#HashEx
BLOCKCHAIN SECURITY

Mad Metaverse Polygon

smart contracts
final audit report

May 2022

hashex.org

contact@hashex.org

Contents

1. Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the disclaimer below – please make sure to read it in full.

By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and HashEx and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) (HashEx) owe no duty of care towards you or any other person, nor does HashEx make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties, or other terms of any kind except as set out in this disclaimer, and HashEx hereby excludes all representations, warranties, conditions, and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, HashEx hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against HashEx, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of the use of this report, and any reliance on this report. The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

2. Overview

HashEx was commissioned by the Mad Metaverse team to perform an audit of their smart contracts. The audit was conducted between 25/04/2022 and 13/05/2022.

The purpose of this audit was to achieve the following:

- Identify potential security issues with smart contracts
- Formally check the logic behind given smart contracts.

Information in this report should be used for understanding the risk exposure of smart contracts, and as a guide to improving the security posture of smart contracts by remediating the issues that were identified.

The code is available at the @MadMetaverse/mad_polygon GitHub repository and was audited after the commit 11f1729.

2.1 Summary

Project name	Mad Metaverse Polygon
URL	https://madmetaverse.com
Platform	Polygon Network
Language	Solidity

2.2 Contracts

Name	Address
IAdmin	https://github.com/MadMetaverse/mad_polygon/blob/11f17299c91baa02b1bec1dfb54ccd309282df5f/contracts/interfaces/IAdmin.sol
ICellRepository	https://github.com/MadMetaverse/mad_polygon/blob/11f17299c91baa02b1bec1dfb54ccd309282df5f/contracts/interfaces/ICellRepository.sol
IERC721URIExtensible	https://github.com/MadMetaverse/mad_polygon/blob/11f17299c91baa02b1bec1dfb54ccd309282df5f/contracts/interfaces/IERC721URIExtensible.sol
IEnhancerRepository	https://github.com/MadMetaverse/mad_polygon/blob/11f17299c91baa02b1bec1dfb54ccd309282df5f/contracts/interfaces/IEnhancerRepository.sol
IExternalNftRepository	https://github.com/MadMetaverse/mad_polygon/blob/11f17299c91baa02b1bec1dfb54ccd309282df5f/contracts/interfaces/IExternalNftRepository.sol
ILaboratory	https://github.com/MadMetaverse/mad_polygon/blob/11f17299c91baa02b1bec1dfb54ccd309282df5f/contracts/interfaces/ILaboratory.sol
IMarketplace	https://github.com/MadMetaverse/mad_polygon/blob/11f17299c91baa02b1bec1dfb54ccd309282df5f/contracts/interfaces/IMarketPlace.sol
IModule	https://github.com/MadMetaverse/mad_polygon/blob/11f17299c91baa02b1bec1dfb54ccd309282df5f/contracts/interfaces/IModule.sol
IModuleBox	https://github.com/MadMetaverse/mad_polygon/blob/11f17299c91baa02b1bec1dfb54ccd309282df5f/contracts/interfaces/IModuleBox.sol
ITokenSetter	https://github.com/MadMetaverse/mad_polygon/blob/11f17299c91baa02b1bec1dfb54ccd309282df5f/contracts/interfaces/ITokenSetter.sol

IStake	https://github.com/MadMetaverse/mad_polygon/blob/11f17299c91baa02b1bec1dfb54ccd309282df5f/contracts/interfaces/IStake.sol
IMintable	https://github.com/MadMetaverse/mad_polygon/blob/11f17299c91baa02b1bec1dfb54ccd309282df5f/contracts/interfaces/IMintable.sol
CellData	https://github.com/MadMetaverse/mad_polygon/blob/11f17299c91baa02b1bec1dfb54ccd309282df5f/contracts/libs/CellData.sol
Enhancer	https://github.com/MadMetaverse/mad_polygon/blob/11f17299c91baa02b1bec1dfb54ccd309282df5f/contracts/libs/Enhancer.sol
Stake	https://github.com/MadMetaverse/mad_polygon/blob/11f17299c91baa02b1bec1dfb54ccd309282df5f/contracts/Stake.sol
Random	https://github.com/MadMetaverse/mad_polygon/blob/11f17299c91baa02b1bec1dfb54ccd309282df5f/contracts/Random.sol
NanoCell	https://github.com/MadMetaverse/mad_polygon/blob/11f17299c91baa02b1bec1dfb54ccd309282df5f/contracts/NanoCell.sol
MetaCell	https://github.com/MadMetaverse/mad_polygon/blob/11f17299c91baa02b1bec1dfb54ccd309282df5f/contracts/MetaCell.sol
Module	https://github.com/MadMetaverse/mad_polygon/blob/11f17299c91baa02b1bec1dfb54ccd309282df5f/contracts/Module.sol
MDMA	https://github.com/MadMetaverse/mad_polygon/blob/11f17299c91baa02b1bec1dfb54ccd309282df5f/contracts/MDMA.sol
Marketplace	https://github.com/MadMetaverse/mad_polygon/blob/main/contracts/Marketplace.sol
Laboratory	https://github.com/MadMetaverse/mad_polygon/blob/main/contracts/Laboratory.sol

