0x Guard

# Smart contracts security assessment

Final report

Tariff: Standard

**Catsluck**

December 2021

0xguard.com

hello@0xguard.com

## Contents

## Introduction

The report has been prepared for the Catsluck team. The code is published to GitHub and was audited after commit 0921847.

Users interacting with contracts must check that the contract is the same as was audited.

The gambling project with two betting strategies.

| Name | Catsluck |
|---|---|
| Audit date | 2021-12-23 - 2021-12-24 |
| Language | Solidity |
| Platform | SmartBCH |

## Contracts checked

| Name | Address |
|---|---|
| catsluck | |
| catsluck4bch | |
| fun | |
| catsluck4fun | |
| catsluck4erc20 | |
| comment | |

## 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 | 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| Outdated Compiler Version | passed |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

**1. function buyback is susceptible to sandwich attacks (catsluck)**

The function buyback swaps tokens with 100% slippage which makes it susceptible to sandwich attacks.

**2. Predictable prize (catsluck)**

If the lottery contract's balance increased during deposit, a better will definitely receive more tokens than he deposited. This simple rule can be used by some gamblers for choosing appropriate withdrawal time.

This issue also refers to catsluck4erc20, catsluck4bch, catsluck4fun.

**3. lockUntil can be set to minimum value without any side affect (catsluck)**

lockUntil parameter in depositCATS() function determines the block number after which a gambler can withdraw his funds. It can be set to a value that is less than a current block. Also, it doesn't affect withdrawal value, so the variable is considered redundant.

This issue also refers to catsluck4erc20, catsluck4bch, catsluck4fun.

**4. Call usage (catsluck4bch)**

In withrawCATS() and getMyReward() functions prize blockchain currency is transferred with call() method that can return negative boolean instead of transaction reverting. Insufficient contract balance may lead to not full prize payment without proper unsuccess handling.In

This issue also refers to catsluck4erc20 and catsluck4fun.

**Recommendation:** Use transfer() method.

### 5. Open mint (fun)

The token has the open mint. This means, that anyone can mint it in any desired amount. The Catsluck team didn't share any documentation about the project, but during the audit contracts with 'fun' suffix in the name were considered as demo versions of the betting and token. Users must be informed about it.

As SafeMath is not used in the mint function so everyone can set any balance of his address.

**Update:** The Catsluck team responded that the fun contract is an in-game token for just fun or test(practice).

## Low severity issues

### 1. Overdue tickets are not deleted from the mapping (catsluck)

In case of overdue tickets, getMyReward() the function returns without ticket delete, missing gas refund.

This issue also refers to catsluck4erc20, catsluck4bch, catsluck4fun.

**Recommendation:** swap L:124 and L:125.

### 2. A player should claim his bet within ~22 minutes (catsluck)

After a gambler receives a ticket he has just 256 blocks to play his bet. Otherwise blockchainhash() the function will return zero hash on his block number and the bet won't be played.

This issue also refers to catsluck4erc20, catsluck4bch, catsluck4fun.

### 3. Unsafe token transfer (catsluck)

In withrawCATS() and getMyReward() functions prize tokens are transferred with unsecured ERC20 transfer() method that can return negative boolean instead of transaction reverting. Insufficient contract's token balance may lead to not full prize payment without proper unsuccess handling.

This issue also refers to catsluck4erc20 and catsluck4fun.

**Recommendation:** Although token transfers with insufficient balance are usually reverted by token's contract we advise using SafeERC20 transfer wrapper.

## Conclusion

Catsluck catsluck, catsluck4bch, fun, catsluck4fun, catsluck4erc20, comment contracts were audited. 5 medium, 3 low severity issues were found.

The fun-contract has an open mint function and everyone can mint an unlimited amount of tokens to his address or set any balance of his address. It should not be traded or taken seriously.

Users interacting with contracts must check that the contract has is the same as was audited.

