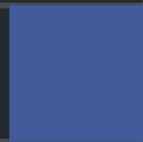




Security Assessment

MOBOX

Apr 30th, 2021



Summary

This report has been prepared for MOBOX smart contracts, to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Dynamic Analysis, Static Analysis, and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases given they are currently missing in the repository;
- Provide more comments per each function for readability, especially contracts are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	MOBOX
Description	DeFi
Platform	BSC
Language	Solidity
Codebase	https://github.com/moboxio/NFTfarmer/tree/f3b6a2176ced7828867d366648063f3b672cd212
Commits	9443add71a3dadfefab237ac2b016c942d16e204

Audit Summary

Delivery Date	Apr 30, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

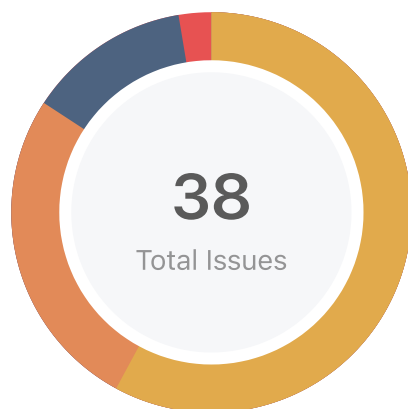
Vulnerability Summary

Total Issues	38
● Critical	1
● Major	10
● Minor	22
● Informational	5
● Discussion	0

Audit Scope

ID	file	SHA256 Checksum
KTM	KeyToken.sol	0ca758d4ea6dfcc2570330f5f3e917fc7d5e57d2c48e574e0565006db9cc5e4d
MFM	MoboxFarm.sol	175b601762befc9dc4e228144ea1c14d1a52305a886e1ec057f5b67773c05048
MSP	MoboxStrategyP.sol	f4fdac6c357352990cfe144ac9417eb67597aba9e76b0ce64ccb3ee5f0cfda48
MSV	MoboxStrategyV.sol	5953f3c3ad07bc482fea4d32f13dd7eeadedbefc25efb48e37b5329d443413a1
MTM	MoboxToken.sol	40ada3a9edf02f55eb862482e85d56cafc998da8794822c4b739c0c56b65b0a1
MSM	MomoStaker.sol	569c40585b02e7daad1adf2300146f04b51a70357905428a3e5889578fae6ecc

Findings



■ Critical	1 (2.63%)
■ Major	10 (26.32%)
■ Minor	22 (57.89%)
■ Informational	5 (13.16%)
■ Discussion	0 (0.00%)

ID	Title	Category	Severity	Status
MOBOX-01	Privileged Ownerships	Centralization / Privilege	● Minor	ⓘ Pending
CMO-01	Redundant Expression	Volatile Code	● Minor	ⓘ Pending
KTM-01	Missing Zero Address Validation	Volatile Code	● Minor	ⓘ Acknowledged
MFM-01	Implicitly Return Values	Coding Style	● Informational	ⓘ Acknowledged
MFM-02	Missing Return Value Handling	Logical Issue	● Minor	☑ Resolved
MFM-03	Dangerous Usage of block.timestamp	Logical Issue	● Minor	ⓘ Acknowledged
MFM-04	Missing Checks for Reentrancy	Logical Issue	● Major	☑ Resolved
MSM-01	Divide Before Multiply	Logical Issue	● Minor	ⓘ Pending
MSM-02	userHashratePercent is Undefined for One amountV6 Holders	Logical Issue	● Major	ⓘ Pending
MSM-03	Incorrect ERC20 Interface	Volatile Code	● Minor	ⓘ Pending
MSM-04	Dangerous Strict Equalities	Logical Issue	● Minor	ⓘ Pending
MSM-05	Inaccurate Parameter of HashrateChange Event	Logical Issue	● Major	ⓘ Pending
MSM-06	Stake Nft without doing real transfer	Logical Issue	● Major	ⓘ Pending
MSM-07	Reentrancy vulnerabilities	Logical Issue	● Major	ⓘ Pending