3. Found issues

Critical	2 (3%)
High	10 (15%)
Medium	7 (11%)
Low	36 (55%)
Info	11 (16%)

C1. IAdmin

ID	Severity	Title	Status
C1-01	Low	Lack of validation of input parameters	Open
C1-02	Info	Typos	Open

C2. ICellRepository

ID	Severity	Title	Status
C2-01	High	Open storage operations	Open
C2-02	Low	Redundant validation for uint type	Open
C2-03	Low	Lack of events	Open

C3. IERC721URIExtensible

ID	Severity	Title	Status
C3-01	Low	Unused library	Open
C3-02	Low	Gas optimization	Open

C4. IEnhancerRepository

ID	Severity	Title	Status
C4-01	Low	Gas optimization	Open
C4-02	Info	Naming convention	Open

C5. IExternalNftRepository

ID	Severity	Title	Status
C5-01	High	Adding NFT	Open
C5-02	Low	Lack of events	Open
C5-03	Low	Gas optimization	Open

C6. ILaboratory

ID	Severity	Title	Status
C6-01	Info	Typos	Open

C7. IMarketplace

ID	Severity	Title	Status
C7-01	Info	Typos	Open

C8. IModule

ID	Severity	Title	Status
C8-01	Low	Unused import	Open

C10. ITokenSetter

ID	Severity	Title	Status
C10-01	Info	Typos	Open

C11. IStake

ID	Severity	Title	Status
C11-01	Low	Unused import	Open
C11-02	Info	Typos	Open

C13. CellData

ID	Severity	Title	Status
C13-01	Info	Typos	Open

C14. Enhancer

ID	Severity	Title	Status
C14-01	Low	Gas optimization	Open

C15. Stake

ID	Severity	Title	Status
C15-01	Low	Lack of events	Open
C15-02	Low	Unused modifier	Open
C15-03	Low	Gas optimization	Open
C15-04	Low	Unused event	Open
C15-05	Info	Typos	Open

C16. Random

ID	Severity	Title	Status
C16-01	Medium	Predictable "random"	Open
C16-02	Medium	Uninitialized state variables	Open
C16-03	Low	Lack of events	Open

C16-04	Low	Redundant validation for uint type	Open
C16-05	Low	Gas optimization	Open

C17. NanoCell

ID	Severity	Title	Status
C17-01	High	Open burn	Open
C17-02	Low	Gas optimization	Open

C18. MetaCell

ID	Severity	Title	Status
C18-01	Critical	Open burn	Open

C19. Module

ID	Severity	Title	Status
C19-01	Critical	Open burn	Open
C19-02	Low	Gas optimization	Open

C20. MDMA

ID	Severity	Title	Status
C20-01	Medium	Minting by admins	Open
C20-02	Low	Gas optimization	Open

C21. Marketplace

ID	Severity	Title	Status
C21-01	High	Changing the characteristics of the purchased Enhancer	Open
C21-02	High	Unlimited fee amount	Open
C21-03	High	Wrong check for msg.sender	Open
C21-04	High	Purchase at zero cost	Open
C21-05	High	Locked native currency	Open
C21-06	Medium	No change for purchase	Open
C21-07	Low	Lack of event	Open
C21-08	Low	Redundant balance check	Open
C21-09	Low	Two prices payable	Open
C21-10	Low	Using approve with transferFrom instaed of transfer	Open
C21-11	Low	Changing fee percent	Open
C21-12	Low	Lack of validation	Open
C21-13	Low	Gas optimization	Open