## 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.

## Static code analysis result

catsluck.getMyReward() (contracts/catsluck.sol#117-138) uses a weak PRNG: "rand = rand % DIV (contracts/catsluck.sol#129)"

catsluck4bch.getMyReward() (contracts/catsluck4bch.sol#103-124) uses a weak PRNG: "rand = rand % DIV (contracts/catsluck4bch.sol#115)"

catsluck4erc20.getMyReward() (contracts/catsluck4erc20.sol#118-139) uses a weak PRNG: "rand = rand % DIV (contracts/catsluck4erc20.sol#130)"

catsluck4fun.getMyReward() (contracts/catsluck4fun.sol#97-118) uses a weak PRNG: "rand = rand % DIV (contracts/catsluck4fun.sol#109)"

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG

IBenSwapRouter is re-used:

- contracts/catsluck4erc20.sol#6-13
- contracts/catsluck.sol#6-13

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#name-reused

Reentrancy in catsluck4bch.buyLottery(uint256) (contracts/catsluck4bch.sol#75-101):

External calls:

- getMyReward() (contracts/catsluck4bch.sol#82)
    - msg.sender.call{value: reward}() (contracts/catsluck4bch.sol#121)

State variables written after the call(s):

- buyerTickets[msg.sender] = (remainedAmount << 96) | (block.number << 32) | multiplierX100

(contracts/catsluck4bch.sol#99)

Reentrancy in catsluck4bch.buyback() (contracts/catsluck4bch.sol#126-136):

External calls:

- amounts = IBenSwapRouterNative(routerAddress).swapExactETHForTokens{value: oldBuybackPoolFunds}(0,path,address(1),9000000000) (contracts/catsluck4bch.sol#132-133)

State variables written after the call(s):

- buybackPoolTokenCount = oldBuybackPoolFunds - amounts[0] (contracts/catsluck4bch.sol#134)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities

catsluck.safeTransferFrom(address,uint256) (contracts/catsluck.sol#38-43) ignores return value by IERC20(erc20Address).transferFrom(addr,address(this),amount) (contracts/catsluck.sol#40)

catsluck.withdrawCATS(uint256) (contracts/catsluck.sol#74-88) ignores return value by IERC20(erc20Address).transfer(msg.sender,deltaBalance) (contracts/catsluck.sol#86)

catsluck.getMyReward() (contracts/catsluck.sol#117-138) ignores return value by IERC20(erc20Address).transfer(msg.sender,reward) (contracts/catsluck.sol#134)

catsluck4erc20.safeTransferFrom(address,uint256) (contracts/catsluck4erc20.sol#38-43) ignores return value by IERC20(erc20Address).transferFrom(addr,address(this),amount) (contracts/catsluck4erc20.sol#40)

catsluck4erc20.withdraw(uint256) (contracts/catsluck4erc20.sol#75-89) ignores return value by IERC20(erc20Address).transfer(msg.sender,deltaBalance) (contracts/catsluck4erc20.sol#87)

catsluck4erc20.getMyReward() (contracts/catsluck4erc20.sol#118-139) ignores return value by IERC20(erc20Address).transfer(msg.sender,reward) (contracts/catsluck4erc20.sol#135) catsluck4fun.safeTransferFrom(address,uint256) (contracts/catsluck4fun.sol#20-25) ignores return value by IERC20(erc20Address).transferFrom(addr,address(this),amount) (contracts/catsluck4fun.sol#22)

catsluck4fun.withdraw(uint256) (contracts/catsluck4fun.sol#56-69) ignores return value by IERC20(erc20Address).transfer(msg.sender,deltaBalance) (contracts/catsluck4fun.sol#67)

catsluck4fun.getMyReward() (contracts/catsluck4fun.sol#97-118) ignores return value by IERC20(erc20Address).transfer(msg.sender,reward) (contracts/catsluck4fun.sol#114)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

catsluck.getMyReward() (contracts/catsluck.sol#117-138) uses a dangerous strict equality:

- amountAndHeightAndMul == 0 (contracts/catsluck.sol#119)

catsluck.getMyReward() (contracts/catsluck.sol#117-138) uses a dangerous strict equality:

- uint256(hash) == 0 (contracts/catsluck.sol#124)

catsluck4bch.deposit(uint256) (contracts/catsluck4bch.sol#31-53) uses a dangerous strict equality:

- totalShare == 0 (contracts/catsluck4bch.sol#42)

catsluck4bch.getMyReward() (contracts/catsluck4bch.sol#103-124) uses a dangerous strict equality:

- amountAndHeightAndMul == 0 (contracts/catsluck4bch.sol#105)

catsluck4bch.getMyReward() (contracts/catsluck4bch.sol#103-124) uses a dangerous strict equality:

- uint256(hash) == 0 (contracts/catsluck4bch.sol#110)

catsluck4erc20.getMyReward() (contracts/catsluck4erc20.sol#118-139) uses a dangerous strict equality:

- amountAndHeightAndMul == 0 (contracts/catsluck4erc20.sol#120)

catsluck4erc20.getMyReward() (contracts/catsluck4erc20.sol#118-139) uses a dangerous strict equality:

- uint256(hash) == 0 (contracts/catsluck4erc20.sol#125)

catsluck4fun.getMyReward() (contracts/catsluck4fun.sol#97-118) uses a dangerous strict equality:

- amountAndHeightAndMul == 0 (contracts/catsluck4fun.sol#99)

catsluck4fun.getMyReward() (contracts/catsluck4fun.sol#97-118) uses a dangerous strict equality:

- uint256(hash) == 0 (contracts/catsluck4fun.sol#104)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in catsluck.buyLottery(uint256,uint256) (contracts/catsluck.sol#90-115):

External calls:

- getMyReward() (contracts/catsluck.sol#96)

- IERC20(erc20Address).transfer(msg.sender,reward) (contracts/catsluck.sol#134)

- (amount,oldBalance) = safeTransferFrom(msg.sender,amount) (contracts/catsluck.sol#101)

- IERC20(erc20Address).transferFrom(addr,address(this),amount) (contracts/catsluck.sol#40)

State variables written after the call(s):

- buyerTickets[msg.sender] = (remainedAmount << 96) | (block.number << 32) | multiplierX100 (contracts/catsluck.sol#110)

Reentrancy in catsluck4erc20.buyLottery(uint256,uint256) (contracts/catsluck4erc20.sol#91-116):

External calls:

- getMyReward() (contracts/catsluck4erc20.sol#97)

  - IERC20(erc20Address).transfer(msg.sender,reward) (contracts/catsluck4erc20.sol#135)

- (amount,oldBalance) = safeTransferFrom(msg.sender,amount) (contracts/catsluck4erc20.sol#101)

  - IERC20(erc20Address).transferFrom(addr,address(this),amount) (contracts/catsluck4erc20.sol#40)

State variables written after the call(s):

- buyerTickets[msg.sender] = (remainedAmount << 96) | (block.number << 32) | multiplierX100 (contracts/catsluck4erc20.sol#114)

Reentrancy in catsluck4fun.buyLottery(uint256,uint256) (contracts/catsluck4fun.sol#71-95):

External calls:

- getMyReward() (contracts/catsluck4fun.sol#77)

  - IERC20(erc20Address).transfer(msg.sender,reward) (contracts/catsluck4fun.sol#114)

- (amount,oldBalance) = safeTransferFrom(msg.sender,amount) (contracts/catsluck4fun.sol#81)

  - IERC20(erc20Address).transferFrom(addr,address(this),amount) (contracts/catsluck4fun.sol#22)

State variables written after the call(s):

- buyerTickets[msg.sender] = (remainedAmount << 96) | (block.number << 32) | multiplierX100 (contracts/catsluck4fun.sol#93)

```
Reentrancy in catsluck.buyback() (contracts/catsluck.sol#140-149):
	External calls:
	- amounts = IBenSwapRouter(routerAddress).swapExactTokensForTokens(oldBuybackPoolFund
s,0,path,address(1),9000000000) (contracts/catsluck.sol#145-146)
	State variables written after the call(s):
	- buybackPoolTokenCount = oldBuybackPoolFunds - amounts[0] (contracts/catsluck.sol#147)
Reentrancy in catsluck4erc20.buyback() (contracts/catsluck4erc20.sol#141-152):
	External calls:
	- amounts = IBenSwapRouter(routerAddress).swapExactTokensForTokens(oldBuybackPoolFund
s,0,path,address(1),9000000000) (contracts/catsluck4erc20.sol#148-149)
	State variables written after the call(s):
	- buybackPoolTokenCount = oldBuybackPoolFunds - amounts[0] (contracts/
catsluck4erc20.sol#150)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

catsluck4bch.withdraw(uint256) (contracts/catsluck4bch.sol#60-73) ignores return value by
msg.sender.call{value: deltaBalance}() (contracts/catsluck4bch.sol#72)
catsluck4bch.getMyReward() (contracts/catsluck4bch.sol#103-124) ignores return value by
msg.sender.call{value: reward}() (contracts/catsluck4bch.sol#121)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-low-level-calls

catsluck.constructor() (contracts/catsluck.sol#34-36) ignores return value by
IERC20(erc20Address).approve(routerAddress,~ uint256(0)) (contracts/catsluck.sol#35)
```

catsluck4erc20.constructor(address) (contracts/catsluck4erc20.sol#33-36) ignores return value by IERC20(addr).approve(routerAddress,~ uint256(0)) (contracts/catsluck4erc20.sol#35)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

catsluck4erc20.constructor(address).addr (contracts/catsluck4erc20.sol#33) lacks a zero-check on :

