

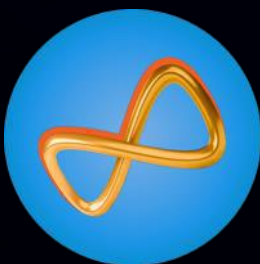


Request your audit at coinsult.net

Advanced Manual

Smart Contract Audit

May 1, 2023



Audit requested by

Moox DrawManager

0xcc52be74e94c2afef596b8a9f95a35e0a8fecc21

Audit Summary

Project Name	Moox
Website	moox.one
Blockchain	Binance Smart Chain
Smart Contract Language	Solidity
Contract Address	0xcc52be74e94c2afef596b8a9f95a35e0a8fecc21
Audit Method	Static Analysis, Manual Review
Start date of audit	May 1, 2023

This audit report has been prepared by Coinsult's experts at the request of the client. In this audit, the results of the static analysis and the manual code review will be presented. The purpose of the audit is to see if the functions work as intended, and to identify potential security issues within the smart contract.

The information in this report should be used to understand the risks associated with the smart contract. This report can be used as a guide for the development team on how the contract could possibly be improved by remediating the issues that were identified.

The audit of the Solidity codebase presented significant challenges due to its disorganized nature and lack of accompanying documentation. The convoluted structure of the code made it increasingly difficult to discern the intended functionality of individual components. Furthermore, the absence of any guiding documentation surrounding the functions made it even more arduous to review and assess the overall quality of the code. Despite these obstacles, the auditing process was diligently carried out to ensure the highest standards of security and stability.

Audit Scope

Source Code

Coinsult was commissioned by MooX to perform an audit based on the following code:

<https://github.com/mooxtoken/moox-contract/tree/master/contracts/v2>

Note that we only audited the code available to us on this URL at the time of the audit. If the URL is not from any block explorer (main net), it may be subject to change. Always check the contract address on this audit report and compare it to the token you are doing research for.

Audit Method

Coinsult's manual smart contract audit is an extensive methodical examination and analysis of the smart contract's code that is used to interact with the blockchain. This process is conducted to discover errors, issues and security vulnerabilities in the code in order to suggest improvements and ways to fix them.

Automated Vulnerability Check

Coinsult uses software that checks for common vulnerability issues within smart contracts. We use automated tools that scan the contract for security vulnerabilities such as integer-overflow, integer-underflow, out-of-gas-situations, unchecked transfers, etc.

Manual Code Review

Coinsult's manual code review involves a human looking at source code, line by line, to find vulnerabilities. Manual code review helps to clarify the context of coding decisions. Automated tools are faster but they cannot take the developer's intentions and general business logic into consideration.





Used Tools

- ✓ Slither: Solidity static analysis framework
- ✓ Remix: IDE Developer Tool
- ✓ CWE: Common Weakness Enumeration
- ✓ SWC: Smart Contract Weakness Classification and Test Cases
- ✓ DEX: Testnet Blockchains

Audit Results





Risk Classification

Coinsult uses certain vulnerability levels, these indicate how bad a certain issue is. The higher the risk, the more strictly it is recommended to correct the error before using the contract.

Vulnerability Level	Description
 Informational	Does not compromise the functionality of the contract in any way
 Low-Risk	Won't cause any problems, but can be adjusted for improvement
 Medium-Risk	Will likely cause problems and it is recommended to adjust
 High-Risk	Will definitely cause problems, this needs to be adjusted

Manual Code Review

In this audit report we will highlight the following issues:

Vulnerability Level	Total	Pending	Acknowledged	Resolved
 Informational	0	0	0	0
 Low-Risk	13	0	7	6
 Medium-Risk	0	0	0	0
 High-Risk	0	0	0	0

Error Code	Description	Severity
CNS-010	DrawManager.sol: Tx.origin as authorization	<div><div></div> Logical</div> <div><div></div> Acknowledged</div>

Issue

tx.origin-based protection can be abused by a malicious contract if a legitimate user interacts with the malicious contract.

Code

```
UserInfo storage userInfo = userInfoMap[tx.origin];  
...  
emit TicketAward(tx.origin, _amount, ticketCount, 1);
```

Recommendation

tx.origin should not be used for authorization in smart contracts.

Moox Comment

tx.origin is not used for authorization here.

Even if this function is called over a malicious contract, tx.origin will only earn tickets.

