)
        DynamicRelayerVault.balance (contracts/DynamicRelayerVault.sol#1114) can be
used in cross function reentrancies:
        - DynamicRelayerVault.balance (contracts/DynamicRelayerVault.sol#1114)
        - DynamicRelayerVault.getPricePerFullShare() (contracts/
DynamicRelayerVault.sol#1313-1315)
        - DynamicRelayerVault.getUserDeposits(address) (contracts/
DynamicRelayerVault.sol#1327-1330)
        - DynamicRelayerVault.spreadExcess() (contracts/
DynamicRelayerVault.sol#1271-1281)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities
```

```
INFO:Detectors:
DynamicRelayerVault._deposit(uint256) (contracts/DynamicRelayerVault.sol#1137-1152)
uses a dangerous strict equality:
        - totalSupply() == 0 (contracts/DynamicRelayerVault.sol#1146)
DynamicRelayerVault.getPricePerFullShare() (contracts/
DynamicRelayerVault.sol#1313-1315) uses a dangerous strict equality:
        - totalSupply() == 0 (contracts/DynamicRelayerVault.sol#1314)
DynamicRelayerVault.isCellEmpty(uint256) (contracts/DynamicRelayerVault.sol#1323-1325)
uses a dangerous strict equality:
        - depositToken.balanceOf(holdingCells[cell]) == 0 (contracts/
DynamicRelayerVault.sol#1324)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-
strict-equalities
INFO:Detectors:
Reentrancy in DynamicRelayerVault._retract(uint256) (contracts/
DynamicRelayerVault.sol#1202-1226):
        External calls:
        - currentCell.withdraw(totalDeposits) (contracts/DynamicRelayerVault.sol#1217)
        State variables written after the call(s):
        - cellPointer -- (contracts/DynamicRelayerVault.sol#1220)
        DynamicRelayerVault.cellPointer (contracts/DynamicRelayerVault.sol#1116) can be
used in cross function reentrancies:
        - DynamicRelayerVault._spread(uint256) (contracts/
DynamicRelayerVault.sol#1155-1184)
        - DynamicRelayerVault.cellPointer (contracts/DynamicRelayerVault.sol#1116)
Reentrancy in DynamicRelayerVault._spread(uint256) (contracts/
DynamicRelayerVault.sol#1155-1184):
        External calls:
        - depositToken.safeTransfer(address(currentCell),canDeposit) (contracts/
DynamicRelayerVault.sol#1169)
        - currentCell.deposit(canDeposit) (contracts/DynamicRelayerVault.sol#1170)
        State variables written after the call(s):
        - cellPointer ++ (contracts/DynamicRelayerVault.sol#1173)
        DynamicRelayerVault.cellPointer (contracts/DynamicRelayerVault.sol#1116) can be
used in cross function reentrancies:
        - DynamicRelayerVault._spread(uint256) (contracts/
DynamicRelayerVault.sol#1155-1184)
        - DynamicRelayerVault.cellPointer (contracts/DynamicRelayerVault.sol#1116)
        - holdingCells.push(address(new HoldingCell(address(this)))) (contracts/
DynamicRelayerVault.sol#1176)
        DynamicRelayerVault.holdingCells (contracts/DynamicRelayerVault.sol#1112) can
```

```
be used in cross function reentrancies:
        - DynamicRelayerVault._spread(uint256) (contracts/
DynamicRelayerVault.sol#1155-1184)
        - DynamicRelayerVault.constructor(address,address,address) (contracts/
DynamicRelayerVault.sol#1123-1131)
        - DynamicRelayerVault.harvest(uint256,uint256) (contracts/
DynamicRelayerVault.sol#1284-1287)
        - DynamicRelayerVault.holdingCells (contracts/DynamicRelayerVault.sol#1112)
        - DynamicRelayerVault.isCellDormant(uint256) (contracts/
DynamicRelayerVault.sol#1318-1320)
        - DynamicRelayerVault.isCellEmpty(uint256) (contracts/
DynamicRelayerVault.sol#1323-1325)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-1
INFO:Detectors:
Reentrancy in DynamicRelayerVault._deposit(uint256) (contracts/
DynamicRelayerVault.sol#1137-1152):