C21-14	Low	Locked tokens	Open
C21-15	Low	Lack of zero-address validation	Open
C21-16	Low	Redundant uint validation	Open
C21-17	Info	Typos	Open

C22. Laboratory

ID	Severity	Title	Status
C22-01	High	Adding NFT	Open
C22-02	High	Administrator intervention in user tokens	Open
C22-03	Medium	Mutating with NFT	Open
C22-04	Medium	Collision of tokens ID	Open
C22-05	Medium	Predictable "random" values	Open
C22-06	Low	Lack of reason message in require	Open
C22-07	Low	Locked tokens in the contract	Open
C22-08	Low	Lack of validation	Open
C22-09	Low	Redundant uint validation	Open
C22-10	Low	Gas optimization	Open
C22-11	Info	Modifier for view function	Open
C22-12	Info	Same image for different tokens	Open

4. Contracts

C1. IAdmin

Overview

This contract stores addresses with admin rights.

Issues

C1-01 Lack of validation of input parameters

Low Open

The functions `addAdmin()`, `removeAdmin()` do not check the address `_admin` for a non-zero address.

C1-02 Typos

Info Open

Typos reduce the code's readability.1) L7 'alowed' should be replaced with 'allowed'2) L7 'additiona' should be replaced with 'additional'

C2. ICellRepository

Overview

The abstract contract stores information about issued MetaCell tokens.

Issues

C2-01 Open storage operations

High Open

Each user can call the `addMetaCell()`, `removeMetaCell()` and `updateMetaCell()` functions, thereby:a. can create information about a non-existent NFT token using the `addMetaCell()`

function. Also, it breaks the minting of new NFT tokens by Laboratory (by `mintMetaCell()` function), if somebody mints a token whose id (index) already exists in ICellRepository.b. can remove information about an existing NFT token using the removeMetaCell() function.c. can change the price of the token being sold or any other parameters using the `updateMetaCell()` function. This allows buying the token at zero price or breaking the sale.

As a result, with this vulnerability, anyone can change any information, change sales prices, withdraw from sales, appropriate other people's tokens and receive proceeds for the sale of someone else's token.

C2-02 Redundant validation for uint type

Low Open

In L29, L34, L51, L56, and L104 there are redundant validations because the `uint` type >= 0 by default.

C2-03 Lack of events

Low Open

The functions `addMetaCell()`, `removeMetaCell()`, `updateMetaCell()` don't emit events, which complicates the tracking of important off-chain changes.

C3. IERC721URIExtensible

Overview

ERC721 token with storage based token URI management

Issues

C3-01 Unused library

Low Open

The functionality of the imported `hardhat/console.sol` library (L7) is not used in the contract. Thus, it can be removed to save gas in deployment.

C3-02 Gas optimization

Low Open

The variable `_extension` can be declared as `constant` to save gas.

C4. IEnhancerRepository

Overview

The abstract contract stores information about enhancers for NFT.

Issues

C4-01 Gas optimization

Low Open

a. The `getAllEnhancers()`, `getEnhancerTypes()`, `getUserEnhancers()` functions can be declared as external to save gas.b. Using `mapping` can optimize the search in `_increaseEnchancersAmount()`, `_decreaseEnhancersAmount()`, `getEnhancerAmount()` and `findEnhancerById()` functions and save gas.

c. The value of `ownedEnhancers[_owner].length` is read at each step of the loop on L78. To save gas, consider using a local variable instead of reading the storage every time.

C4-02 Naming convention

Info Open

a. The `getAllEnhancers()`, `getEnhancerTypes()`, `getUserEnhancers()` functions can be declared as external to save gas.b. Using `mapping` can optimize the search in `_increaseEnchancersAmount()`, `_decreaseEnhancersAmount()`, `getEnhancerAmount()` and `findEnhancerById()` functions and save gas.

C5. IExternalNftRepository

Overview

The abstract contract stores information about issued NFT tokens.

The logic of the contract looks unfinished, and there is a lack of detailed documentation.