Error Code	Description	Severity
CNS-011	DrawManager.sol: The return value of an external transfer/transferFrom call is not checked	<div><div></div> Logical</div> <div><div></div> Fixed</div>

Issue

The return value of an external transfer/transferFrom call is not checked

Code

```
LINK.transferFrom(msg.sender, address(this), requestPrice);
```

Recommendation

Check the return value

Moox Comment

Fixed

Error Code	Description	Severity
CNS-012	DrawManager.sol: Unchecked return value on constructor	<div><div></div> Logical</div> <div><div></div> Fixed</div>

Issue

Unchecked return value on constructor

Code

```
busd.approve(address(pancakeswapRouter), type(uint256).max);
```

Recommendation

Check the return value

Moxx Comment

Fixed

Error Code	Description	Severity
CNS-013	DrawManager.sol: The return value of an external transfer/transferFrom call is not checked	<div> ● Logical </div> <div> ● Fixed </div>

Issue

The return value of an external transfer/transferFrom call is not checked

Code

```

busd.approve(address(pancakeswapRouter), type(uint256).max);
...
pancakeswapRouter.swapExactTokensForTokens(busdAmount, minMooxAmount, path,
address(this), block.timestamp);

```

Recommendation

Check the return value

Moox Comment

Fixed

Error Code	Description	Severity
CNS-014	DrawManager.sol: Lack of zero wallet checks on the constructor	<div><div></div> Logical</div> <div><div></div> Fixed</div>

Issue

Lack of zero wallet checks on the constructor

Code

```
charityWalletSetter = _creator;  
marketingWalletSetter = _creator;
```

Recommendation

Check that the address is not zero

Moxx Comment

Fixed

Error Code	Description	Severity
CNS-015	DrawManager.sol: Denial-of-service attack.	<div><div>● Logical</div><div>● Acknowledged</div></div>

Issue

DrawManager.draw() has external calls inside a loop might lead to a denial-of-service attack. moox.transfer(winners[i], award) if the number of iterations is large. This is because each call incurs a certain amount of gas cost, and if the loop iterates too many times, it can lead to a high gas consumption and possible out-of-gas errors. This can make the contract unusable or vulnerable to attacks like denial-of-service (DoS)

Code

```
for (uint256 i = 0; i < roundWinnerCount; i++) {  
    if (moox.transfer(winners[i], award)) {  
        emit AwardUser(roundCount, winnerTicketNumbers[i], winners[i], award);  
    } else {  
        emit MooxTransferFailed(winners[i], award);  
    }  
}
```

Recommendation

Make sure the DoS attack cannot happen.

Moox Comment

This function is tested with extreme cases. Maximum winner count is 50, and tested for 10,000,000 draw entries which is almost impossible. No out-of-gas error is expected.

Error Code	Description	Severity
CNS-016	DrawManager.sol: Too many digits	<div><div></div> Logical</div> <div><div></div> Fixed</div>

Issue

The use of very large numbers with too many digits was detected in the code that could have been optimized using a different notation also supported by Solidity

Code

```
uint32 private constant RNG_CALLBACK_GAS_LIMIT = 100000;
```

Recommendation

Use: [Ether suffix](#), [Time suffix](#), or [The scientific notation](#)

Moox Comment

This part of the code is removed. RNG parameter values will be passed from frontend.

Error Code	Description	Severity
CNS-017	DrawManager.sol: No events emitted	<div><div></div> Logical</div> <div><div></div> Acknowledged</div>

Issue

The contract DrawManager was found to be missing these events on the function requestRandomNumber() which would make it difficult or impossible to track these transactions off-chain

Code

```
function requestRandomNumber() external returns (uint256 requestId) {
    require(drawTime > INITIAL_DRAW_TIME && drawTime <= block.timestamp, 'DrawManager: Invalid time');

    uint256 requestPrice =
VRF_V2_WRAPPER.calculateRequestPrice(RNG_CALLBACK_GAS_LIMIT);

    LINK.transferFrom(msg.sender, address(this), requestPrice);

    requestId = requestRandomness(RNG_CALLBACK_GAS_LIMIT, RNG_REQUEST_CONFIRMATIONS,
RNG_RANDOM_NUMBER_COUNT);

    rngRequestStatusMap[requestId] = RngRequestStatus({
        paid: requestPrice,
        randomWords: new uint256[](0),
        fulfilled: false
    });

    rngRequestId = requestId;
}
```

Recommendation

Emit an event.

Moxx Comment

Acknowledged, won't fix.