        External calls:
        - depositToken.safeTransferFrom(msg.sender,address(this),_amt) (contracts/
DynamicRelayerVault.sol#1140)
        - _spread(_amt) (contracts/DynamicRelayerVault.sol#1142)
                - returndata = address(token).functionCall(data,SafeERC20: low-level
call failed) (contracts/DynamicRelayerVault.sol#668)
                - depositToken.safeTransfer(address(currentCell),_amt) (contracts/
DynamicRelayerVault.sol#1165)
                - (success,returndata) = target.call{value: value}(data) (contracts/
DynamicRelayerVault.sol#211)
                - currentCell.deposit(_amt) (contracts/DynamicRelayerVault.sol#1166)
                - depositToken.safeTransfer(address(currentCell),canDeposit) (contracts/
DynamicRelayerVault.sol#1169)
                - currentCell.deposit(canDeposit) (contracts/
DynamicRelayerVault.sol#1170)
                - depositToken.safeTransfer(address(currentCell),rest) (contracts/
DynamicRelayerVault.sol#1181)
                - currentCell.deposit(rest) (contracts/DynamicRelayerVault.sol#1182)
        External calls sending eth:
        - _spread(_amt) (contracts/DynamicRelayerVault.sol#1142)
                - (success,returndata) = target.call{value: value}(data) (contracts/
DynamicRelayerVault.sol#211)
        State variables written after the call(s):
        - _mint(msg.sender,shares) (contracts/DynamicRelayerVault.sol#1151)
```

```
                - _balances[account] += amount (contracts/DynamicRelayerVault.sol#969)
        - _mint(msg.sender,shares) (contracts/DynamicRelayerVault.sol#1151)
                - _totalSupply += amount (contracts/DynamicRelayerVault.sol#968)
Reentrancy in DynamicRelayerVault.spreadExcess() (contracts/
DynamicRelayerVault.sol#1271-1281):
        External calls:
        - _spread(excessTokens) (contracts/DynamicRelayerVault.sol#1278)
                - returndata = address(token).functionCall(data,SafeERC20: low-level
call failed) (contracts/DynamicRelayerVault.sol#668)
                - depositToken.safeTransfer(address(currentCell),_amt) (contracts/
DynamicRelayerVault.sol#1165)
                - (success,returndata) = target.call{value: value}(data) (contracts/
DynamicRelayerVault.sol#211)
                - currentCell.deposit(_amt) (contracts/DynamicRelayerVault.sol#1166)
                - depositToken.safeTransfer(address(currentCell),canDeposit) (contracts/
DynamicRelayerVault.sol#1169)
                - currentCell.deposit(canDeposit) (contracts/
DynamicRelayerVault.sol#1170)
                - depositToken.safeTransfer(address(currentCell),rest) (contracts/
DynamicRelayerVault.sol#1181)
                - currentCell.deposit(rest) (contracts/DynamicRelayerVault.sol#1182)
        External calls sending eth:
        - _spread(excessTokens) (contracts/DynamicRelayerVault.sol#1278)
                - (success,returndata) = target.call{value: value}(data) (contracts/
DynamicRelayerVault.sol#211)
        State variables written after the call(s):
        - balance += excessTokens (contracts/DynamicRelayerVault.sol#1279)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-2
INFO:Detectors:
Reentrancy in DynamicRelayerVault._spread(uint256) (contracts/
DynamicRelayerVault.sol#1155-1184):
        External calls:
        - depositToken.safeTransfer(address(currentCell),canDeposit) (contracts/
DynamicRelayerVault.sol#1169)
        - currentCell.deposit(canDeposit) (contracts/DynamicRelayerVault.sol#1170)
        Event emitted after the call(s):
        - CreateRelayer(cellPointer) (contracts/DynamicRelayerVault.sol#1174)
Reentrancy in DynamicRelayerVault.spreadExcess() (contracts/