Issues

C5-01 Adding NFT

High Open

Anyone has the ability to perform the `addNft()` function and update the storage of the `IExternalNftRepository` with any data without any validation.

Recommendation

Make sure this behavior is intended. Otherwise, it is necessary to refine the logic of using and adding NFT.

C5-02 Lack of events

Low Open

The function `addNft()` doesn't emit an event, which complicates the tracking of important off-chain changes.

C5-03 Gas optimization

Low Open

a. The private state variables `nftIdToIndex` and `nftLatestIndex` are never used and should be removed.b. The private state variable `nftIndexesArray` is only used in L49, but is not called anywhere. Thus, the `nftIndexesArray` variable is redundant and can be removed from the contract. It can save gas in deployment and executions.

C6. ILaboratory

Overview

This interface has a description of the functionality for the Laboratory contract.

Issues

C6-01 Typos

Info Open

Typos reduce the code's readability.1) L57 'Evoution' should be replaced with 'Evolution' 2) L65 'Evoution' should be replaced with 'Evolution'3) L11 'emited' should be replaced with 'emitted'4) L18 'emited' should be replaced with 'emitted'5) L25 'emited' should be replaced with 'emitted'6) L31 'emited' should be replaced with 'emitted'7) L57 'emmited' should be replaced with 'emitted'8) L65 'emmited' should be replaced with 'emitted'9) L73 'emmited' should be replaced with 'emitted'

C7. IMarketplace

Overview

This interface describes the functionality of the Marketplace contract, in which the sale of various NFT tokens will be carried out.

Issues

C7-01 Typos

Info Open

Typos reduce the code's readability.1) L8 'adress' should be replaced with 'address'2) L10 'Adress' should be replaced with 'Address'3) L89 'Adress' should be replaced with 'Address'4) L96 'Adress' should be replaced with 'Address'5) L103 'Adress' should be replaced with 'Address'6) L158 'thata' should be replaced with 'that'7) L180 'availbale' should be replaced with 'available'

C8. IModule

Overview

This interface describes the functionality of the Module contract, which changes the module state on the marketplace.

Issues

C8-01 Unused import

Low Open

The functionality of the imported `Module` library (L3) is not used in the interface. Thus, it can be removed to save gas in deployment.

C9. IModuleBox

Overview

This interface describes the functionality of working with boxes.

C10. ITokenSetter

Overview

This interface describes the functionality of storing token addresses, with which you can buy enhancers on the marketplace.

Issues

C10-01 Typos

Info Open

Typos reduce the code's readability.1) L10 'Preveleged' should be replaced with 'privileged' 2) L15 'Preveleged' should be replaced with 'privileged'3) L20 'preveleged' should be replaced with 'privileged'4) L23 'preveledged' should be replaced with 'privileged'

C11. IStake

Overview

This interface describes the functionality of a staking contract.

Issues

C11-01 Unused import

Low Open

The functionality of the imported `Pair` library (L3) is not used in this interface. Thus, it can be removed to save gas in deployment.

C11-02 Typos

Info Open

Typos reduce the code's readability.L13 'mutatitions' should be replaced with 'mutations'

C12. IMintable

Overview

This interface describes the functionality of mint and burning tokens.

C13. CellData

Overview

This library describes the Cell structure with its fields and auxiliary functions for it.

Issues

C13-01 Typos

Info Open

Typos reduce the code's readability.L9 'standart' should be replaced with 'standard'

C14. Enhancer

Overview

This library describes the Enhancer structure with its fields and auxiliary functions for it.

Issues

C14-01 Gas optimization

Low Open

The variable `probability` of the structure `Enhancer` should be casted to uint16 type. This saves gas on storage in the structure, and does not require extra costs when performing function `Laboratory._evolve()` on L178.

C15. Stake

Overview

The contract allows to add tokens to the privileged list

In the current version, the contract is not explicitly linked to other contracts. Without documentation, it is impossible to fully speak about the correctness of the contract

Issues

C15-01 Lack of events

Low Open

The function `setRepository()` doesn't emit events, which complicates the tracking of important off-chain changes.

C15-02 Unused modifier

Low Open

The `supportedTokens()` modifier is never used in the contract. Consider using it otherwise, it can be removed to save gas on deployment.