Error Code	Description	Severity
CNS-018	DrawManager.sol: No events emitted	<div><div></div> Logical</div> <div><div></div> Acknowledged</div>

Issue

The contract DrawManager was found to be missing these events on the function requestRandomNumber() which would make it difficult or impossible to track these transactions off-chain

Code

```
function fulfillRandomWords(uint256 _requestId, uint256[] memory _randomWords)
internal override {
    require(rngRequestStatusMap[_requestId].paid > 0, 'DrawManager: Rng request not
paid');

    rngRequestStatusMap[_requestId].fulfilled = true;
    rngRequestStatusMap[_requestId].randomWords = _randomWords;
}
```

Recommendation

Emit an event.

Moxx Comment

Acknowledged, won't fix.

Error Code	Description	Severity
CNS-019	DrawManager.sol: No events emitted	<div><div></div> Logical</div> <div><div></div> Acknowledged</div>

Issue

The contract DrawManager was found to be missing these events on the function requestRandomNumber() which would make it difficult or impossible to track these transactions off-chain

Code

```
function startDraw() external {
    require(msg.sender == marketingWalletSetter, 'DrawManager: Not allowed');
    require(drawTime == INITIAL_DRAW_TIME, 'DrawManager: Already set');

    drawTime +=
block.timestamp.sub(drawTime).div(DRAW_TIME_BUFFER).add(1).mul(DRAW_TIME_BUFFER);
}
```

Recommendation

Emit an event.

Moox Comment

Acknowledged, won't fix.

Error Code	Description	Severity
CNS-020	DrawManager.sol: Outdated floating solidity version	<div><div></div> Logical</div> <div><div></div> Acknowledged</div>

Issue

Outdated floating versions were detected pragma solidity ^0.8.0;

Code

```
pragma solidity ^0.8.0;
```

Recommendation

Use a fixed recent version of solidity

Moxx Comment

Acknowledged, won't fix.

Error Code	Description	Severity
CNS-021	DrawManager.sol: Hardcoded gas limit	<div><div></div> Logical</div> <div><div></div> Fixed</div>

Issue

The contract defined hardcoded gas limits in the variable RNG_CALLBACK_GAS_LIMIT. Hardcoded gas limits are not future-proof as some hard forks may introduce changes in gas costs. If some hard fork in the future breaks the smart contract, you would need to deploy new contracts with adjusted limits

Code

```
uint32 private constant RNG_CALLBACK_GAS_LIMIT = 100000;
```

Recommendation

Don't use hardcoded gas limits

Moxx Comment

This part of the code is removed. RNG parameter values will be passed from frontend.

Error Code	Description	Severity
CNS-022	DrawManager.sol: Operator gas optimization	<div><div></div> Logical</div> <div><div></div> Acknowledged</div>

Issue

When optimization is disabled, $x > 0$ is cheaper than $x \neq 0$ for unsigned integers in solidity

Code

```
require(rngRequestId != 0, 'DrawManager: Update random number');
```

Recommendation

Use $x > 0$

Moxx Comment

Acknowledged, won't fix.

Notes

Notes by Moox

No notes provided by the team.

Notes by Coinsult

These contracts have been audited based on GitHub code, this code is updateable.

Disclaimer

This audit report has been prepared by Coinsult's experts at the request of the client. In this audit, the results of the static analysis and the manual code review will be presented. The purpose of the audit is to see if the functions work as intended, and to identify potential security issues within the smart contract.

The information in this report should be used to understand the risks associated with the smart contract. This report can be used as a guide for the development team on how the contract could possibly be improved by remediating the issues that were identified.

Coinsult is not responsible if a project turns out to be a scam, rug-pull or honeypot. We only provide a detailed analysis for your own research.

Coinsult is not responsible for any financial losses. Nothing in this contract audit is financial advice, please do your own research.

The information provided in this audit is for informational purposes only and should not be considered investment advice. Coinsult does not endorse, recommend, support or suggest to invest in any project.

Coinsult can not be held responsible for when a project turns out to be a rug-pull, honeypot or scam.

The audit of the Solidity codebase presented significant challenges due to its disorganized nature and lack of accompanying documentation. The convoluted structure of the code made it increasingly difficult to discern the intended functionality of individual components. Furthermore, the absence of any guiding documentation surrounding the functions made it even more arduous to review and assess the overall quality of the code. Despite these obstacles, the auditing process was diligently carried out to ensure the highest standards of security and stability.