Unblock Labs

An EatTheBlocks Company

Audit report

Darkside - FarmerLandNFT & Wheat Token

Table of Contents

Summary

This report has been prepared by Unblock Labs for Darkside.Finance to discover issues and vulnerabilities in the source code of their FarmerlandNFT and WHEAT smart contracts as well as any contract dependencies used in the project. A comprehensive examination has been performed utilizing Static Analysis and Manual Code 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;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project summary

Project name	Darkside.finance - FarmerlandNFT / WHEAT Token
Platform	Polygon
Language	Solidity
Codebase	

Audit summary

Delivery date	December 2, 2022
Methodology	Static Analysis, Manual Review

Vulnerability summary

Level	Total	Acknowledge	Mitigated	Resolved
Critical	9	0	0	9
High	13	3	0	10
Medium	16	2	0	14
Low	25	2	0	23
Information	6	1	1	4
Discussion	0	0	0	0

Audit scope

ID	Contract	SHA256 checksum
FLN	FarmerLandNFT.sol	a799bcb188f661807eec05d4bb89a6fd06f44419837d0934fc6ca139e514f295
FLU	FarmerLandNFTLevelUpper.sol	8a40d6c3af557dd9bada43d4b0ac31dae25f4fc31b7a34142f7ee348e522bdd9
MAS	MasterChef.sol	79323267664931d4ead83900fbd9e52557e33658a11cbe5ce8cb27b9bc1d5dac
WHT	WHEAT.sol	e3c4abfff106eab106d45bca5c26229b6bfd19503c52fee2732451b0cfa1a5f4

Revised deployed code

ID	Contract	Polygon address
FLN	FarmerLandNFT.sol	0xc8D5460275eB20E26c524c8590ccb63748A37D15 0xC23DC736B6cC013904A8080C4101479f45076BC7 0xbB9f78D9396E5b22F01F76F62f0775D573be304A
FLU	FarmerLandNFTLevelUpper.sol	0x14A324B6355689C563D3031D3245346858A58025
MAS	MasterChef.sol	0x9982eF79551c21Ca7cC3E5Ff49050d43ef047C88
WHT	WHEAT.sol	0x5254463c5adBEDc54373d32fa0bF7545Ea62CA41

Findings

ID	Title	Category	Severity	Status
FLN-01	Contract not deployable	Volatile Code	Critical	Resolved
FLN-02	Invalid constant value	Volatile Code	Critical	Resolved
WHT-01	USDC funds can be fully withdrawn	Volatile Code	Critical	Resolved
WHT-02	Underflow when LAUNCH_TIME > block.timestamp	Volatile Code	Critical	Resolved
WHT-03	Division by 0 exception	Volatile Code	Critical	Resolved
WHT-04	Wrong management of reentrancy	Volatile Code	Critical	Resolved
WHT-05	Potential revert in ExitLobby()	Volatile Code	Critical	Resolved
WHT-06	Revert in _sendPartnersShare	Volatile Code	Critical	Resolved
WHT-07	Unbound value for lastLobbyPool	Volatile Code	Critical	Resolved
FLN-03	No restriction on mint quantity	Volatile Code	High	Resolved
FLN-04	Random NFT Ids affectation	Gas optimisation	High	Resolved
FLU-01	levelUpNFT will revert if the NFT is not staked in MasterChef	Volatile code	High	Resolved

ID	Title	Category	Severity	Status
FLU-02	setAbility will revert when MAX_ABILITY is reached	Volatile code	High	Acknowledge
FLU-03	levelUpNFT can be called by anybody	Volatile code	High	Acknowledge
MAS-01	Potential lost of NFT	Volatile code	High	Resolved
MAS-02	No upper distribution timeframe	Volatile code	High	Resolved
MAS-03	updateAbilityForDeposit can be called by anybody	Volatile code	High	Acknowledge
WHT-08	Unbound value for dividendsPoolCapDays	Volatile Code	High	Resolved
WHT-09	Potential revert in _sendShares	Volatile Code	High	Resolved
WHT-10	Rounding errors	Language specific	High	Resolved
WHT-11	lottery_topBuyer_today is not reseted	Volatile Code	High	Resolved
WHT-12	Invalid tracking of lottery_topBuy_today	Volatile Code	High	Resolved
GLB-01	Centralization related risks	Centralization / Privilege	Medium	Acknowledge
FLN-05	Duplicate functionality	Gas optimisation	Medium	Resolved
FLN-06	No events emitted on state change	Language Specific	Medium	Resolved
FLN-07	Use of weak random number generation	Language Specific	Medium	Resolved
FLN-08	Mint can be stopped	Volatile code	Medium	Resolved

ID	Title	Category	Severity	Status
FLN-09	Centralisation privilege	Volatile code	Medium	Acknowledge
FLU-04	recoverTokens is used to withdraw tokens received from fees	Volatile code	Medium	Resolved
FLU-05	startTime can be changed after started	Volatile code	Medium	Resolved
FLU-06	No events emitted on state change	Volatile code	Medium	Resolved
MAS-04	use safeTransferFrom during ERC721 transfers	Volatile code	Medium	Resolved
MAS-05	Check Effects Interactions pattern violation	Volatile code	Medium	Resolved
WHT-13	Check Effects Interactions pattern violation	Volatile Code	Medium	Resolved
WHT-14	daoAddress can be changed	Volatile Code	Medium	Resolved
WHT-15	Missing address(0) validation	Volatile Code	Medium	Resolved
WHT-16	No events emitted on state change	Volatile Code	Medium	Resolved
WHT-17	percentOfLobbyToBePooled is initialised from non-initialised variables	Language specific	Medium	Resolved
FLN-10	hasUserRoostedAny does not keep history	Coding style	Low	Resolved
FLN-11	Potential high gas usage in walletOfOwner	Gas optimisation	Low	Resolved
FLN-12	More than 1 NFT can be minted for free	Volatile code	Low	Acknowledge

ID	Title	Category	Severity	Status
FLN-13	Unused property	Gas optimisation	Low	Resolved
FLN-14	SafeMath is not required with solc >= 0.8	Gas optimisation	Low	Resolved
FLN-15	Use of ERC721Enumerable	Gas optimisation	Low	Acknowledge
FLN-16	Functions could be external	Gas optimisation	Low	Resolved
FLN-17	MAX_ELEMENTS should be constants	Coding style	Low	Resolved
MAS-06	Gas optimisation in updatePool()	Gas optimisation	Low	Resolved
MAS-07	Potential invalid test in set_MAX_NFT_COUNT	Code volatility	Low	Resolved
MAS-08	Variable name shadows a state variable	Language specific	Low	Resolved
MAS-09	Duplicate variable	Gas optimisation	Low	Resolved
MAS-10	Unused variables	Gas optimisation	Low	Resolved
WHT-18	Missing validation in EnterStake	Volatile Code	Low	Resolved
WHT-19	Possible underflow in getLoanOnStake	Volatile Code	Low	Resolved
WHT-20	Possible error in _clcNFTBoost	Volatile Code	Low	Resolved
WHT-21	Parameter nftType not required in setUserNFTRoostings	Gas optimisation	Low	Resolved
WHT-22	Unused variable	Gas optimisation	Low	Resolved
WHT-23	Multiple calls to _clcDay()	Gas optimisation	Low	Resolved