C15-03 Gas optimization

Low Open

The functionality of the imported `IERC721Receiver`, `IERC1155`, `IERC1155Receiver` contracts is not used in this contract. Thus, it can be removed to save gas in deployment.

C15-04 Unused event

Low Open

The event `ExternalNftAdded` is never used in the contract. Consider using it otherwise, it can be removed to save gas in deployment.

C15-05 Typos

Info Open

Typos reduce the code's readability.1) L57 'Preveleged' should be replaced with 'privileged' 2) L66 'Preveleged' should be replaced with 'privileged'3) L72 'preveleged' should be replaced with 'privileged'4) L88 'preveledged' should be replaced with 'privileged'

C16. Random

Overview

The contract implements the logic of generating pseudo-random values for the subsequent change in the characteristics of objects.

Issues

C16-01 Predictable "random"

Medium Open

The result of the execution of the `random()` function can be predicted in advance because users can access variable `mSeed` since all data on the blockchain can be read.

Recommendation

Consider changing the approach to obtaining random values. For example, you can use ChainLink Verifiable Random Function.

C16-02 Uninitialized state variables

Medium Open

State variables `_baseURI`, `_stageRange`, `_tokenURIs` are not initialized anywhere, but are used in the contract functions with zero-values.

Recommendation

Consider adding functionality to update the variables mentioned above.

C16-03 Lack of events

Low Open

The function `setSeed()` doesn't emit events, which complicates the tracking of important off-chain changes.

C16-04 Redundant validation for uint type

Low Open

On L97, L103 there is redundant validation because the `uint` variable >= 0 by default.

C16-05 Gas optimization

Low Open

a. The state variable `_baseURI` has default private visisbility and is never defined in the contract. It can be removed to save gas in deployment.

b. The function `tokenURI()` has default private visisbility and never used in the contract. It can be removed to save gas in deployment.

c. The assignment on L77 looks a bit complicated.

C17. NanoCell

Overview

This contract provides the basic functionality of the NFT of the ERC-721 standard.

The contract admin has the ability to mint these tokens.

Issues

C17-01 Open burn

High Open

The `burn()` function does not perform any checks of the owner of the token and is completely open to any user, thus anyone can burn the NFT of another user by passing the required tokenId to the function arguments.

Recommendation

The `burn()` function must check the ownership of the token. Only the owner of the token should be able to burn it.

C17-02 Gas optimization

Low Open

Calling `super.transferOwnership(msg.sender)` on L18 is redundant because the contract is inherited from the Ownable contract. And in the Ownable contract, the same call is made in the constructor.

C18. MetaCell

Overview

This contract provides the basic functionality of the NFT of the ERC-721 standard.

The contract owner has the ability to mint these tokens.

Issues

C18-01 Open burn

Critical Open

The `burn()` function does not perform any checks of the owner of the token and is completely open to any user, thus anyone can burn the NFT of another user by passing the required tokenId to the function arguments.

Recommendation

The `burn()` function must check the ownership of the token. Only the owner of the token should be able to burn it.

C19. Module

Overview

This contract provides the basic functionality of the NFT of the ERC-721 standard.

The contract admin has the ability to mint these tokens.

Issues

C19-01 Open burn

Critical Open

The `burn()` function does not perform any checks of the owner of the token and is completely open to any user, thus anyone can burn the NFT of another user by passing the required tokenId to the function arguments.

Recommendation

The `burn()` function must check the ownership of the token. Only the owner of the token should be able to burn it.

C19-02 Gas optimization

Low Open

The internal `_burn()` function is not used anywhere and can be removed to save gas.

C20. MDMA

Overview

This contract provides the basic functionality of the ERC-20 tokens.

The contract admins have the ability to mint these tokens.

Issues

C20-01 Minting by admins

Medium Open

The contract owner has the ability to add unlimited numbers admins, who can mint unlimited amount of tokens using the `mint()` function.

This can lead to incorrect tokenomics.

Recommendation

Make sure this behavior is intended. Otherwise, restrict admins.

C20-02 Gas optimization

Low Open

Calling `_transferOwnership(msg.sender)` in L16 is redundant because the contract is inherited from the Ownable contract. And in the Ownable contract, the same call is made in the constructor.

C21. Marketplace

Overview

The contract allows to buy, sell tokens and buy upgrades for users' tokens.