ID	Title	Category	Severity	Status
MSM-08	Uninitialized Local Variables	Volatile Code	● Minor	⊕ Pending
MSM-09	Unused Return	Volatile Code	● Minor	⊕ Pending
MSM-10	3rd Party Dependencies	Control Flow	● Minor	⊕ Pending
MSM-11	Missing Zero Address Validation	Control Flow	● Minor	⊕ Pending
MSM-12	Calls Inside A Loop	Logical Issue	● Minor	⊕ Pending
MSM-13	Public Function Could Be Declared External	Gas Optimization	● Minor	⊕ Pending
MSP-01	Integer Overflow	Mathematical Operations	● Minor	⊙ Resolved
MSP-02	Missing Checks for Reentrancy	Logical Issue	● Major	⊙ Resolved
MSP-03	Missing Zero Address Validation	Logical Issue	● Minor	⊙ Resolved
MSP-04	Missing slippage protection	Logical Issue	● Minor	ⓘ Acknowledged
MSP-05	Wrong Withdraw Amount	Logical Issue	● Major	⊙ Resolved
MSP-06	Implicitly Return Values	Coding Style	● Informational	ⓘ Acknowledged
MSP-07	Unimpletation Constructor Function	Volatile Code	● Informational	⊙ Resolved
MSP-08	3rd Party Dependencies	Control Flow	● Minor	ⓘ Acknowledged
MSV-01	Integer Overflow	Mathematical Operations	● Minor	⊙ Resolved
MSV-02	Missing slippage protection	Logical Issue	● Minor	ⓘ Acknowledged
MSV-03	Compile Error	Compiler Error	● Critical	⊙ Resolved
MSV-04	Missing Checks for Reentrancy	Logical Issue	● Major	⊙ Resolved
MSV-05	Missing Access Control	Logical Issue	● Major	⊙ Resolved
MSV-06	Implicitly Return Values	Coding Style	● Informational	ⓘ Acknowledged
MSV-07	Ignored Return Values	Logical Issue	● Major	ⓘ Acknowledged
MSV-08	Leverage risk	Logical Issue	● Informational	⊙ Resolved

ID	Title	Category	Severity	Status
MSV-09	3rd Party Dependencies	Control Flow	● Minor	ⓘ Acknowledged
MTM-01	Mint to _dest address instead of mining pool	Logical Issue	● Minor	ⓘ Pending

MOBOX-01 | Privileged Ownerships

Category	Severity	Location	Status
Centralization / Privilege	● Minor	Global	ⓘ Pending

Description

The owner and Strategist has the permission to setDevTeam, setFeeRate and other global controls without obtaining the consensus of the community.

Recommendation

Renounce ownership when it is the right timing, or gradually migrate to a timelock plus multisig governing procedure and let the community monitor in respect of transparency considerations.

CMO-01 | Redundant Expression

Category	Severity	Location	Status
Volatile Code	● Minor	comm/Context.sol: 15~25(Context)	🕒 Pending

Description

```
function _msgData() internal view virtual returns (bytes memory) {
    this; // silence state mutability warning without generating bytecode - see
https://github.com/ethereum/solidity/issues/2691
    return msg.data;
}
```

Redundant expression "this (comm/Context.sol#21)" inContext (comm/Context.sol#15-25)

Recommendation

Remove redundant statements if they congest code but offer no value.

KTM-01 | Missing Zero Address Validation

Category	Severity	Location	Status
Volatile Code	● Minor	KeyToken.sol: 36	ⓘ Acknowledged

Description

lacks a zero-check on : - moboxFarm = farm_ (KeyToken.sol#36)

Recommendation

Adding check that farm_ is not zero. If zero address is used to stop mining, then similar check like `require(msg.sender == moboxFarm, "not farm")` should also be added to `mintForEvent()`.