- erc20Address = addr (contracts/catsluck4erc20.sol#34)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in catsluck.buyLottery(uint256,uint256) (contracts/catsluck.sol#90-115):

External calls:

- getMyReward() (contracts/catsluck.sol#96)

- IERC20(erc20Address).transfer(msg.sender,reward) (contracts/catsluck.sol#134)

- (amount,oldBalance) = safeTransferFrom(msg.sender,amount) (contracts/catsluck.sol#101)

- IERC20(erc20Address).transferFrom(addr,address(this),amount) (contracts/catsluck.sol#40)

State variables written after the call(s):

- _mint(msg.sender,amount * MUL) (contracts/catsluck.sol#113)

- _balances[account] += amount (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#258)

- _mint(msg.sender,amount * MUL) (contracts/catsluck.sol#113)

- _totalSupply += amount (node_modules/@openzeppelin/contracts/token/ERC20/

ERC20.sol#257)

- buybackPoolTokenCount += fee / 2 (contracts/catsluck.sol#105)

Reentrancy in catsluck4bch.buyLottery(uint256) (contracts/catsluck4bch.sol#75-101):

External calls:

- getMyReward() (contracts/catsluck4bch.sol#82)

- msg.sender.call{value: reward}() (contracts/catsluck4bch.sol#121)

State variables written after the call(s):

- buybackPoolTokenCount += amount / 200 (contracts/catsluck4bch.sol#89)

Reentrancy in catsluck4erc20.buyLottery(uint256,uint256) (contracts/catsluck4erc20.sol#91-116):

External calls:

- getMyReward() (contracts/catsluck4erc20.sol#97)

- IERC20(erc20Address).transfer(msg.sender,reward) (contracts/catsluck4erc20.sol#135)

- (amount,oldBalance) = safeTransferFrom(msg.sender,amount) (contracts/catsluck4erc20.sol#101)

- IERC20(erc20Address).transferFrom(addr,address(this),amount) (contracts/catsluck4erc20.sol#40)

State variables written after the call(s):

- buybackPoolTokenCount += amount / 200 (contracts/catsluck4erc20.sol#104)

Reentrancy in catsluck4erc20.deposit(uint256,uint256) (contracts/catsluck4erc20.sol#45-68):

External calls:

- (amount,oldBalance) = safeTransferFrom(msg.sender,amount) (contracts/catsluck4erc20.sol#47)

    - IERC20(erc20Address).transferFrom(addr,address(this),amount) (contracts/catsluck4erc20.sol#40)

State variables written after the call(s):

- sharesAndLockUntil[msg.sender] = (amount << 64) | lockUntil (contracts/catsluck4erc20.sol#59)

- sharesAndLockUntil[msg.sender] = ((deltaShare + oldShare) << 64) | lockUntil (contracts/catsluck4erc20.sol#65)

- totalShare = amount (contracts/catsluck4erc20.sol#60)

- totalShare += deltaShare (contracts/catsluck4erc20.sol#66)

Reentrancy in catsluck4fun.deposit(uint256,uint256) (contracts/catsluck4fun.sol#27-49):

External calls:

- (amount,oldBalance) = safeTransferFrom(msg.sender,amount) (contracts/catsluck4fun.sol#29)

    - IERC20(erc20Address).transferFrom(addr,address(this),amount) (contracts/catsluck4fun.sol#22)

State variables written after the call(s):

- sharesAndLockUntil[msg.sender] = (amount << 64) | lockUntil (contracts/catsluck4fun.sol#40)

- sharesAndLockUntil[msg.sender] = ((deltaShare + oldShare) << 64) | lockUntil (contracts/catsluck4fun.sol#46)

- totalShare = amount (contracts/catsluck4fun.sol#41)

- totalShare += deltaShare (contracts/catsluck4fun.sol#47)

Reentrancy in catsluck.depositCATS(uint256,uint256) (contracts/catsluck.sol#45-67):

External calls:

- (amount,oldBalance) = safeTransferFrom(msg.sender,amount) (contracts/catsluck.sol#47)

- IERC20(erc20Address).transferFrom(addr,address(this),amount) (contracts/catsluck.sol#40)

State variables written after the call(s):

- sharesAndLockUntil[msg.sender] = (amount << 64) | lockUntil (contracts/catsluck.sol#58)