Issues

C21-01 Changing the characteristics of the purchased Enhancer

High Open

The admin of the contract can perform the `modifyEnhancer()` function to change characteristics (probability, price, etc.) of the existing Enhancer stored in the `IEnhancerRepository`.

There may be a situation when a user buys an Enhancer with a "probability" of 30, and the admin later changes this "probability" to 10. Thus the user ends up deceived.

Recommendation

Limit changes to already purchased Enhancers.

C21-02 Unlimited fee amount

High Open

The contract admin has the ability to set any fee (even more than 100%) using the `setFeeQuota()` function. This causes the sellers to get nothing when they try to sell their tokens.

Recommendation

It is necessary to limit the fee percent that the admin can set.

C21-03 Wrong check for msg.sender

High Open

The function `updateModulesPrice()` allows updating the price of the selling token. Due to a `require` check in L523, the price of the token can be changed by NOT the owner. Thus, anyone can set the price to zero and buy the token.

```
function updateModulesPrice(uint256 id, uint256 price) external override {
    require(IERC721(module).ownerOf(id) != msg.sender, "You are the owner");
    require(
        EnumerableSet.contains(modules, id),
        "Module is not in marketplace"
    );
    modulesOnSale[id] = price;
    emit ModulePriceUpdated(id);
}
```

The reason message in the `require` statement looks suspicious.

Recommendation

It is necessary to fix the check for `msg.sender` in L523.

C21-04 Purchase at zero cost

High Open

Due to the wrong operator '`>=`' in L185 of the `buyEnhancerForETH()` function, users have the ability to buy Enhancer by providing `msg.value = 0`.

```
function buyEnhancerForETH(uint256 _enhancerId, uint256 _amount)
        external
        payable
        override
    {
       ...
        require(
            enhancer.basePrice.mul(_amount) >= msg.value,
            "Not enough funds"
        );
        payable(feeWallet).transfer(msg.value);
        _buyEnhancer(_enhancerId, _amount);
    }
```

Recommendation

The `msg.value` should be equal to token (or amount of token) price:

```
require(enhancer.basePrice.mul(_amount) == msg.value, "Not enough funds");
```

C21-05 Locked native currency

High Open

While performing the `buyMetaCell()` function one part of `msg.value` sends to the old owner (L339) and another (fee) stays in the contract. There is no functionality in the contract for the withdrawal of collected fees.

Also, the `ModuleBox` buyers pay in native currency (L633). And all the funds are locked in the

contract.

Recommendation

Add functionality to withdraw native currency from the contract.

C21-06 No change for purchase

Medium Open

Users can buy MDMA token using the `buyMDMAToken()` function. Due to the condition in the `require` statement (L136), the user can mistakenly send more native currency than required and will not receive change.

```
function buyMDMAToken(uint256 _amount) external payable override {
    require(
        msg.value >= _amount.mul(mdmaTokenPrice) / 1 ether,
        "Not enough ether to buy"
    );
    ...
}
```

Recommendation

Consider making a strict check for message value:

```
msg.value == _amount.mul(mdmaTokenPrice) / 1 ether
```

C21-07 Lack of event

Low Open

The function `setModulesAddress()` doesn't emit events, which complicates the tracking of important off-chain changes.

C21-08 Redundant balance check

Low Open

No need to check the user balance before transferring in L457, L539, and L635 because it will

be checked inside the `ERC20._transfer()` function.

C21-09 Two prices payable

Low Open

Users, who want to buy ModuleBox using the `buyBox()` function should pay native tokens (L633) and MDMA tokens (L640) both.

Recommendation

Make sure this behavior is intended. Otherwise, remove one of the payable methods.

C21-10 Using approve with transferFrom instaed of transfer

Low Open

No need to use `approve()` with `transferFrom()` functions instead of the `transfer()` function in L466-470, L548-552 because it is more expensive.

C21-11 Changing fee percent

Low Open

The fee for the token being sold can be changed after the start of the sale.

A seller who has placed a token for sale may receive significantly less than what was originally expected. This is possible if the contract owner changes the fee using the `setFeeQuota()` function, after starting the selling (L328, L465, L547).