MFM-01 | Implicitly Return Values

Category	Severity	Location	Status
Coding Style	● Informational	MoboxFarm.sol: 251	ⓘ Acknowledged

Description

Functions defined with return values but values are returned implicitly.

Recommendation

We recommend always return values explicitly.

MFM-02 | Missing Return Value Handling

Category	Severity	Location	Status
Logical Issue	● Minor	MoboxFarm.sol: 110~111	🕒 Resolved

Description

Approve is not a void-returning function per IERC20 interface. Ignoring the return value might cause some unexpected exception, especially if the callee function doesn't revert automatically when failing.

Recommendation

We recommend checking the return value before continuing processing.

MFM-03 | Dangerous Usage of block.timestamp

Category	Severity	Location	Status
Logical Issue	● Minor	MoboxFarm.sol: 126, 170, 181, 206, 229, 234, 297, 309	① Acknowledged

Description

block.timestamp can be manipulated by miners.

Recommendation

Avoid relying on block.timestamp.

MFM-04 | Missing Checks for Reentrancy

Category	Severity	Location	Status
Logical Issue	● Major	MoboxFarm.sol: 461~462, 178, 123, 450, 107, 148, 201, 328	🕒 Resolved

Description

Functions have state updates or event emits after external calls are vulnerable to reentrancy attack.

Recommendation

We recommend applying OpenZeppelin ReentrancyGuard library - nonReentrant modifier for the aforementioned functions to prevent reentrancy attack.

MSM-01 | Divide Before Multiply

Category	Severity	Location	Status
Logical Issue	● Minor	MomoStaker.sol: 200~204(MomoStaker), 311~360(MomoStaker)	ⓘ Pending

Description

`MoMoStaker.earned(address,uint256)` (*MomoStaker.sol#200-204*) performs a multiplication on the result of a division: `-userHashRate = uint256(info.userHashrateFixed).mul(uint256(info.userHashratePercent) + 10000).div(10000)` (*MomoStaker.sol#202*) - `userHashRate.mul(rewardPerHashrate.sub(uint256(info.userRewardPerTokenPaid))).add(uint256(info.userReward))` (*MomoStaker.sol#203*)

For example,Line #323:

```
else if (amount <= 200) {
    percent = (amount - 100) / 25 * 100 + 1700;    // 17% ~ 21%
}
```

If `25` is greater than `amount - 100`, result will be zero. In general, it's usually a good idea to re-arrange arithmetic to perform multiplication before division, unless the limit of a smaller type makes this dangerous.

Recommendation

Consider ordering multiplication before division.

MSM-02 | userHashratePercent is Undefined for One amountV6 Holders

Category	Severity	Location	Status
Logical Issue	● Major	MomoStaker.sol: 347	ⓘ Pending

Description

```
345 // V6 collection 346 amount = uint256(counter.amountV6); 347 if (amount > 1) { 348 if (amount <= 2) {
```

userHashratePercent is undefined, if "amount == 1".

Recommendation

We recommend to return userHashratePercent explicitly for each possible amount.

MSM-03 | Incorrect ERC20 Interface

Category	Severity	Location	Status
Volatile Code	● Minor	MomoStaker.sol: 10~16(MomoStaker)	⚠ Pending

Description

IMoMoToken (MomoStaker.sol#10-16) has incorrect ERC20 function

interface:IMoMoToken.transferFrom(address,address,uint256) (MomoStaker.sol#11)

Incorrect return values for ERC20 functions. A contract compiled with Solidity > 0.4.22 interacting with these functions will fail to execute them, as the return value is missing.