- sharesAndLockUntil[msg.sender] = ((deltaShare + oldShare) << 64) | lockUntil (contracts/catsluck.sol#64)

- totalShare = amount (contracts/catsluck.sol#59)

- totalShare += deltaShare (contracts/catsluck.sol#65)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in catsluck.buyLottery(uint256,uint256) (contracts/catsluck.sol#90-115):

External calls:

- getMyReward() (contracts/catsluck.sol#96)

- IERC20(erc20Address).transfer(msg.sender,reward) (contracts/catsluck.sol#134)

- (amount,oldBalance) = safeTransferFrom(msg.sender,amount) (contracts/catsluck.sol#101)

```
        - IERC20(erc20Address).transferFrom(addr,address(this),amount) (contracts/catsluck.sol#40)
    Event emitted after the call(s):
    - Buy(msg.sender,(amount << (64 + 32)) | (multiplierX100 << 64) | block.timestamp) (contracts/catsluck.sol#111)
    - Transfer(address(0),account,amount) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#259)
        - _mint(msg.sender,amount * MUL) (contracts/catsluck.sol#113)
Reentrancy in catsluck4bch.buyLottery(uint256) (contracts/catsluck4bch.sol#75-101):
    External calls:
    - getMyReward() (contracts/catsluck4bch.sol#82)
        - msg.sender.call{value: reward}() (contracts/catsluck4bch.sol#121)
    Event emitted after the call(s):
    - Buy(msg.sender,(amount << (64 + 32)) | (multiplierX100 << 64) | block.timestamp) (contracts/catsluck4bch.sol#100)
Reentrancy in catsluck4erc20.buyLottery(uint256,uint256) (contracts/catsluck4erc20.sol#91-116):
    External calls:
    - getMyReward() (contracts/catsluck4erc20.sol#97)
        - IERC20(erc20Address).transfer(msg.sender,reward) (contracts/catsluck4erc20.sol#135)
    - (amount,oldBalance) = safeTransferFrom(msg.sender,amount) (contracts/
```

catsluck4erc20.sol#101)

- IERC20(erc20Address).transferFrom(addr,address(this),amount) (contracts/catsluck4erc20.sol#40)

Event emitted after the call(s):

- Buy(msg.sender,(amount << (64 + 32)) | (multiplierX100 << 64) | block.timestamp) (contracts/catsluck4erc20.sol#115)

Reentrancy in catsluck4fun.buyLottery(uint256,uint256) (contracts/catsluck4fun.sol#71-95):

External calls:

- getMyReward() (contracts/catsluck4fun.sol#77)

- IERC20(erc20Address).transfer(msg.sender,reward) (contracts/catsluck4fun.sol#114)

- (amount,oldBalance) = safeTransferFrom(msg.sender,amount) (contracts/catsluck4fun.sol#81)

- IERC20(erc20Address).transferFrom(addr,address(this),amount) (contracts/catsluck4fun.sol#22)

Event emitted after the call(s):

- Buy(msg.sender,(amount << (64 + 32)) | (multiplierX100 << 64) | block.timestamp) (contracts/catsluck4fun.sol#94)

Reentrancy in catsluck4erc20.deposit(uint256,uint256) (contracts/catsluck4erc20.sol#45-68):

External calls:

- (amount,oldBalance) = safeTransferFrom(msg.sender,amount) (contracts/catsluck4erc20.sol#47)

- IERC20(erc20Address).transferFrom(addr,address(this),amount) (contracts/catsluck4erc20.sol#40)

Event emitted after the call(s):

- Deposit(msg.sender,amount << (96 + 64) | (amount << 64) | block.timestamp) (contracts/catsluck4erc20.sol#58)

- Deposit(msg.sender,amount << (96 + 64) | (deltaShare << 64) | block.timestamp) (contracts/catsluck4erc20.sol#67)

Reentrancy in catsluck4fun.deposit(uint256,uint256) (contracts/catsluck4fun.sol#27-49):

External calls:

- (amount,oldBalance) = safeTransferFrom(msg.sender,amount) (contracts/catsluck4fun.sol#29)

- IERC20(erc20Address).transferFrom(addr,address(this),amount) (contracts/catsluck4fun.sol#22)

Event emitted after the call(s):

- Deposit(msg.sender,amount << (96 + 64) | (amount << 64) | block.timestamp) (contracts/catsluck4fun.sol#39)

- Deposit(msg.sender,amount << (96 + 64) | (deltaShare << 64) | block.timestamp) (contracts/catsluck4fun.sol#48)