For example, the contract `feeQuote` is 10% and Alice wants to sell the token module for 49.5 MDMA tokens. So she sets the price to 55 MDMA and waits for someone to buy it. After the admin sets `feeQuote` to 20%, Bob buys a Module token for 55 MDMA. But Alice will only get 44 (expected 49.5)

C21-12 Lack of validation

Low Open

a. The `setMDMATokenPrice()`, `updateModulesPrice()`, `updateNanoCellPrice()`, `setPricePerBox()` functions allow to set price but have no validation.

b. There is no validation for input parameters of the `createEnhancer()` function.

Recommendation

Consider adding validations to prevent mistakes while executing the contract.

C21-13 Gas optimization

Low Open

a. The result of executing the `IEnhancerRepository(laboratory).getEnhancerInfo(_enhancer.id)` function in L233 and L239 should be written to a local variable. This will save gas, as there will be no double search in the cycle in another contract.

b. Calling `_transferOwnership(msg.sender)` in L44 is redundant because the contract is inherited from the Ownable contract. And in the Ownable contract, the same call is made in the constructor.

C21-14 Locked tokens

Low Open

All tokens paid by users when purchasing using the `buyEnhancerForToken()` function (L210) will be locked in the contract because there is no functionality to withdraw them in the contract.

This also applies to the fees in the `buyModule()` (L473), `buyNanoCell()` (L555) functions.

C21-15 Lack of zero-address validation

Low Open

The `setMDMAToken()`, `setMetaCellToken()`, `setNanoCellToken()`, `setLaboratory()`, `setModulesAddress()` functions don't check the input address parameter for a non-zero address.

C21-16 Redundant uint validation

Low Open

The first part of the `require` statements in L377 and L427 do nothing because `uint` type is >= 0 by default.

Also, the check in L435 is redundant.

Recommendation

Can't say for sure without the documentation, but the `require` statements in L377, L427 should probably be something like the following code, because the first ID of the MetaCell is 1:

```
_tokenId > 0 && _tokenId != type(uint256).max
```

C21-17 Typos

Info Open

Typos reduce the code's readability:1) L236 "doesn't" should be replaced with "doesn't".

C22. Laboratory

Overview

Contract allows to create new NFT tokens and upgrade them using other tokens

Issues

C22-01 Adding NFT

High Open

Anyone has the ability to perform `addNft()` function and update storage of the `IExternalNftRepository` with any data without validation.

Recommendation

Make sure this behavior is intended. Otherwise, it is necessary to refine the logic of using and adding NFT.

C22-02 Administrator intervention in user tokens

High Open

The users can buy `Enhancers` on marketplace for ETH or tokens (using `buyEnhancerForETH()` or `buyEnhancerForToken()` functions) to evolve their `MetaCell` tokens.

The contract owner has the ability to add an unlimited number of admins using the `addAdmin()` function.

Any of the admins can reduce (or increase) the number of `Enhacers` owned by the user using the `decreaseEnhancersAmount()` (or `increaseEnhancersAmount()`) function.

Thus, the user may be deceived, or the user may be unfairly given an advantage over others.

Recommendation

Restrict rights of admins to change users' Enhancer amount.

C22-03 Mutating with NFT

Medium Open

The `mutate()` function looks strange, and the NFT functionality seems to be underdeveloped.

1. The user can successfully perform `mutate()` function with any `_nftId` parameter because there is no validation for NFT.
2. The function `_mergeWithNft()` doesn't use NFT parameter internally.
3. The result of the `_markNftAsUsed()` function is never checked afterward.
4. The `require` statement in L250 does nothing. And probably it should check the following parameters:

```
nft.tokenId != type(uint256).max && !nft.isUsed;
```

Due to the lack of documentation, it cannot be said that the function works as expected.

Recommendation

Consider using all functionality of the `IExternalNftRepository` contract. Add proper validations to the `mutate()` function and documentation.

C22-04 Collision of tokens ID

Medium Open

The `_createNewToken()` function creates new `Cells` in the `ICellRepository` with id starting from 1001 (because `numOptions = 1000`). This approach can lead to collisions of `Cell.tokenId` if there exist more than 1000 `scientistTokens` (L406).

In other words, it will be unable to perform `mintMetaCell()` function 1001 times if somebody already created a new token with ID 1001, due to `require` in L36 of the `ICellRepository`.