Exploit Scenario:

```
interface IMoMoToken {
    function transferFrom(address from_, address to_, uint256 tokenId_) external;
    function getMomoSimpleByTokenId(uint256 tokenId_) external view returns(uint256,
uint256);
    function levelUp(uint256 tokenId_, uint256[] memory protosV1V2V3_, uint256[] memory
amountsV1V2V3_, uint256[] memory tokensV4V5_) external;
    function setMomoName(uint256 tokenId_, bytes memory name_) external payable;
    function addMomoStory(uint256 tokenId_, bytes memory story_) external payable;
}
```

`IMoMoToken.transferFrom()` does not return a `boolean`. Bob deploys the token. Alice creates a contract that interacts with it but assumes a correct `ERC20` interface implementation. Alice's contract is unable to interact with Bob's contract.

Recommendation

Set the appropriate return values and types for the defined `ERC20` functions.

MSM-04 | Dangerous Strict Equalities

Category	Severity	Location	Status
Logical Issue	● Minor	MomoStaker.sol: 187~194(MomoStaker)	ⓘ Pending

Description

Use of strict equalities that can be easily manipulated by an attacker.

MoMoStaker._rewardPerHashrate(uint256) (MomoStaker.sol#187-194) uses a dangerous strict equality:

```
function _rewardPerHashrate(uint256 lastTimeRewardApplicable_) internal view returns
(uint256) {
    if (totalHashrate == 0 || block.timestamp <= rewardStartTime) {
        return rewardPerTokenStored;
    }
    return rewardPerTokenStored.add(
lastTimeRewardApplicable_.sub(lastUpdateTime).mul(rewardRate).div(totalHashrate)
    );
}
```

_rewardPerHashrate relies on `totalHashrate == 0 || block.timestamp <= rewardStartTime` to know rewardPerHashrate or RewardPerTokenStored.

Recommendation

Don't use strict equality to determine if an account has enough Ether or tokens.

MSM-05 | Inaccurate Parameter of HashrateChange Event

Category	Severity	Location	Status
Logical Issue	● Major	MomoStaker.sol: 542, 555, 600	⚠ Pending

Description

```
33 // changeType: 1 stake/2 mint and stake/3 withdraw/4 level up/5 create auction/6  
cancel auction/7 bid auct ion  
34 event HashrateChange(address indexed user, uint256 changeType, uint256  
oldhashRate, uint256 newHashRate);  
35
```

There is no promise HashrateChange will emit with the right changeType as the comment.

```
600 uint256 hashrateFixedSub = _removeNft(msg.sender, protosV1V2V3_,  
amountsV1V2V3_, tokensV4V5_, 0x02, 0) ;
```

Also this call happens in level up but call `_removeNft()` with `changeType=0`.

Recommendation

We recommend to set the right changeType of HashrateChange directly like in Line 612. 612 emit `HashrateChange(msg.sender, 4, oldHashRate, newHashRate);`

And check all calling to `HashrateChange()` with the right changeType.

MSM-06 | Stake Nft without doing real transfer

Category	Severity	Location	Status
Logical Issue	● Major	MomoStaker.sol: 542~543, 588	ⓘ Pending

Description

```
_stakeNft(msg.sender, ids, vals, tokenIds, false, 2);
```

There are places call "_stakeNft()" without doing real transfer.

Recommendation

We recommend to confirm whether this is the right behavior.

MSM-07 | Reentrancy vulnerabilities

Category	Severity	Location	Status
Logical Issue	Major	MomoStaker.sol: 442~530(MomoStaker), 362~438(MomoStaker), 580~590 (MomoStaker), 131~141(MomoStaker)	ⓘ Pending

Description

Reentrancy in `MoMoStaker.removeNft(address,uint256[],uint256[],uint256[],uint256,uint256)`

(`MomoStaker.sol#442-530`): External calls: -

`momomToken.safeBatchTransferFrom(address(this),user,ids,amounts_)` (`MomoStaker.sol#475`) -

`momomToken.transferFrom(address(this),user,tokenIds_[i])` (`MomoStaker.sol#505`) State variables

written after the call(s): - `counter.amountV4 = SafeMathExt.sub64(counter.amountV4,uint64(1))`