ID	Title	Category	Severity	Status
WHT-24	Properties set but not used	Gas optimisation	Low	Resolved
WHT-25	No need to send block.timestamp in events	Gas optimisation	Low	Resolved
WHT-26	token_USDC should be immutable	Gas optimisation	Low	Resolved
WHT-27	stakeCount should be stored in a mapping	Gas optimisation	Low	Resolved
WHT-28	Mutualise code when possible	Coding style	Low	Resolved
FLN-18	Invalid error message	Coding style	Information	Resolved
FLN-19	Transfer of NFTs are locked while roosting	Coding style	Information	Resolved
WHT-29	UserLobby event should emit the referrer address	Coding style	Information	Resolved
WHT-30	_updateDaily and _clcTokenValue should be in mixedCase	Coding style	Information	Resolved
WHT-31	Potential High fees and token mint	Coding style	Information	Mitigated
GLB-02	Coding pattern	Coding style	Information	Acknowledge

FLN-01 | Contract not deployable

Category	Severity	Location	Status
Volatile Code	🔴 Critical	FarmerlandNFT.sol: 77~94	Resolved

Description

In the constructor of the contracts, the variable `remainingIds` is initialized with `2,000` elements pushed in the array, thus preventing the contract from being deployed because of the high amount of gas used.

Recommendation

This initialization should not be required (see FLN-04), but if it is, we recommend moving the initialization code to a separate function.
This function should take a maxIteration parameter as argument to limit the gas used per transaction.

Alleviation

`[UnblockLabs]`: The client opted to make the recommended changes and removed `remainingIds.`

FLN-02 | Invalid constant value

Category	Severity	Location	Status
Volatile Code	🔴 Critical	FarmerlandNFT.sol: 44	Resolved

Description

The constant `MAX_ABILITY` is set to **100** but in `_mintAnElement` the default ability is set between **10,000** and **40,000**. This will make the `setAbility` function revert when trying to update the ability of a token.

Recommendation

`[UnblockLabs]`: The client opted to make the recommended changes and the MAX_ABILITY constant was set to **1,000,000**.

FLN-03 | No restriction on mint quantity

Category	Severity	Location	Status
Volatile Code	High	FarmerlandNFT.sol: 130~152	Resolved

Description

In the current implementation of `mint()` users can potentially mint as much NFT as they want. An attacker could use this to mint all the NFT of the project.

Recommendation

Add a max quantity minted per transaction or per wallet.

Alleviation

`[UnblockLabs]`: The client opted to make the recommended changes and implemented a limit of **50** mint per transaction when not executed by an owner account.

FLN-04 | Random NFT Ids affectation

Category	Severity	Location	Status
Gas optimisation	High	FarmerlandNFT.sol: 190~208	Resolved

Description

The logic to get a random Id for the next token Id should be simplified and optimized to use less gas and not rely on a pre-initialized list.

Recommendation

Change the logic to use a simpler and cheaper implementation.

Alleviation

`[UnblockLabs]`: The client opted to make the recommended changes.

FLN-05 | Duplicate functionality

Category	Severity	Location	Status
Gas optimisation	Medium	FarmerlandNFT.sol: 21	Resolved

Description

The `_tokenIdTracker` duplicates the functionality already present in the base class `ERC721Enumerable`

Recommendation

Remove the `_tokenIdTracker` variable and use `totalSupply()` from base class.

Alleviation

`[UnblockLabs]`: The client opted to make the recommended changes.

FLN-06 | No events emitted on state change

Category	Severity	Location	Status
Language Specific	Medium	FarmerlandNFT.sol: 108~112, 118~122, 203~205, 206~201, 259~263	Resolved

Description

The following functions do not emit events to pass the changes out of chain.

- `setAbility`
- `setLevel`
- `setBaseURI`
- `withdrawAll`
- `addToWhiteList`

Recommendation

We recommend declaring and emitting corresponding events for all the essential state variables that are possible to be changed during runtime.

Alleviation

`[UnblockLabs]`: The client opted to make the recommended changes.

FLN-07 | Use of weak random number generation

Category	Severity	Location	Status
Language Specific	Medium	FarmerlandNFT.sol: 190	Resolved

Description

The random generation is based on `block.timestamp`, and a fixed `entropy` (seed) passed in the constructor. This seed does not add any randomness to the number generated.

We do not recommend relying solely on `block.timestamp` to generate a random number since all the calls in a block will return the same value.

Recommendation

If a true random number generation is required, we recommend using an external service like ChainLink VRF.

If pseudo random numbers can be sufficient, we recommend incrementing a `counter` and using more variables like `msg.sender` or `block.difficulty` to ensure the number returned is always different.

Alleviation

`[UnblockLabs]`: The client adapted the algorithm to use more information to generate the random number.

`[Darkside.finance]`: We are aware of the limitation of the generation of a random number on-chain, but a true random number is not required in our case, the pseudo random number is sufficient.

FLN-08 | Mint can be stopped

Category	Severity	Location	Status
Volatile code	Medium	FarmerlandNFT.sol: 274~278	Resolved

Description

`setStartTime` does not validate that the mint is already started and does not validate the date set, thus enabling the owner to stop the mint by setting a new `startTime` in the future.

Recommendation

Add a verification to check that the mint is not started before changing `startTime` and validate that the date is not in the past.
Implement a specific “pause” function if this functionality is required.

Alleviation

`[UnblockLabs]`: The client opted to make the recommended changes.

FLN-09 | Centralisation privilege

Category	Severity	Location	Status
Volatile code	Medium	FarmerlandNFT.sol: 108~112, 118~122;	Acknowledge

Description

The functions `setAbility` and `setLevel` can be called by the owner to advantage or disadvantage any token by setting any value.

```
function setAbility(uint tokenId, uint _ability) external {
  require(admins[msg.sender], "sender not admin!");
  require(_ability <= MAX_ABILITY, "sender not admin!");
  ability[tokenId] = _ability;
}

function setLevel(uint tokenId, uint _level) external{
  require(admins[msg.sender], "sender not admin!");
  require(_level <= MAX_LEVEL, "sender not admin!");
  level[tokenId] = _level;
}
```

Recommendation

Restrict access to `FarmerlandNFTLevelUpper` contract only.