Reentrancy in catsluck.depositCATS(uint256,uint256) (contracts/catsluck.sol#45-67):

External calls:

- (amount,oldBalance) = safeTransferFrom(msg.sender,amount) (contracts/catsluck.sol#47)

- IERC20(erc20Address).transferFrom(addr,address(this),amount) (contracts/

catsluck.sol#40)

Event emitted after the call(s):

- Deposit(msg.sender,amount << (96 + 64) | (amount << 64) | block.timestamp) (contracts/
catsluck.sol#56)

- Deposit(msg.sender,amount << (96 + 64) | (deltaShare << 64) | block.timestamp) (contracts/
catsluck.sol#66)

Reentrancy in catsluck.getMyReward() (contracts/catsluck.sol#117-138):

External calls:

- IERC20(erc20Address).transfer(msg.sender,reward) (contracts/catsluck.sol#134)

Event emitted after the call(s):

- Win(msg.sender,(reward << (64 + 32)) | (multiplierX100 << 64) | block.timestamp) (contracts/
catsluck.sol#135)

Reentrancy in catsluck4erc20.getMyReward() (contracts/catsluck4erc20.sol#118-139):

External calls:

- IERC20(erc20Address).transfer(msg.sender,reward) (contracts/catsluck4erc20.sol#135)

Event emitted after the call(s):

- Win(msg.sender,(reward << (64 + 32)) | (multiplierX100 << 64) | block.timestamp) (contracts/
catsluck4erc20.sol#136)

Reentrancy in catsluck4fun.getMyReward() (contracts/catsluck4fun.sol#97-118):

External calls:

- IERC20(erc20Address).transfer(msg.sender,reward) (contracts/catsluck4fun.sol#114)

Event emitted after the call(s):

- Win(msg.sender,(reward << (64 + 32)) | (multiplierX100 << 64) | block.timestamp) (contracts/catsluck4fun.sol#115)

Reentrancy in catsluck4erc20.withdraw(uint256) (contracts/catsluck4erc20.sol#75-89):

External calls:

- IERC20(erc20Address).transfer(msg.sender,deltaBalance) (contracts/catsluck4erc20.sol#87)

Event emitted after the call(s):

- Withdraw(msg.sender,deltaBalance << (96 + 64) | (deltaShare << 64) | block.timestamp) (contracts/catsluck4erc20.sol#88)

Reentrancy in catsluck4fun.withdraw(uint256) (contracts/catsluck4fun.sol#56-69):

External calls:

- IERC20(erc20Address).transfer(msg.sender,deltaBalance) (contracts/catsluck4fun.sol#67)

Event emitted after the call(s):

- Withdraw(msg.sender,deltaBalance << (96 + 64) | (deltaShare << 64) | block.timestamp) (contracts/catsluck4fun.sol#68)

Reentrancy in catsluck.withdrawCATS(uint256) (contracts/catsluck.sol#74-88):

External calls:

- IERC20(erc20Address).transfer(msg.sender,deltaBalance) (contracts/catsluck.sol#86)

Event emitted after the call(s):

- Withdraw(msg.sender,deltaBalance << (96 + 64) | (deltaShare << 64) | block.timestamp) (contracts/catsluck.sol#87)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

catsluck.withdrawCATS(uint256) (contracts/catsluck.sol#74-88) uses timestamp for comparisons

Dangerous comparisons:

- require(bool,string)(block.timestamp > lockUntil,still locked) (contracts/catsluck.sol#79)

catsluck.buyLottery(uint256,uint256) (contracts/catsluck.sol#90-115) uses timestamp for comparisons

Dangerous comparisons:

- block.timestamp < EndMintingTime (contracts/catsluck.sol#112)

catsluck4bch.deposit(uint256) (contracts/catsluck4bch.sol#31-53) uses timestamp for comparisons

Dangerous comparisons:

- lockUntil < oneHourLater (contracts/catsluck4bch.sol#38)

- oldShare > 0 (contracts/catsluck4bch.sol#39)

- require(bool,string)(lockUntil >= oldLockUntil,invalid lockUntil) (contracts/catsluck4bch.sol#40)

catsluck4bch.withdraw(uint256) (contracts/catsluck4bch.sol#60-73) uses timestamp for comparisons

Dangerous comparisons:

- require(bool,string)(oldShare >= deltaShare,not enough share) (contracts/catsluck4bch.sol#64)