(`MomoStaker.sol#509`) - `counter.amountV5 = SafeMathExt.sub64(counter.amountV5,1)`

(`MomoStaker.sol#519`) - `counter.amountV6 = SafeMathExt.sub64(counter.amountV6,1)`

(`MomoStaker.sol#521`)

Reentrancy in `MoMoStaker.stakeNft(address,uint256[],uint256[],uint256[],bool,uint256)`

(`MomoStaker.sol#362-438`): External calls: -

`momomToken.safeBatchTransferFrom(user,address(this),ids,amounts_)` (`MomoStaker.sol#380`) -

`momomToken.transferFrom(user,address(this),tokenIds_[i])` (`MomoStaker.sol#407`) State variables

written after the call(s): - `momos.push(tokenIds_[i])` (`MomoStaker.sol#414`) - `counter.amountV4 =`

`SafeMathExt.add64(counter.amountV4,uint64(1))` (`MomoStaker.sol#417`) - `counter.amountV5 =`

`SafeMathExt.add64(counter.amountV5,1)` (`MomoStaker.sol#427`) - `counter.amountV6 =`

`SafeMathExt.add64(counter.amountV6,1)` (`MomoStaker.sol#429`)

Reentrancy in `MoMoStaker.mintAndStake(uint256)` (`MomoStaker.sol#580-590`): External calls: -

`(ids,vals,tokenIds) = momoMinter.mintByStaker(msg.sender,amount_)` (`MomoStaker.sol#587`) -

`stakeNft(msg.sender,ids,vals,tokenIds,false,2)` (`MomoStaker.sol#588`) -

`momomToken.safeBatchTransferFrom(user,address(this),ids,amounts_)` (`MomoStaker.sol#380`) -

`momomToken.transferFrom(user,address(this),tokenIds_[i])` (`MomoStaker.sol#407`) State variables

written after the call(s): - `stakeNft(msg.sender,ids,vals,tokenIds,false,2)` (`MomoStaker.sol#588`) -

`info.userHashrateFixed =`

`SafeMathExt.sub128(info.userHashrateFixed,SafeMathExt.safe128(hashrateFixedSub))`

(`MomoStaker.sol#563`) - `info.userHashrateFixed =`

`SafeMathExt.add128(info.userHashrateFixed,SafeMathExt.safe128(hashrateFixedAdd_))`

(`MomoStaker.sol#567`) - `info.userHashratePercent = SafeMathExt.safe128(checkCollection(user))`

(MomoStaker.sol#570) - `_stakeNft(msg.sender,ids,vals,tokenIds,false,2)` (MomoStaker.sol#588) -
`totalHashrate = totalHashrate.add(newHashRate).sub(oldHashRate)` (MomoStaker.sol#572)

Recommendation

Apply the check-effects-interactions pattern.

MSM-08 | Uninitialized Local Variables

Category	Severity	Location	Status
Volatile Code	● Minor	MomoStaker.sol: 376(MomoStaker)	ⓘ Pending

Description

MoMoStaker._stakeNft(address,uint256[],uint256[],uint256[],bool,uint256).hashrateFixed (MomoStaker.sol#376) `hashrateFixed` is a local variable never initialized

Recommendation

Initialize all the variables. If a variable is meant to be initialized to zero, explicitly set it to zero to improve code readability.

MSM-09 | Unused Return

Category	Severity	Location	Status
Volatile Code	● Minor	MomoStaker.sol: 134(MomoStaker), 140(MomoStaker), 299(MomoStaker), 638(MomoStaker)	ⓘ Pending

Description

MoMoStaker.setRewardMgr(address) (MomoStaker.sol#131-141) ignores return value by _moboxToken.approve(rewardMgr,0) (MomoStaker.sol#134) MoMoStaker.setRewardMgr(address) (MomoStaker.sol#131-141) ignores return value by _moboxToken.approve(rewardMgr,uint256(- 1)) (MomoStaker.sol#140) MoMoStaker.getDevTeamReward() (MomoStaker.sol#293-300) ignores return value by _moboxToken.transfer(devTeam,amount) (MomoStaker.sol#299) MoMoStaker.getReward() (MomoStaker.sol#631-641) ignores return value by _moboxToken.transfer(msg.sender,userReward) (MomoStaker.sol#638)

Recommendation

Ensure that all the return values of the function calls are used.