Alleviation

`[UnblockLabs]`: The client opted to keep the implementation as-is.

`[Darkside.finance]`: The owner may have to update the information for a specific NFT so this feature is needed for the proper functioning of the project.

FLN-10 | hasUserRoostedAny does not keep history

Category	Severity	Location	Status
Coding style	🟡 Low	FarmerlandNFT.sol: 100~102	Resolved

Description

The function `hasUserRoostedAny` returns a boolean indicating if `userRoostingsCount` is greater than `0`.
When an NFT is “unroosted”, the counter `userRoostingsCount` is decremented thus setting the value back to false when no NFT are currently roosting.

Recommendation

Create a variable `mapping(address => boolean)` that is set to `true` the first time a user calls `roostNftId`.

If the history is not required, remove this function as it does not add any value over `getUsersNumberOfRoostings.`

Alleviation

`[UnblockLabs]`: The client opted to remove the function.

FLN-11 | Potential high gas usage in walletOfOwner

Category	Severity	Location	Status
Gas optimisation	Low	FarmerlandNFT.sol: 211~220	Resolved

Description

In the current implementation of `WalletOfOwner`, a “for” loop is used to list all the tokens of a user without restrictions on the max quantity.
If the user has many tokens, this function can fail by consuming too much gas.

Recommendation

`walletOfOwner` should take a `startIndex` and `count` parameters so the caller can be responsible for the max gas used.

Alleviation

`[UnblockLabs]`: The client opted to make the recommended changes.

FLN-12 | More than 1 NFT can be minted for free

Category	Severity	Location	Status
Volatile code	Low	FarmerlandNFT.sol: 259~263	Acknowledge

Description

A call to `addToWhiteList` from the owner account can enable an address to mint for free multiple times.

Recommendation

Keep track of addresses that already minted for free.

Alleviation

`[UnblockLabs]`: The client opted to keep the implementation as-is.

FLN-13 | Unused property

Category	Severity	Location	Status
Gas optimisation	Low	FarmerlandNFT.sol: 40	Resolved

Description

The property `masterChef` is set but never used within the contract.

Recommendation

Remove this property if not required.

Alleviation

`[UnblockLabs]`: The client opted to make the recommended changes.

FLN-14 | SafeMath is not required with solc >= 0.8

Category	Severity	Location	Status
Gas optimisation	🟡 Low	FarmerlandNFT.sol: 4	Resolved

Description

`SafeMath` is not required with `solc >= 0.8` as the compiler already checks for over and underflows.

Recommendation

Remove `SafeMath` to save gas.

Alleviation

`[UnblockLabs]`: The client opted to make the recommended changes.

FLN-15 | Use of ERC721Enumerable

Category	Severity	Location	Status
Gas optimisation	🟡 Low	FarmerlandNFT.sol	Acknowledge

Description

The contract `FamerlandNFT` inherits from `ERC721Enumerable`.
This contract is quite gas consuming and most of its functionalities can be recreated off chain by listening to events emitted by the contract.

Recommendation

If the functionalities provided by `ERC721Enumerable` are not used on chain, we recommend using the “classic” implementation of `ERC721.`

Alleviation

`[UnblockLabs]`: The client opted to keep the implementation as-is.

`[Darkside.finance]`: The features provided by `ERC721Enumerable` are useful and needed for the project.

FLN-16 | Functions could be external

Category	Severity	Location	Status
Gas optimisation	🟡 Low	FarmerlandNFT.sol: 130, 203;	Resolved

Description

The function `mint` and `setBaseURI` are not directly used in the contract, they can be declared `external`.

Recommendation

Change the functions visibility to `external.`

Alleviation

`[UnblockLabs]`: The client opted to make the recommended changes.

FLN-17 | MAX_ELEMENTS should be constants

Category	Severity	Location	Status
Coding style	Low	FarmerlandNFT.sol: 23;	Resolved

Description

The `MAX_ELEMENTS` property is never changed within the implementation of the contract and should be declared as a constant.

Recommendation

Change the property to `constant.`

Alleviation

`[UnblockLabs]`: The client opted to make the recommended changes.

FLN-18 | Invalid error message

Category	Severity	Location	Status
Coding style	Information	FarmerlandNFT.sol: 125; 139	Resolved

Description

The error message in `setAbility` and `setLevel` does not correspond to the actual error.

```
require(_ability <= MAX_ABILITY, "sender not admin!");
```

```
require(_level <= MAX_LEVEL, "sender not admin!");
```

Recommendation

Update the message to match the error.

Alleviation

`[UnblockLabs]`: The client opted to make the recommended changes.

FLN-19 | Transfer of NFTs are locked while roosting

Category	Severity	Location	Status
Coding style	Information	FarmerlandNFT.sol: 227	Resolved

Description

The `_transfer` function prevents NFTs from being transferred while roosting. This can have side effects if the NFTs are listed on external marketplaces preventing the transfer while still being listed by the owner.

Recommendation

Consider transferring the NFTs to the smart contract during the roosting period.

Alleviation

`[UnblockLabs]`: The client opted to adapt the code to trigger an “unlock” of the NFT it is transferred while roosting.

FLU-01 | levelUpNFT will revert if the NFT is not staked in MasterChef

Category	Severity	Location	Status
Volatile code	High	FarmerlandNFTLevelUpper.sol: 171	Resolved

Description

In `levelUpNFT`, the call to `MasterChef` will revert if the token is not staked:

```
nftMasterChef.updateAbilityForDeposit(msg.sender, series, tokenId);
```

MasterChef.sol: 160

```
require(userStakedMap[_user][_series][_tokenId], "nft not staked by
specified user");
```

Recommendation

Add a condition to only call `MasterChef` when required.
If the tokens must be staked, add a verification in `levelUpNFT` before setting the states.

Alleviation

`[UnblockLabs]`: The client opted to make the recommended change, MasterChef is called only when the token is staked.

FLU-02 | setAbility will revert when MAX_ABILITY is reached

Category	Severity	Location	Status
Volatile code	High	FarmerlandNFTLevelUpper.sol: 168	Acknowledge

Description

In `levelUpNFT`, the call to `setAbility` will revert when `MAX_ABILITY` is reached, though the value keeps being incremented.

```
FarmerLandNFT(series).setAbility(tokenId, oldAbility + (levelsToUp *
(baseBoostPerLevel * getLobbyVolumeScore(msg.sender))) / 1e4);
```

FarmerLandNFT.sol: 125

```
require(_ability <= MAX_ABILITY, "sender not admin!");
```

Recommendation

Stop incrementing the ability once `MAX_ABILITY` is reached.