Recommendation

Make sure this behavior is intended. Otherwise, consider increasing the value of the `NUM_OPTIONS` state variable.

C22-05 Predictable "random" values

Medium Open

The values of the `cell` calculated in L183, L188, L196, L201 of the `_evolve()` function can be predicted in advance (see details in the contract Random). This can cause users to call the `_evolve()` function at the right time to get the result they want.

Recommendation

The approach to obtaining random values should be changed.

C22-06 Lack of reason message in require

Low Open

The reason message in the `require` statement is omitted in L407.

The message should be included as this can make it easier to understand what errors are occurring.

C22-07 Locked tokens in the contract

Low Open

When users execute the `boostCell()` function, `madTokens` are transferred to the `Laboratory` contract. These tokens will be locked, as there is no functionality for withdrawing them from the contract.

C22-08 Lack of validation

Low Open

There is no validation for the `_price` parameter of the function `setBoostPerBlockPrice()`.

C22-09 Redundant uint validation

Low Open

In L45, L228, and L250 there is redundant validation because the `uint` type is >= 0 by default.

C22-10 Gas optimization

Low Open

a. The state variable `metaCell` can be declared as `immutable` to save gas.

b. The state variable `NUM_OPTIONS` can be declared as `constant` to save gas.

c. Checking the ERC20-token balance in L224 is redundant, because it will be checked in performing `transferFrom()` function in L238.

d. The '`nft`' parameter of the `_mergeWithNft()` function is never used and can be removed to save gas. e. Calling `super.transferOwnership(msg.sender)` in L62 is redundant because the contract is inherited from the Ownable contract. And in the Ownable contract, the same call is made in the constructor.

C22-11 Modifier for view function

Info Open

The view function `getSeed()` is restricted by the modifier `isAdmin()` and allows viewing the private variable of the Random contract. But all information, even private, stored in the blockchain can be read at any time. Therefore, such a restriction seems redundant.

C22-12 Same image for different tokens

Info Open

When the user executes the `mutate()` function they gets a new `tokenUri` in L307 for his cell. Due to the `getRandomImageByRange()` function, a situation may occur when several users have the same image.

Recommendation

Make sure this behavior is intended.

5. Conclusion

2 critical, 10 high, 7 medium, 36 low and 11 informational severity issues were found.

The contracts MetaCell, Module, NanoCell, and MDMA are highly dependent on the owner's account. During deployment, the ownership of the MetaCell contract must be transferred to the Marketplace contract. The contracts MDMA and NanoCell must be administered by the contract Laboratory.

We strongly suggest adding unit and functional tests for all contracts.

We also recommend using pragma fixed to the version the contracts have been tested and are intended to be deployed with. This helps to avoid deploying using an outdated compiler version and shields from possible bugs in future solidity releases.

This audit includes recommendations on improving the code and preventing potential attacks.

Appendix A. Issues severity classification

- **Critical.** Issues that may cause an unlimited loss of funds or entirely break the contract workflow. Malicious code (including malicious modification of libraries) is also treated as a critical severity issue. These issues must be fixed before deployments or fixed in already running projects as soon as possible.
- **High.** Issues that may lead to a limited loss of funds, break interaction with users, or other contracts under specific conditions. Also, issues in a smart contract, that allow a privileged account the ability to steal or block other users' funds.
- **Medium.** Issues that do not lead to a loss of funds directly, but break the contract logic. May lead to failures in contracts operation.
- **Low.** Issues that are of a non-optimal code character, for instance, gas optimization tips, unused variables, errors in messages.
- **Info.** Issues that do not impact the contract operation. Usually, info severity issues are related to code best practices, e.g. style guide.

Appendix B. List of examined issue types

- Business logic overview
- Functionality checks
- Following best practices
- Access control and authorization
- Reentrancy attacks
- Front-run attacks
- DoS with (unexpected) revert
- DoS with block gas limit
- Transaction-ordering dependence
- ERC/BEP and other standards violation
- Unchecked math
- Implicit visibility levels
- Excessive gas usage
- Timestamp dependence
- Forcibly sending ether to a contract
- Weak sources of randomness
- Shadowing state variables
- Usage of deprecated code

contact@hashex.org

@hashex_manager

blog.hashex.org

linkedin

github

twitter

#HashEx
BLOCKCHAIN SECURITY