MSM-10 | 3rd Party Dependencies

Category	Severity	Location	Status
Control Flow	● Minor	MomoStaker.sol: 604, 409, 587	ⓘ Pending

Description

409 (prototype, hashrate) = *momoToken.getMomoSimpleByTokenId(tokenIds[i]);* 587 (ids, vals, tokenIds) = *momoMinter.mintByStaker(msg.sender, amount_);* 604 *momoToken.levelUp(tokenId, protosV1V2V3_, amountsV1V2V3_, tokensV4V5_);*

"hashrate", mint and levelUp depends on 3rd party entities.

Recommendation

We encourage the team to constantly monitor the status of those 3rd parties to mitigate the side effects when unexpected activities are observed.

MSM-11 | Missing Zero Address Validation

Category	Severity	Location	Status
Control Flow	● Minor	MomoStaker.sol: 162~167(MomoStaker)	ⓘ Pending

Description

MoMoStaker.setMoMoStakerAuction(address).addr_ (MomoStaker.sol#162) lacks a zero-check on : - stakerAuction = addr_ (MomoStaker.sol#163) MoMoStaker.setDevTeam(address).addr_ (MomoStaker.sol#166) lacks a zero-check on : - devTeam = addr_ (MomoStaker.sol#167)

Recommendation

Check that the address is not zero.

MSM-12 | Calls Inside A Loop

Category	Severity	Location	Status
Logical Issue	● Minor	MomoStaker.sol: 362~438(MomoStaker), 442~530(MomoStaker)	ⓘ Pending

Description

Calls inside a loop might lead to a denial-of-service attack.

MoMoStaker.stakeNft(address,uint256[],uint256[],uint256[],bool,uint256) (MomoStaker.sol#362-438) has external calls inside a loop: momoToken.transferFrom(user,address(this),tokenIds[i]) (MomoStaker.sol#407)

MoMoStaker._stakeNft(address,uint256[],uint256[],uint256[],bool,uint256) (MomoStaker.sol#362-438) has external calls inside a loop: (prototype,hashrate) = momoToken.getMomoSimpleByTokenId(tokenIds[i]) (MomoStaker.sol#409)

MoMoStaker._removeNft(address,uint256[],uint256[],uint256[],uint256,uint256) (MomoStaker.sol#442-530) has external calls inside a loop: (prototype,shareParams[1]) = momoToken.getMomoSimpleByTokenId(tokenIds[i]) (MomoStaker.sol#487)

MoMoStaker.removeNft(address,uint256[],uint256[],uint256[],uint256,uint256) (MomoStaker.sol#442-530) has external calls inside a loop: momoToken.transferFrom(address(this),user,tokenIds[i]) (MomoStaker.sol#505)

Recommendation

Favor pull over push strategy for external calls.

MSM-13 | Public Function Could Be Declared External

Category	Severity	Location	Status
Gas Optimization	● Minor	MomoStaker.sol: 51~59(MomoStaker)	ⓘ Pending

Description

transferOwnership(address) should be declared external: - Ownable.transferOwnership(address)
(comm/Ownable.sol#51-59)

`public` functions that are never called by the contract should be declared external to save gas.

Recommendation

Use the external attribute for functions never called from the contract.

MSP-01 | Integer Overflow

Category	Severity	Location	Status
Mathematical Operations	● Minor	MoboxStrategyP.sol: 266, 242, 252, 313	🕒 Resolved

Description

Although integer overflows would not happen if some variables such as now are within regular ranges, SafeMath is still highly recommended for mathematical operations, if gas costs are not considered as a most significant factor in implementations, to prevent exceptions.