Alleviation

`[UnblockLabs]`: The client opted to keep the implementation as-is. After discussion with the client, we do not believe that it implies any security concerns to remain unchanged.

`[Darkside.finance]`: This behavior is expected and the function should revert when MAX_ABILITY is reached.

FLU-03 | levelUpNFT can be called by anybody

Category	Severity	Location	Status
Volatile code	High	FarmerlandNFTLevelUpper.sol: 123,174	Acknowledge

Description

Anybody can call the `levelUpNFT()` function even if they are not the token's owner.

Recommendation

Add a test to ensure `msg.sender` is the owner of the token.

Alleviation

`[UnblockLabs]`: The client opted to keep the implementation as-is.

`[Darkside.finance]`: This behavior is expected and will be kept since it does not have any negative impact for the user.

FLU-04 | recoverTokens is used to withdraw tokens received from fees

Category	Severity	Location	Status
Volatile code	Medium	FarmerlandNFTLevelUpper.sol: 183~188	Resolved

Description

The fees collected by `levelUpNFT` are withdrawn using the function `recoverTokens.`

Recommendation

Add a specific `withdraw` function to retrieve the tokens collected by the contract.

Alleviation

`[UnblockLabs]`: The client opted to make the recommended change.

FLU-05 | startTime can be changed after started

Category	Severity	Location	Status
Volatile code	Medium	FarmerlandNFTLevelUpper.sol: 176~180	Resolved

Description

The function `setStartTime` does not validate the data passed nor check that the `startTime` is already started.

Recommendation

Do not change `startTime` after starting.
Use the `levellingUpIsPaused` flag to pause the contract if required.

Alleviation

`[UnblockLabs]`: The client opted to make the recommended change.

FLU-06 | No events emitted on state change

Category	Severity	Location	Status
Volatile code	Medium	FarmerlandNFTLevelUpper.sol: 73~75,79~81,85~89, 91~95	Resolved

Description

The following functions do not emit events to pass the changes out of chain.

- `set_levellingUpIsPaused`
- `set_nftMasterChef`
- `set_usdcLobbyVolumeForMaxAbilityBoost`
- `set_baseBoostPerLevel`

Recommendation

We recommend declaring and emitting corresponding events for all the essential state variables that are possible to be changed during runtime.

Alleviation

`[UnblockLabs]`: The client opted to make the recommended changes.

MAS-01 | Potential lost of NFT

Category	Severity	Location	Status
Volatile code	🟠 High	Masterchef.sol: 158;	Resolved

Description

If a collection previously allowed is removed from `nftAddressAllowListSet`, the users won't be allowed to use the `emergencyWithdraw` function since this function uses the set to enumerate only the allowed collection.

```
for (uint i = 0;i<nftAddressAllowListSet.length();i++) {
  ...
}
```

Recommendation

Keep track of collections previously approved to always let the user withdraw his tokens.
An alternative could be to take the collection address and the tokenId as parameters and verify that the token was staked by the caller.

Alleviation

`[UnblockLabs]`: The client opted to make the recommended change.

MAS-02 | No upper distribution timeframe

Category	Severity	Location	Status
Volatile code	🟠 High	Masterchef.sol: 461~467, 473~479;	Resolved

Description

The functions `setUSDCDistributionTimeFrame` and `setWHEATDistributionTimeFrame` do not validate the upper range of the timeframe. A high value can prevent the rewards from being distributed.

Recommendation

Add a reasonable upper limit.

Alleviation

`[UnblockLabs]`: The client opted to make the recommended change.

`[Darkside.finance]`: An upper limit of 32 days was added.

MAS-03 | updateAbilityForDeposit can be called by anybody

Category	Severity	Location	Status
Volatile code	High	Masterchef.sol: 158	Acknowledge

Description

Anybody can call the `updateAbilityForDeposit()` function even if they are not the token's owner.

Recommendation

Add to test to ensure this function is called only by FarmerlandNFTLevelUpper contract.

Alleviation

`[UnblockLabs]`: The client opted to keep the implementation as-is.

`[Darkside.finance]`: This behavior is expected and will be kept since it does not have any negative impact for the user.

MAS-04 | use safeTransferFrom during ERC721 transfers

Category	Severity	Location	Status
Volatile code	Medium	Masterchef.sol: 203,236,269;	Resolved

Description

In the `deposit`, `withdraw` and `emergencyWithdraw` functions, the NFTs are transferred using the `transfer`/`transferFrom` functions.

Recommendation

We recommend using the `safeTransfer`/`safeTransferFrom` functions to validate the receiver.

Alleviation

`[UnblockLabs]`: The client opted to make the recommended change.

MAS-05 | Check Effects Interactions pattern violation

Category	Severity	Location	Status
Volatile code	Medium	Masterchef.sol: 222~252, 255~289, 445~455;	Resolved

Description

Inside the following functions, state variables are set after an external call

- `withdraw()`
- `transferUSDCToUser()`
- `transferUSDCToUser()`
- `transferWHEATToUser()`

Recommendation

State variables should be set before an external call:

change

```
IERC721(_series).transferFrom(address(this), msg.sender, _tokenId);
userStakeCounts[msg.sender]--;
...
```

to

```
userStakeCounts[msg.sender]--;
...
IERC721(_series).transferFrom(address(this), msg.sender, _tokenId);
```

Alleviation

`[UnblockLabs]`: The client opted to make the recommended changes.

MAS-06 | Gas optimisation in updatePool()

Category	Severity	Location	Status
Gas optimisation	Low	Masterchef.sol: 145~155	Resolved

Description

Inside the `updatePool` function, gas can be saved by setting the state only if `usdcRelease > 0` or `wheatRelease > 0`

Recommendation

Add a test to only update the states when changed.

Alleviation

`[UnblockLabs]`: The client opted to make the recommended changes.

MAS-07 | Potential invalid test in set_MAX_NFT_COUNT

Category	Severity	Location	Status
Code volatility	Low	Masterchef.sol: 352;353;	Resolved

Description

`MAX_NFT_COUNT` is set by default at 150 during initialisation but the function `set_MAX_NFT_COUNT` only allows a range between 21 and 149.

```
require(new_MAX_NFT_COUNT > 20, "MAX_NFT_COUNT must be greater than 0");
require(new_MAX_NFT_COUNT < 150, "MAX_NFT_COUNT must be less than 150");
```

Recommendation

Change the test to `>=` and `<=` if required.

Alleviation

`[UnblockLabs]`: The client opted to make the recommended change.