- require(bool,string)(block.timestamp > lockUntil,still locked) (contracts/catsluck4bch.sol#65)

catsluck4erc20.deposit(uint256,uint256) (contracts/catsluck4erc20.sol#45-68) uses timestamp for comparisons

Dangerous comparisons:

- lockUntil < oneHourLater (contracts/catsluck4erc20.sol#53)

- require(bool,string)(lockUntil >= oldLockUntil,invalid lockUntil) (contracts/catsluck4erc20.sol#55)

catsluck4erc20.withdraw(uint256) (contracts/catsluck4erc20.sol#75-89) uses timestamp for comparisons

Dangerous comparisons:

- require(bool,string)(oldShare >= deltaShare,not enough share) (contracts/catsluck4erc20.sol#79)

- require(bool,string)(block.timestamp > lockUntil,still locked) (contracts/catsluck4erc20.sol#80)

catsluck4fun.deposit(uint256,uint256) (contracts/catsluck4fun.sol#27-49) uses timestamp for comparisons

Dangerous comparisons:

- lockUntil < oneHourLater (contracts/catsluck4fun.sol#34)

- require(bool,string)(lockUntil >= oldLockUntil,invalid lockUntil) (contracts/catsluck4fun.sol#36)

catsluck4fun.withdraw(uint256) (contracts/catsluck4fun.sol#56-69) uses timestamp for comparisons

Dangerous comparisons:

- require(bool,string)(oldShare >= deltaShare,not enough share) (contracts/catsluck4fun.sol#60)

- require(bool,string)(block.timestamp > lockUntil,still locked) (contracts/catsluck4fun.sol#61)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Different versions of Solidity is used:

- Version used: ['0.8.10', '^0.8.0']

- ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#4)

- ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4)

- ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#4)

- ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4)

- 0.8.10 (contracts/catsluck.sol#2)

- 0.8.10 (contracts/catsluck4bch.sol#2)

- 0.8.10 (contracts/catsluck4erc20.sol#2)

- 0.8.10 (contracts/catsluck4fun.sol#2)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6

Pragma version0.8.10 (contracts/catsluck.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6

Pragma version0.8.10 (contracts/catsluck4bch.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6

Pragma version0.8.10 (contracts/catsluck4erc20.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6

Pragma version0.8.10 (contracts/catsluck4fun.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6

solc-0.8.10 is not recommended for deployment

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Pragma version0.8.10 (contracts/comment.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in catsluck4bch.withdraw(uint256) (contracts/catsluck4bch.sol#60-73):

- msg.sender.call{value: deltaBalance}() (contracts/catsluck4bch.sol#72)

Low level call in catsluck4bch.getMyReward() (contracts/catsluck4bch.sol#103-124):

- msg.sender.call{value: reward}() (contracts/catsluck4bch.sol#121)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

fun (contracts/fun.sol#8-69) should inherit from IERC20 (contracts/fun.sol#4-6)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-inheritance

Contract fun (contracts/fun.sol#8-69) is not in CapWords

Constant fun.catsAddress (contracts/fun.sol#12) is not in UPPER_CASE_WITH_UNDERSCORES

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Contract catsluck (contracts/catsluck.sol#15-150) is not in CapWords

Constant catsluck.erc20Address (contracts/catsluck.sol#16) is not in UPPER_CASE_WITH_UNDERSCORES

Constant catsluck.routerAddress (contracts/catsluck.sol#17) is not in UPPER_CASE_WITH_UNDERSCORES

Constant catsluck.CatsDecimal (contracts/catsluck.sol#18) is not in UPPER_CASE_WITH_UNDERSCORES

Constant catsluck.EndMintingTime (contracts/catsluck.sol#22) is not in UPPER_CASE_WITH_UNDERSCORES

Contract catsluck4bch (contracts/catsluck4bch.sol#14-137) is not in CapWords

Constant catsluck4bch.routerAddress (contracts/catsluck4bch.sol#15) is not in UPPER_CASE_WITH_UNDERSCORES

Constant catsluck4bch.clkAddress (contracts/catsluck4bch.sol#16) is not in UPPER_CASE_WITH_UNDERSCORES

Constant catsluck4bch.wbchAddress (contracts/catsluck4bch.sol#17) is not in UPPER_CASE_WITH_UNDERSCORES

Contract catsluck4erc20 (contracts/catsluck4erc20.sol#15-153) is not in CapWords

Constant catsluck4erc20.routerAddress (contracts/catsluck4erc20.sol#17) is not in UPPER_CASE_WITH_UNDERSCORES