Recommendation

We recommend applying SafeMath.add at the aforementioned line

MSP-02 | Missing Checks for Reentrancy

Category	Severity	Location	Status
Logical Issue	● Major	MoboxStrategyP.sol: 266	☑ Resolved

Description

Function harvest() have state updates or event emits after external calls and thus are vulnerable to reentrancy attack.

Recommendation

Applying nonReentrant modifier.

MSP-03 | Missing Zero Address Validation

Category	Severity	Location	Status
Logical Issue	● Minor	MoboxStrategyP.sol: 319	☑ Resolved

Description

lacks a zero-check on :

- strategist = strategist_ (MoboxStrategyP.sol#319)

Recommendation

Adding check that strategist_ is not zero address.

MSP-04 | Missing slippage protection

Category	Severity	Location	Status
Logical Issue	● Minor	MoboxStrategyP.sol: 239, 249, 310	ⓘ Acknowledged

Description

Missing slippage protection in all `swapExactTokensForTokensSupportingFeeOnTransferTokens()` functions.

Recommendation

There is well-known sandwich attacks in uniswap, we recommend always set slippage protection in similar functions.

MSP-05 | Wrong Withdraw Amount

Category	Severity	Location	Status
Logical Issue	● Major	MoboxStrategyP.sol: 165~171	🕒 Resolved

Description

```
165     uint256 lpBalance = IERC20(wantToken).balanceOf(address(this));
166     if (lpBalance < amount_) {
169     }
170
171     uint256 wantAmount = lpBalance;
```

If `lpBalance > amount_`, then `wantAmount` still set to `lpBalance`.

Recommendation

Confirm this is intended, or set `wantAmount` to `amount_`.

MSP-06 | Implicitly Return Values

Category	Severity	Location	Status
Coding Style	● Informational	MoboxStrategyP.sol: 198~199, 129	ⓘ Acknowledged

Description

Functions defined with return values but values are returned implicitly.

Recommendation

We recommend always return values explicitly.

MSP-07 | Unimpletation Constructor Function

Category	Severity	Location	Status
Volatile Code	● Informational	MoboxStrategyP.sol: 84~86	🕒 Resolved

Description

```
constructor() public Unimpletation constructor function
```

Recommendation

Impletation constructor function,should delete if not needed

MSP-08 | 3rd Party Dependencies

Category	Severity	Location	Status
Control Flow	● Minor	MoboxStrategyP.sol	ⓘ Acknowledged

Description

MoboxStrategyP and MoboxStrategyV are serving as the underlying entity to interact with third party cake and venus protocols. The scope of the audit would treat those 3rd party entities as black boxes and assume its functional correctness. However in the real world, 3rd parties may be compromised that led to assets lost or stolen.

Recommendation

We understand that the business logic of Mobox requires the interaction with cake and venus for the sake of pursuing capital gains of its users. We encourage the team to constantly monitor the status of those 3rd parties to mitigate the side effects when unexpected activities are observed.

MSV-01 | Integer Overflow

Category	Severity	Location	Status
Mathematical Operations	● Minor	MoboxStrategyV.sol: 407, 447	☑ Resolved

Description

Although integer overflows would not happen if some variables such as now are within regular ranges, SafeMath is still highly recommended for mathematical operations, if gas costs are not considered as a most significant factor in implementations, to prevent exceptions.

Recommendation

We recommend applying SafeMath.add at the aforementioned line

MSV-02 | Missing slippage protection

Category	Severity	Location	Status
Logical Issue	● Minor	MoboxStrategyV.sol: 404, 444	ⓘ Acknowledged

Description

Missing slippage protection in all `swapExactTokensForTokensSupportingFeeOnTransferTokens()` functions.

Recommendation

There is well-known sandwich attacks in uniswap, we recommend always set slippage protection in similar functions.