MAS-08 | Variable name shadows a state variable

Category	Severity	Location	Status
Language specific	Low	Masterchef.sol: 165;	Resolved

Description

The variable `userInfo` in `updateAbilityForDeposit` shadows a state variable with an identical name:

```
UserInfo storage userInfo = userInfo[_user];
```

Recommendation

Rename the variable within `updateAbilityForDeposit` implementation.

Alleviation

`[UnblockLabs]`: The client opted to make the recommended change.

MAS-09 | Duplicate variable

Category	Severity	Location	Status
Gas optimisation	Low	Masterchef.sol: 160~187	Resolved

Description

The variables `nftAddressAllowListMap` and `nftAddressAllowListSet` both store the same information.

Recommendation

Remove `nftAddressAllowListMap` and use `nftAddressAllowListSet.contains()` where needed.

Alleviation

`[UnblockLabs]`: The client opted to make the recommended change.

MAS-10 | Unused variables

Category	Severity	Location	Status
Gas optimisation	Low	Masterchef.sol: 101,103	Resolved

Description

The variables `totalAllocPoint` and `startTimestamp` are never used or set within the implementation of the contract.

Recommendation

Remove unused variables.

Alleviation

`[UnblockLabs]`: The client opted to make the recommended change.

WHT-01 | USDC funds can be fully withdrawn

Category	Severity	Location	Status
Volatile Code	🔴 Critical	WHEAT.sol: 267~269	Resolved

Description

In the function `flushLottyPool()`, the variable `lottery_Pool` is never reseted.

```
function flushLottyPool() external onlyOwner() nonReentrant {
   token_USDC.transfer(address(daoAddress), lottery_Pool);
}
```

This poses 2 major problems:

- Since the amount affected to `lottery_Pool` keeps being incremented, the next call to `flushLottyPool` will withdraw more than expected and will end up draining all the tokens.

- In case of a compromised owner's account private key, an attacker can use this method to withdraw all the USDC from the contract without restrictions. Since the `daoAddress` can be changed by the owner's account, USDC can be withdrawn to any address.

Recommendation

Update the function to reset the amount available to withdraw.
ie:

```
function flushLottyPool() external onlyOwner() nonReentrant {
  if (lottery_Pool > 0) {
    uint256 amount = lottery_Pool;
    lottery_Pool = 0;
    token_USDC.transfer(daoAddress, amount);
  }
}
```

Alleviation

`[UnblockLabs]`: The client opted to make the recommended change.

WHT-02 | Underflow when LAUNCH_TIME > block.timestamp

Category	Severity	Location	Status
Volatile Code	🔴 Critical	WHEAT.sol: 283	Resolved

Description

The function _clcDay will generate an underflow exception when LAUNCH_TIME is in the future, preventing the contract execution from working.

```
function _clcDay() public view returns (uint) {
   return (block.timestamp - LAUNCH_TIME) / 1 days;
}
```

Recommendation

Update the function to handle this case.
ie:

```
function _clcDay() public view returns (uint) {
   if (block.timestamp <= LAUNCH_TIME) return 0;
   return (block.timestamp - LAUNCH_TIME) / 1 days;
}
```

Alleviation

`[UnblockLabs]`: The client opted to make the recommended change.

WHT-03 | Division by 0 exception

Category	Severity	Location	Status
Volatile Code	Critical	WHEAT.sol: 290	Resolved

Description

In the function `_updateDaily()`, when `currentDay == 0` and `_clcDay() == 1`, the code will execute a division by 0, thus generating an exception and preventing the contract execution from working.
The contract will stay stuck in this state.

```
dayUSDCPool[_day] += (lobbyEntry[currentDay] * percentOfLobbyToBePooled)
/ (currentDay * 10000);
```

Recommendation

Update the function to handle this case.
ie:

```
for (uint _day = currentDay + 1; _day <= (currentDay * 2 + 1); _day++) {
  if (currentDay == 0) {
    dayUSDCPool[_day] = 0;
  }
  else {
    dayUSDCPool[_day] += (lobbyEntry[currentDay] *
percentOfLobbyToBePooled) / (currentDay * 10000);
  }
}
```

Alleviation

`[UnblockLabs]`: The client opted to make the recommended change.

WHT-04 | Wrong management of reentrancy

Category	Severity	Location	Status
Volatile Code	Critical	WHEAT.sol: 339; 359; 1134	Resolved

Description

The following internal functions are declared with a `nonReentrant` modifier:

- `_sendShares()`
- `_sendPartnersShare()`
- `checkLottery()`

Those functions are called within the function `_updateDaily()`, called by functions with the `nonRentrant` modifier:

- `EndStake()`
- `lendOnStake()`
- `collectLendReturn()`

This generates an exception as the flag `_status` is already set to `_ENTERED` in the `ReentrancyGuard` contract.

Recommendation

Remove the `nonRentrant` modifier on internal functions.

Alleviation

`[UnblockLabs]`: The client opted to make the recommended change.

WHT-05 | Potential revert in ExitLobby()

Category	Severity	Location	Status
Volatile Code	Critical	WHEAT.sol: 549,550;	Resolved

Description

The function `ExitLobby()` will revert if any of `nftMasterChefAddress` or `daoAddress` are set to `address(0)`. The mint of tokens to `address(0)` will fail, thus preventing the users from withdrawing their tokens.

Since both `nftMasterChefAddress` and `daoAddress` can be changed by the owner without restriction, the case must be handled accordingly.

Recommendation

Add a test to only mint tokens if addresses are set and shares greater than 0.
ie

```
if (nftMasterChefAddress != address(0) && exitLobbyWHEATAmount > 0 &&
masterchefWHEATShare > 0) {
  _mint(nftMasterChefAddress, (exitLobbyWHEATAmount *
masterchefWHEATShare) /10000);
}
if (daoAddress != address(0) && exitLobbyWHEATAmount > 0 && daoWHEATShare
> 0) {
  _mint(daoAddress, (exitLobbyWHEATAmount * daoWHEATShare) /10000);
}
```

Alleviation

`[UnblockLabs]`: The client opted to make the recommended change.

WHT-06 | Revert in _sendPartnersShare

Category	Severity	Location	Status
Volatile Code	🔴 Critical	WHEAT.sol: 327~330; 339~356;	Resolved

Description

The function `_sendPartnersShare` will revert if any of the partners addresses are set to `address(0):`

- `partner_1_addr`
- `partner_2_addr`
- `partner_3_addr`

In the function `partner_1_addr_set()`, the address of `partner_3_addr` is not correctly set, `partner_2_addr` is set instead, thus preventing the contract execution from working.

```
} else if (partner_id == 3) {
  partner_2_addr = addr;
  partner_2_share = share;
}
```

Recommendation

Update the function `partner_1_addr_set()` to set `partner_3_addr` correctly, and add a test in `_sendPartnersShare` to only send if the addresses are set.