Constant catsluck4erc20.clkAddress (contracts/catsluck4erc20.sol#18) is not in UPPER_CASE_WITH_UNDERSCORES

Constant catsluck4erc20.wbchAddress (contracts/catsluck4erc20.sol#19) is not in UPPER_CASE_WITH_UNDERSCORES

Contract catsluck4fun (contracts/catsluck4fun.sol#6-119) is not in CapWords

Constant catsluck4fun.erc20Address (contracts/catsluck4fun.sol#7) is not in UPPER_CASE_WITH_UNDERSCORES

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Contract comment (contracts/comment.sol#4-42) is not in CapWords

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

catsluck.buyLottery(uint256,uint256) (contracts/catsluck.sol#90-115) uses literals with too many digits:

- require(bool,string)(105 <= multiplierX100 && multiplierX100 <= 100000,invalid multiplier) (contracts/catsluck.sol#106)

catsluck.buyback() (contracts/catsluck.sol#140-149) uses literals with too many digits:

- amounts = IBenSwapRouter(routerAddress).swapExactTokensForTokens(oldBuybackPoolFunds,0,path,address(1),9000000000) (contracts/catsluck.sol#145-146)

catsluck4bch.buyLottery(uint256) (contracts/catsluck4bch.sol#75-101) uses literals with too many digits:

- require(bool,string)(105 <= multiplierX100 && multiplierX100 <= 100000,invalid multiplier) (contracts/catsluck4bch.sol#90)

catsluck4bch.buyback() (contracts/catsluck4bch.sol#126-136) uses literals with too many digits:

- amounts = IBenSwapRouterNative(routerAddress).swapExactETHForTokens{value: oldBuybackPoolFunds}(0,path,address(1),9000000000) (contracts/catsluck4bch.sol#132-133)

catsluck4erc20.buyLottery(uint256,uint256) (contracts/catsluck4erc20.sol#91-116) uses literals with too many digits:

- require(bool,string)(105 <= multiplierX100 && multiplierX100 <= 100000,invalid multiplier) (contracts/catsluck4erc20.sol#105)

catsluck4erc20.buyback() (contracts/catsluck4erc20.sol#141-152) uses literals with too many digits:

- amounts = IBenSwapRouter(routerAddress).swapExactTokensForTokens(oldBuybackPoolFunds,0,path,address(1),9000000000) (contracts/catsluck4erc20.sol#148-149)

catsluck4fun.buyLottery(uint256,uint256) (contracts/catsluck4fun.sol#71-95) uses literals with too many digits:

- require(bool,string)(105 <= multiplierX100 && multiplierX100 <= 100000,invalid multiplier) (contracts/catsluck4fun.sol#84)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

mint() should be declared external:

- fun.mint() (contracts/fun.sol#25-31)

totalSupply() should be declared external:

- fun.totalSupply() (contracts/fun.sol#33-35) balanceOf(address) should be declared external:

- fun.balanceOf(address) (contracts/fun.sol#37-39)

transfer(address,uint256) should be declared external:

- fun.transfer(address,uint256) (contracts/fun.sol#41-47)

approve(address,uint256) should be declared external:

- fun.approve(address,uint256) (contracts/fun.sol#49-53)

allowance(address,address) should be declared external:

- fun.allowance(address,address) (contracts/fun.sol#55-57)

transferFrom(address,address,uint256) should be declared external:

- fun.transferFrom(address,address,uint256) (contracts/fun.sol#59-68)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external

name() should be declared external:

- ERC20.name() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#62-64)

symbol() should be declared external:

- ERC20.symbol() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#70-72)

decimals() should be declared external:

- ERC20.decimals() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#87-89)

totalSupply() should be declared external:

- ERC20.totalSupply() (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#94-96)

balanceOf(address) should be declared external:

- ERC20.balanceOf(address) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#101-103)

transfer(address,uint256) should be declared external:

- ERC20.transfer(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#113-116)

allowance(address,address) should be declared external:

- ERC20.allowance(address,address) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#121-123)

approve(address,uint256) should be declared external:

- ERC20.approve(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#132-135)

transferFrom(address,address,uint256) should be declared external:

- ERC20.transferFrom(address,address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#150-164)

increaseAllowance(address,uint256) should be declared external:

- ERC20.increaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#178-181)

decreaseAllowance(address,uint256) should be declared external:

- ERC20.decreaseAllowance(address,uint256) (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#197-205)

Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external

. analyzed (14 contracts with 75 detectors), 119 result(s) found

0x Guard