MSV-03 | Compile Error

Category	Severity	Location	Status
Compiler Error	● Critical	MoboxStrategyV.sol: 450~451	☑ Resolved

Description

Error: Expected '(' but got identifier --> MoboxStrategyV.sol:452:14:

Recommendation

Fix the compile error by moving the "}" to an new line.

MSV-04 | Missing Checks for Reentrancy

Category	Severity	Location	Status
Logical Issue	● Major	MoboxStrategyV.sol: 329~330, 282, 248, 248, 316, 321	✓ Resolved

Description

Many functions have state updates or event emits after external calls and thus are vulnerable to reentrancy attack.

Recommendation

We recommend applying OpenZeppelin ReentrancyGuard library - nonReentrant modifier for the aforementioned functions to prevent reentrancy attack. We recommend applying OpenZeppelin ReentrancyGuard library - nonReentrant modifier for the aforementioned functions to prevent reentrancy attack.

MSV-05 | Missing Access Control

Category	Severity	Location	Status
Logical Issue	● Major	MoboxStrategyV.sol: 103~104	🕒 Resolved

Description

It's unsafe that init function can be called by anyone.

Recommendation

Add onlyOwner modifier.

MSV-06 | Implicitly Return Values

Category	Severity	Location	Status
Coding Style	● Informational	MoboxStrategyV.sol: 161, 360	ⓘ Acknowledged

Description

Functions defined with return values but values are returned implicitly.

Recommendation

We recommend always return values explicitly.

MSV-07 | Ignored Return Values

Category	Severity	Location	Status
Logical Issue	● Major	MoboxStrategyV.sol: 133~149	ⓘ Acknowledged

Description

Ignored return values may cause unexpected risks. For example, if `repayBorrow()` failed then leverage can't decreased while no place aware this failure.

Recommendation

Check every function return values.

MSV-08 | Leverage risk

Category	Severity	Location	Status
Logical Issue	● Informational	MoboxStrategyV.sol: 268, 118	☑ Resolved

Description

This strategy uses leverage, which may introduce potential risk. E.g If the Strategist doesn't deleverage in time, the system may enter liquidity crisis in an extreme market.

Recommendation

Don't use leverage.

MSV-09 | 3rd Party Dependencies

Category	Severity	Location	Status
Control Flow	● Minor	MoboxStrategyV.sol	ⓘ Acknowledged

Description

MoboxStrategyP and MoboxStrategyV are serving as the underlying entity to interact with third party cake and venus protocols. The scope of the audit would treat those 3rd party entities as black boxes and assume its functional correctness. However in the real world, 3rd parties may be compromised that led to assets lost or stolen.

Recommendation

We understand that the business logic of Mobox requires the interaction with cake and venus for the sake of pursuing capital gains of its users. We encourage the team to constantly monitor the status of those 3rd parties to mitigate the side effects when unexpected activities are observed.

MTM-01 | Mint to `_dest` address instead of mining pool

Category	Severity	Location	Status
Logical Issue	● Minor	MoboxToken.sol: 85	ⓘ Pending

Description

The comment of `mint()` said "Distribute MBOX to the main mining pool", but the code mints to `_dest` "`mint(dest, amountThisYear)`"

Recommendation

We recommend to confirm whether this is intended.

Appendix

Finding Categories

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Mathematical Operations

Mathematical Operation findings relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how `block.timestamp` works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Data Flow

Data Flow findings describe faults in the way data is handled at rest and in memory, such as the result of a struct assignment operation affecting an in-memory struct rather than an in-storage one.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of `private` or `delete`.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

Magic Numbers

Magic Number findings refer to numeric literals that are expressed in the codebase in their raw format and should otherwise be specified as constant contract variables aiding in their legibility and maintainability.

Compiler Error

Compiler Error findings refer to an error in the structure of the code that renders it impossible to compile using the specified version of the project.

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 CertiK's 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 CertiK 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.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.