Alleviation

`[UnblockLabs]`: The client opted to remove the partner shares.

WHT-07 | Unbound value for lastLobbyPool

Category	Severity	Location	Status
Volatile Code	Critical	WHEAT.sol: 543; 576~588;	Resolved

Description

When `lastLobbyPool` is equal to `0`, users won't be able to withdraw their tokens when calling `ExitLobby()`. The function `_clcTokenValue` will return `0`, and the call to `_mint()` will revert, thus preventing users from withdrawing their tokens.

```
_tokenValue = (lastLobbyPool *
mapMemberLobby[_address][_Day].entryAmount) / lobbyEntry[entryDay];
```

Since the owner has authority over the function `set_lastLobbyPool` an invalid value passed to the function can result in lost of tokens.

Recommendation

Validate that the new value is always greater than the previous one and never equals to 0.

```
function set_lastLobbyPool(uint lastLobbyPool_) external onlyOwner() {
  require(lastLobbyPool_ > lastLobbyPool, "Invalid value");
  lastLobbyPool = lastLobbyPool_;
}
```

Alleviation

`[UnblockLabs]`: The client opted to make the recommended change.

WHT-08 | Unbound value for dividendsPoolCapDays

Category	Severity	Location	Status
Volatile Code	High	WHEAT.sol: 543; 576~588;	Resolved

Description

When `dividendsPoolCapDays` is equal to `0`, `dayUSDCPool` will be equal to `0` and the function `calcStakeCollecting()` will return `0`, thus preventing users from getting profits from their stakings.

```
function set_dividendsPoolCapDays(uint_dividendsPoolCapDays) external
onlyOwner() {
  require(_dividendsPoolCapDays <= 300);
  dividendsPoolCapDays = _dividendsPoolCapDays;
}
```

```
userDivs += (dayUSDCPool[_day] * _stakeValue) /
totalTokensInActiveStake[_day];
```

Recommendation

Validate the lower bound of the value.
ie:

```
function set_dividendsPoolCapDays(uint dividendsPoolCapDays_) external
onlyOwner() {
  require(dividendsPoolCapDays_ > 0 && _dividendsPoolCapDays <= 300);
  dividendsPoolCapDays = dividendsPoolCapDays_;
}
```

Alleviation

`[UnblockLabs]`: The client opted to make the recommended change.

WHT-09 | Potential revert in _sendShares

Category	Severity	Location	Status
Volatile Code	High	WHEAT.sol: 359~371	Resolved

Description

The function `_sendShares()` will revert if any of the following addresses are set to `address(0):`

- `daoAddress`
- `nftMasterChefAddress`

Recommendation

Update the function to handle those cases correctly and execute the transfers only when the addresses are set and the amount greater than `0.`
ie:

```
if (lobbyEntry[currentDay - 1] > 0) {
  if (daoUSDCRawShare > 0) {
    uint daoUSDCRawShare = (lobbyEntry[currentDay - 1] * daoUSDCShare)
/10000;
    token_USDC.transfer(address(daoAddress), daoUSDCRawShare);
  }
  if (masterchefSDCRawShare > 0) {
    uint masterchefSDCRawShare = (lobbyEntry[currentDay - 1] *
masterchefUSDCShare) /10000;
    token_USDC.transfer(address(nftMasterChefAddress),
masterchefSDCRawShare);
  }
}
```

Alleviation

`[UnblockLabs]`: The client opted to make the recommended change.

WHT-10 | Rounding errors

Category	Severity	Location	Status
Language specific	High	WHEAT.sol: 826; 1214; 1332	Resolved

Description

In the functions `buyStakeRequest()`, `lendOnStake()` and `checkLottery()`, the code does not account for the rounding precision and the amounts will be wrong.

Recommendation

To keep the results correct, change the last multiplication to a subtraction.
ie change:

```
uint winnerAmount = (lottery_Pool * 30) /100;
token_USDC.transfer(address(lottery_topBuyer_today), winnerAmount);
lottery_Pool = (lottery_Pool * 70) /100;
```

to:

```
uint winnerAmount = lottery_Pool * 30 /100;
lottery_Pool -= winnerAmount;
token_USDC.transfer(lottery_topBuyer_today, winnerAmount);
```

Alleviation

`[UnblockLabs]`: The client opted to make the recommended change.

WHT-11 | lottery_topBuyer_today is not reseted

Category	Severity	Location	Status
Volatile Code	High	WHEAT.sol: 1134~1159	Resolved

Description

In the functions `checkLottery()`, the value of `lottery_topBuyer_today` is not reseted everyday. If no deposits are done during 1 day, the last winner will be selected to win again.

Recommendation

Set `lottery_topBuyer_today` to `address(0)` when resetting `lottery_topBuy_today.`

Alleviation

`[UnblockLabs]`: The client opted to make the recommended change.

Alleviation

`[UnblockLabs]`: The client opted to make the recommended change.

WHT-12 | Invalid tracking of lottery_topBuy_today

Category	Severity	Location	Status
Volatile Code	High	WHEAT.sol: 420~424	Resolved

Description

In the functions `DoEnterLobby()`, only the last amount entered is considered to select the top buyer though the user can enter the lobby multiple times in a day.

```
if (rawAmount >= lottery_topBuy_today) {
  // new top buyer
  lottery_topBuy_today = rawAmount;
  lottery_topBuyer_today = msg.sender;
}
```

Recommendation

Consider the total of the day for the user when checking the winner.

```
if (mapMemberLobby[msg.sender][currentDay].entryAmount >=
lottery_topBuy_today) {
  // new top buyer
  lottery_topBuy_today =
mapMemberLobby[msg.sender][currentDay].entryAmount;
  lottery_topBuyer_today = msg.sender;
}
```

Alleviation

`[UnblockLabs]`: The client opted to make the recommended change.

WHT-13 | Check Effects Interactions pattern violation

Category	Severity	Location	Status
Volatile Code	Medium	WHEAT.sol: 538; 633	Resolved

Description

In the function `EnterStake()`, the tokens of the user should be burnt before updating the state.

In the function `ExitLobby()`, the state should be updated before the mint.

Recommendation

Follow the Check Effects Interactions pattern to improve your code security.

Alleviation

`[UnblockLabs]`: The client opted to make the recommended change.

WHT-14 | daoAddress can be changed

Category	Severity	Location	Status
Volatile Code	Medium	WHEAT.sol: 70~72;	Resolved

Description

The owner account has privileges over the method `changeDaoAddress()`.
If the owner's account private key is compromised an attacker can change the address of the DAO and receive the USDC.
Since no events are emitted in `changeDaoAddress()`, the change is hard to track off chain.