DynamicRelayerVault.sol#1271-1281):
        External calls:
```

```
        - _spread(excessTokens) (contracts/DynamicRelayerVault.sol#1278)
                - returndata = address(token).functionCall(data,SafeERC20: low-level
call failed) (contracts/DynamicRelayerVault.sol#668)
                - depositToken.safeTransfer(address(currentCell),_amt) (contracts/
DynamicRelayerVault.sol#1165)
                - (success,returndata) = target.call{value: value}(data) (contracts/
DynamicRelayerVault.sol#211)
                - currentCell.deposit(_amt) (contracts/DynamicRelayerVault.sol#1166)
                - depositToken.safeTransfer(address(currentCell),canDeposit) (contracts/
DynamicRelayerVault.sol#1169)
                - currentCell.deposit(canDeposit) (contracts/
DynamicRelayerVault.sol#1170)
                - depositToken.safeTransfer(address(currentCell),rest) (contracts/
DynamicRelayerVault.sol#1181)
                - currentCell.deposit(rest) (contracts/DynamicRelayerVault.sol#1182)
        External calls sending eth:
        - _spread(excessTokens) (contracts/DynamicRelayerVault.sol#1278)
                - (success,returndata) = target.call{value: value}(data) (contracts/
DynamicRelayerVault.sol#211)
        Event emitted after the call(s):
        - SpreadExcess(excessTokens) (contracts/DynamicRelayerVault.sol#1280)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-3
INFO:Detectors:
Address.verifyCallResult(bool,bytes,string) (contracts/DynamicRelayerVault.sol#275-295)
uses assembly
        - INLINE ASM (contracts/DynamicRelayerVault.sol#287-290)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Different versions of Solidity are used:
        - Version used: ['^0.8.0', '^0.8.1']
        - ^0.8.0 (contracts/DynamicRelayerVault.sol#12)
        - ^0.8.0 (contracts/DynamicRelayerVault.sol#303)
        - ^0.8.0 (contracts/DynamicRelayerVault.sol#366)
        - ^0.8.0 (contracts/DynamicRelayerVault.sol#393)
        - ^0.8.0 (contracts/DynamicRelayerVault.sol#478)
        - ^0.8.0 (contracts/DynamicRelayerVault.sol#563)
        - ^0.8.0 (contracts/DynamicRelayerVault.sol#681)
        - ^0.8.0 (contracts/DynamicRelayerVault.sol#711)
        - ^0.8.0 (contracts/DynamicRelayerVault.sol#1095)
        - ^0.8.1 (contracts/DynamicRelayerVault.sol#78)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-
pragma-directives-are-used
INFO:Detectors:
Address.functionCall(address,bytes) (contracts/DynamicRelayerVault.sol#159-161) is
never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (contracts/
DynamicRelayerVault.sol#188-194) is never used and should be removed
Address.functionDelegateCall(address,bytes) (contracts/DynamicRelayerVault.sol#248-250)
is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (contracts/
DynamicRelayerVault.sol#258-267) is never used and should be removed
Address.functionStaticCall(address,bytes) (contracts/DynamicRelayerVault.sol#221-223)
is never used and should be removed
Address.functionStaticCall(address,bytes,string) (contracts/
DynamicRelayerVault.sol#231-240) is never used and should be removed
Address.sendValue(address,uint256) (contracts/DynamicRelayerVault.sol#134-139) is never
used and should be removed
Context._msgData() (contracts/DynamicRelayerVault.sol#383-385) is never used and should
be removed
SafeERC20.safeApprove(IERC20,address,uint256) (contracts/
DynamicRelayerVault.sol#604-617) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (contracts/
DynamicRelayerVault.sol#628-639) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (contracts/
DynamicRelayerVault.sol#619-626) is never used and should be removed