Recommendation

Remove the function if not required, set the address to a constant value, and use a multi signature wallet for the owner account and DAO address.
If the function is required, send an event in case of change and use a service to monitor changes off chain.

Alleviation

`[UnblockLabs]`: The client opted to keep the daoAddress editable and to add an event to monitor changes.

`[Darkside.finance]`: The owner account will use a multi signature wallet.

WHT-15 | Missing address(0) validation

Category	Severity	Location	Status
Volatile Code	Medium	WHEAT.sol: 65~67, 70~72;	Resolved

Description

The functions `set_nftMasterChefAddress()` and `changeDaoAddress()` do not check address(0).
The contract execution will revert if those addresses are not set correctly.

Recommendation

Validate the parameters of the functions to be sure `address(0)` is never set.

Alleviation

`[UnblockLabs]`: The client opted to make the recommended change.

WHT-16 | No events emitted on state change

Category	Severity	Location	Status
Volatile Code	Medium	WHEAT.sol: 65~67, 70~72;	Resolved

Description

The following functions do not emit events to pass the changes out of chain:

- `set_nftMasterChefAddress`
- `changeDaoAddress`
- `set_lottery_share_percentage`
- `set_masterchefUSDCWHEATShare`
- `set_daoUSDCShare`
- `set_lastLobbyPool`
- `set_dividendsPoolCapDays`
- `switchVirtualBalanceEntering`
- `switchLoaningStatus`
- `switchVirtualBalanceEntering`
- `switchStakeSellingStatus`
- `flushLottyPool`
- `flushdevShareOfStakeSells`
- `ExitLobby`

Recommendation

We recommend declaring and emitting corresponding events for all the essential state variables that are possible to be changed during runtime.

Alleviation

`[UnblockLabs]`: The client opted to make the recommended change.

WHT-17 | percentOfLobbyToBePooled is initialized from non-initialised variables

Category	Severity	Location	Status
Language specific	Medium	WHEAT.sol: 131~132;	Resolved

Description

`percentOfLobbyToBePooled` is initialized from non-initialized variables.

```
uint public partner_1_share;
uint public partner_2_share;
uint public partner_3_share;
...
uint public percentOfLobbyToBePooled = 10000 - (partner_1_share +
partner_2_share + partner_3_share +
lottery_share_percentage + masterchefUSDCShare + daoUSDCShare);
```

Recommendation

Move the initialisation of the variable in the constructor of the contract.

Alleviation

`[UnblockLabs]`: The client opted to make the recommended change.

WHT-18 | Missing validation in EnterStake

Category	Severity	Location	Status
Volatile Code	Low	WHEAT.sol: 600	Resolved

Description

The function `EnterStake()` should validate that `amount` is greater than `0.`

Recommendation

Add a test to validate the parameters sent to the function.

Alleviation

`[UnblockLabs]`: The client opted to make the recommended change.

WHT-19 | Possible underflow in getLoanOnStake

Category	Severity	Location	Status
Volatile Code	Low	WHEAT.sol: 927	Resolved

Description

The function `getLoanOnStake()` can underflow if `endDay < loanDuration.`

```
require(mapMemberStake[msg.sender][stakeId].endDay - loanDuration > currentDay);
```

Recommendation

Change the test to never underflow.
ie:

```
require(mapMemberStake[msg.sender][stakeId].endDay > currentDay + loanDuration);
```

Alleviation

`[UnblockLabs]`: The client opted to make the recommended change.

WHT-20 | Possible error in _clcNFTBoost

Category	Severity	Location	Status
Volatile code	Low	WHEAT.sol	Resolved

Description

The description of the function `_clcNFTBoost()` states that the rewards are calculated as follow:

```
// _clcNFTBoost = amount * (1.1 + ability * 0.01)
```

Though, the implementation is:

```
return (amount * (1e12 * 1.1 + (((1e12 * ability) / 100) / 1e4))) / 1e12;
```

resulting in different returned values than the description.

Also, the implementation uses a float `1.1` which should be expressed as `110 / 100`

Recommendation

Make sure that the comment matches how the boost is calculated and do not use float in Solidity.

Alleviation

`[UnblockLabs]`: The client opted to keep the code as-is.

`[Darside.finance]`: The code has been verified and behaves as expected.

WHT-21 | Parameter nftType not required in setUserNFTRoostings

Category	Severity	Location	Status
Gas optimisation	Low	WHEAT.sol: 475	Resolved

Description

The parameter `nftType` is not required in `setUserNFTRoostings`.
The function already executes an external call to `FarmerLandNFT` to load and validate the value.

```
require(getNFTType(series) == nftType, "Bad nfttype");
```

Recommendation

Remove the parameter from the function and use the result from `getNFTType(series).`

Alleviation

`[UnblockLabs]`: The client opted to make the recommended change.

WHT-22 | Unused variable

Category	Severity	Location	Status
Gas optimisation	Low	WHEAT.sol: 235	Resolved

Description

The variable `daysActiveInStakeTokens` is never used or set within the implementation of the contract.

Recommendation

Remove unused variables.

Alleviation

`[UnblockLabs]`: The client opted to make the recommended change.

WHT-23 | Multiple calls to _clcDay()

Category	Severity	Location	Status
Gas optimisation	Low	WHEAT.sol: 235	Resolved

Description

The function `_updateDaily()` calls `_clcDay()` several times.

```
if (currentDay != _clcDay()) {...}
currentDay = _clcDay();
```

Recommendation

Add a local variable to store the result and reuse it within the function.

Alleviation

`[UnblockLabs]`: The client opted to make the recommended change.

WHT-24 | Properties set but not used

Category	Severity	Location	Status
Gas optimisation	Low	WHEAT.sol	Resolved

Description

The following properties only set data but their values are never read or used within the implementation of the contract:

- `mapMemberLobby_overallData`
- `daysActiveInStakeTokens`
- `daysActiveInStakeTokensIncrese`
- `daysActiveInStakeTokensDecrase`
- `totalStakeTradeAmount`

Recommendation

If the properties are only used for UI purposes, we recommend recreating those values off chain by indexing the contract events.

Alleviation

`[UnblockLabs]`: The client opted to make the recommended change and to remove the properties.