SafeERC20.safePermit(IERC20Permit,address,address,uint256,uint256,uint8,bytes32,bytes32)
 (contracts/DynamicRelayerVault.sol#641-655) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (contracts/DynamicRelayerVault.sol#12) allows old versions
Pragma version^0.8.1 (contracts/DynamicRelayerVault.sol#78) allows old versions
Pragma version^0.8.0 (contracts/DynamicRelayerVault.sol#303) allows old versions
Pragma version^0.8.0 (contracts/DynamicRelayerVault.sol#366) allows old versions
Pragma version^0.8.0 (contracts/DynamicRelayerVault.sol#393) allows old versions
Pragma version^0.8.0 (contracts/DynamicRelayerVault.sol#478) allows old versions
Pragma version^0.8.0 (contracts/DynamicRelayerVault.sol#563) allows old versions
Pragma version^0.8.0 (contracts/DynamicRelayerVault.sol#681) allows old versions
Pragma version^0.8.0 (contracts/DynamicRelayerVault.sol#711) allows old versions
Pragma version^0.8.0 (contracts/DynamicRelayerVault.sol#1095) allows old versions
solc-0.8.21 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-
```

```
versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (contracts/
DynamicRelayerVault.sol#134-139):
        - (success) = recipient.call{value: amount}() (contracts/
DynamicRelayerVault.sol#137)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string)
(contracts/DynamicRelayerVault.sol#202-213):
        - (success,returndata) = target.call{value: value}(data) (contracts/
DynamicRelayerVault.sol#211)
Low level call in Address.functionStaticCall(address,bytes,string) (contracts/
DynamicRelayerVault.sol#231-240):
        - (success,returndata) = target.staticcall(data) (contracts/
DynamicRelayerVault.sol#238)
Low level call in Address.functionDelegateCall(address,bytes,string) (contracts/
DynamicRelayerVault.sol#258-267):
        - (success,returndata) = target.delegatecall(data) (contracts/
DynamicRelayerVault.sol#265)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-
calls
INFO:Detectors:
Function IERC20Permit.DOMAIN_SEPARATOR() (contracts/DynamicRelayerVault.sol#358) is not
in mixedCase
Parameter DynamicRelayerVault.deposit(uint256)._amt (contracts/
DynamicRelayerVault.sol#1245) is not in mixedCase
Parameter DynamicRelayerVault.withdraw(uint256)._amt (contracts/
DynamicRelayerVault.sol#1262) is not in mixedCase
Parameter DynamicRelayerVault.harvest(uint256,uint256)._cellNumber (contracts/
DynamicRelayerVault.sol#1284) is not in mixedCase
Parameter DynamicRelayerVault.harvest(uint256,uint256)._fee (contracts/
DynamicRelayerVault.sol#1284) is not in mixedCase
Parameter DynamicRelayerVault.changeHarvestor(address)._harvestor (contracts/
DynamicRelayerVault.sol#1293) is not in mixedCase
Parameter DynamicRelayerVault.changeFeeAddress(address)._feeAddress (contracts/
DynamicRelayerVault.sol#1304) is not in mixedCase
Parameter DynamicRelayerVault.getUserDeposits(address)._user (contracts/
DynamicRelayerVault.sol#1327) is not in mixedCase
Parameter HoldingCell.deposit(uint256)._amt (contracts/DynamicRelayerVault.sol#1347) is
not in mixedCase
Parameter HoldingCell.withdraw(uint256)._amt (contracts/DynamicRelayerVault.sol#1352)
is not in mixedCase
```

```
Parameter HoldingCell.harvest(uint256)._fee (contracts/DynamicRelayerVault.sol#1357) is
not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-
solidity-naming-conventions
INFO:Detectors:
DynamicRelayerVault.depositToken (contracts/DynamicRelayerVault.sol#1105) should be
immutable
HoldingCell.vault (contracts/DynamicRelayerVault.sol#1337) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-
variables-that-could-be-declared-immutable
INFO:Slither:. analyzed (11 contracts with 88 detectors), 52 result(s) found
```

0x Guard