WHT-25 | No need to send block.timestamp in events

Category	Severity	Location	Status
Gas optimisation	Low	WHEAT.sol: 23~55	Resolved

Description

The following events send `block.timestamp` in their arguments.

```
event UserStake(address indexed addr, uint timestamp, uint rawAmount,
uint duration, uint stakeId);

event UserStakeCollect(address indexed addr, uinttimestamp, uint
rawAmount, uint stakeId, uintbonusAmount);

event UserLobby(address indexed addr, uint timestamp, uint rawAmount,
uint extraAmount);

event UserLobbyCollect(address indexed addr, uinttimestamp, uint
rawAmount, uint day, uintboostedAmount);

event StakeSellRequest(address indexed addr, uinttimestamp, uint price,
uint rawAmount, uint stakeId);;

event StakeLoanRequest(address indexed addr, uinttimestamp, uint
rawAmount, uint returnAmount, uintduration, uint stakeId);

event StakeLend(address indexed addr, uint lendId, address indexed
loaner, uint stakeId, uint amount, uint timestamp);

event DayLobbyEntry(uint timestamp, uint day, uintvalue);

event LotteryWinner(address indexed addr, uint amount, uint timestamp,
uint lastRecord);
```

Recommendation

The `block.timestamp` value is already present in the event through the block informations and can be safely removed.

Alleviation

`[UnblockLabs]`: The client opted to make the recommended change.

WHT-26 | token_USDC should be immutable

Category	Severity	Location	Status
Gas optimisation	Low	WHEAT.sol: 9	Resolved

Description

The value of `token_USDC` is never changed within the implementation of the contract, it can be safely marked as `immutable`.

Recommendation

Declare the property as `immutable.`

Alleviation

`[UnblockLabs]`: The client opted to make the recommended change.

WHT-27 | stakeCount should be stored in a mapping

Category	Severity	Location	Status
Gas optimisation	Low	WHEAT.sol: 642~650;	Resolved

Description

The function `calcStakeCount()` enumerates over all the staked `memberStake` to calculate the next `stakeId` using unnecessary gas during a call.

Recommendation

Keep a tracking of the user's stakeCount.
ie:

```
mapping(address => uint256) public stakeCounts;
```

Alleviation

`[UnblockLabs]`: The client opted to make the recommended change.

WHT-28 | Mutualise code when possible

Category	Severity	Location	Status
Coding style	Low	WHEAT.sol: 94; 103; 131; 332	Resolved

Description

The affectation of the variable percentOfLobbyToBePooled is repeated in different functions and should be mutualized to increase maintainability.

Recommendation

Create a specific function to update the value.
ie:

```
function _updatePercentOfLobbyToBePooled() private {
  percentOfLobbyToBePooled = 10000 - (partner_1_share + partner_2_share +
partner_3_share +
  lottery_share_percentage + masterchefUSDCShare + daoUSDCShare);
}
```

Alleviation

`[UnblockLabs]`: The client opted to make the recommended change.

WHT-29 | UserLobby event should emit the referrer address

Category	Severity	Location	Status
Coding style	Information	WHEAT.sol: 458	Resolved

Description

The `UserLobby` events does not send the referrer address making it harder to track infos off chain.

Recommendation

Add the referrer address to the event.

Alleviation

`[UnblockLabs]`: The client opted to make the recommended change.

WHT-30 | _updateDaily and _clcTokenValue should be in mixedCase

Category	Severity	Location	Status
Coding style	Information	WHEAT.sol: 285, 576;	Resolved

Description

To follow the naming conventions, properties and function names should use mixed casing.

Recommendation

The following pattern:

```
function _updateDaily() public { ... }

function _clcTokenValue(address _address, uint _Day) public view returns
(uint) { ... }
```

should be:

```
function updateDaily() public { ... }

function clcTokenValue(address address_, uint day_) public view returns
(uint) { ... }
```

Alleviation

`[UnblockLabs]`: The client opted to make the recommended change.

WHT-31 | Potential High fees and token mint

Category	Severity	Location	Status
Coding style	Information	WHEAT.sol	Mitigated

Description

The max fees withdrawn by the project can be set up to 50% of the lobby USDC entries (10% per partners, 10% for the DAO, and 10% for the lottery tax). An additional maximum 10% is redistributed to the masterchef contract.

The contract can also mint up to 20% of tokens on top of the rewarded token, distributed to the masterchef contract and the DAO, contributing to a fast depreciating value for the token.

Recommendation

Limit the taxes redistributed to the project.

Alleviation

`[UnblockLabs]`: The client opted to reduce the maximum fees.

`[Darkside.finance]`: We have removed partner share slots and limited the lottery to 2%.

GLB-01 | Centralization related risks

Category	Severity	Location	Status
Centralization / Privilege	Medium	FarmerLandNFT.sol FarmerlandNFTLevelUpper.sol MasterChef.sol WHEAT.sol	Acknowledge

Description

The owner has authority over many functions that can influence or stop the users from receiving their tokens.

Any compromise to the owner's private key account may allow an attacker to take advantage of this authority and mint new tokens, manipulate the parameters of the contracts, or block the withdrawals of staked tokens.
If a hacker takes control of this account, they can withdraw the majority of the staked funds

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign combination mitigate by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, mitigate by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement;
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered fully resolved.

- Renounce the ownership and never claim back the privileged roles;
  OR
- Remove the risky functionality.

Alleviation

`[UnblockLabs]`: The client acknowledges this point and will work to improve security and transparency around privilege actions.

GLB-02 | Coding pattern

Category	Severity	Location	Status
Coding style	Information	FarmerLandNFT.sol FarmerlandNFTLevelUpper.sol MasterChef.sol WHEAT.sol	Acknowledge

Description

To follow the naming conventions:

- Constant should be uppercase
- Properties and function names should use mixed casing
- Properties visibility should be explicit
- Properties should be declared before the constructor
- Functions should be declared after the constructor
- Properties and variables should be initialized
- do not use get_ / set_ as accessor
- do not compare to boolean constants
  (instead of `myVar == false`, use `!myVar`)

Alleviation

`[UnblockLabs]`: The client acknowledges this point.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

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.

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.

Volatile Code

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

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.

Checksum calculation method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

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 you ("Customer" or 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, nor may copies be delivered to any other person other than the Company, without Unblock Labs's prior written consent in each instance.

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 Unblock Labs 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 intended 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. Unblock Labs's position is that each company and individual are responsible for their own due diligence and continuous security. Unblock Labs'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.

The assessment services provided by Unblock Labs are subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is,

where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED “AS IS” AND “AS AVAILABLE” AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, UNBLOCK LABS HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, UNBLOCK LABS SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, UNBLOCK LABS MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER’S OR ANY OTHER PERSON’S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, UNBLOCK LABS PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER’S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER UNBLOCK LABS NOR ANY OF UNBLOCK LABS’S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. UNBLOCK LABS WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE

WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT UNBLOCK LABS'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST UNBLOCK LABS WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.
THE REPRESENTATIONS AND WARRANTIES OF UNBLOCK LABS CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST UNBLOCK LABS WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

Unblock Labs

An EatTheBlocks Company

www.unblock-labs.com