Security Audit

Date : 2024-05-17

Zus Contracts

Executive Summary

Type	Smart Contract Audit
Audit Timeline	16 days
Runtime Environment	EVM
Languages	Solidity

Scope

Github Repository Link : https://github.com/0chain/nft-dstorage-core

/contracts/Config.sol

/contracts/DStorageERC721.sol

/contracts/DStorageERC721Fixed.sol

/contracts/DStorageERC721Pack.sol

/contracts/DStorageERC721Random.sol

/contracts/Factory.sol

/contracts/FactoryModuleERC721.sol

/contracts/FactoryModuleERC721Fixed.sol

/contracts/FactoryModuleERC721Pack.sol

/contracts/FactoryModuleERC721Random.sol

/contracts/interfaces/*

Summary of Findings

ID	Name	Description	Severity
M-01	Centralization Risk: Funds can be frozen when critical key holders lose access to their keys	Funds will be frozen in case of lost keys to the account	Medium
L-01	Use safeTransferFrom instead of transferFrom for ERC721 transfers	use of safeTransferFrom is encouraged and considered a best practice	Low

L-02	Lack of input validation on State Changing functions	The contract does not validate inputs for many state changing functions.	Low
L-03	Unused Event Declaration	The Test event is declared but never used in the contract DStorageERC721	Low
L-04	Incorrect Payment Amount Check in mintOwner function	Check should allow for msg.value to be greater than or equal to amount * p to handle edge cases where more Ether is sent.	Low
G-01	Caching msg.value in Local Variables can help save gas	The contracts do not cache msg.value in local variables, leading to higher gas costs.	Gas
G-02	Use Assembly for Zero Address Checks	Contracts perform zero address checks using standard Solidity code, which can be optimized using assembly.	Gas
G-03	Inlining Internal Functions can help save gas	Internal functions that are called only once can be inlined to save gas	Gas
G-04	Using `calldata` instead of `memory` for read-only arguments saves gas	Use calldata for arguments that are not changed while the function calls.	Gas

G-05	For Loop can be optimized	Loop can be optimized to save gas	Gas
I-01	Consider using custom revert messages	Custom revert messages are considered best practice and also help save gas	Informational

Findings

[M-01] Centralization Risk: Funds can be frozen when critical key holders lose access to their keys

Severity: Medium

Location : Config.sol , DStorageERC721.sol , DStorageERC721Fixed.sol , DStorageERC721Pack.sol, DStorageERC721Random

Description: The Owner can update key parameters and withdraw funds from the protocol. This introduces a high centralization risk, which can cause funds to be frozen in the contract if the key holders lose access to their keys.

POC :

```
    function setUint256(bytes32 key, uint256 value) external onlyOwner {
        emit ConfigUpdated(key, _config[key], value);
        _config[key] = value;
    }
```

Impact
The funds can get locked in the contract in a case of lost keys and the key protocol parameters will also be at risk of getting changed.

Recommendation

Ensure that the owner is a MultiSig wallet . Also you can introduce a time bounded transfer of ownership of the contracts in case the keys to the owner account are lost.

[L-01] Use safeTransferFrom instead of transferFrom for ERC721 transfers

Severity: Low

Location : DStorageERC721Pack.sol

Description: OpenZeppelin's documentation discourages the use of transferFrom(), use safeTransferFrom() whenever possible

Impact
Since the contract uses its own NFT's and there is no option to actually list other NFT's contracts so there is not much of an impact on the protocol.It's a good design choice to use safeTransferFrom for ERC721 transfers.

Recommendation
Call the **safeTransferFrom()** method instead of **transferFrom()** for NFT transfers.

POC :

```
for (uint256 i; i < size; i++) {
            r.transferFrom(address(this), msg.sender, c + i);
        }
```

Github Permalink :
https://github.com/0chain/nft-dstorage-core/blob/main/contracts/DStorageERC721Pack.sol#L220

Similar Findings :
https://solodit.xyz/issues/m-09-use-safetransferfrom-instead-of-transferfrom-for-erc721-transfers-code4rena-cally-cally-contest-git

[L-02] Lack of input validation on State Changing functions

Severity: Low

Location : DStorageERC721Pack.sol, DStorageERC721Pack.sol, DStorageERC721Random.sol , Factory.sol, FactoryModuleERC721, FactoryModuleERC721Fixed, FactoryModuleERC721Pack.sol, FactoryModuleERC721Random

Description: The contract does not validate inputs for many state changing functions like **Config::setUint256()** and **Config::setAddress()** functions.

Impact
It's a good practice to check for Null Addresses and values that might not be permitted by the function.

Recommendation
Add basic Null value and address checks at the start of the function.

```
function setAddress(bytes32 key, address value) external onlyOwner {
    require(key != bytes32(0), "Config: invalid key");
    require(value != address(0), "Config: invalid address");
    uint256 val = uint256(uint160(value));
    emit ConfigUpdated(key, _config[key], val);
    _config[key] = val;
}
```

POC :

```
  function setUint256(bytes32 key, uint256 value) external onlyOwner {
    emit ConfigUpdated(key, _config[key], value);
    _config[key] = value;
}
```

Github Permalink :
https://github.com/0chain/nft-dstorage-core/blob/main/contracts/Config.sol#L25
https://github.com/0chain/nft-dstorage-core/blob/main/contracts/Config.sol#L25
https://github.com/0chain/nft-dstorage-core/blob/main/contracts/DStorageERC721Pack.sol#L49
https://github.com/0chain/nft-dstorage-core/blob/main/contracts/DStorageERC721Pack.sol#L49
https://github.com/0chain/nft-dstorage-core/blob/main/contracts/DStorageERC721Pack.sol#L105
https://github.com/0chain/nft-dstorage-core/blob/main/contracts/DStorageERC721Random.sol#L63
https://github.com/0chain/nft-dstorage-core/blob/main/contracts/DStorageERC721Random.sol#L83
https://github.com/0chain/nft-dstorage-core/blob/main/contracts/Factory.sol#L44
https://github.com/0chain/nft-dstorage-core/blob/main/contracts/FactoryModuleERC721.sol#L24
https://github.com/0chain/nft-dstorage-core/blob/main/contracts/FactoryModuleERC721Fixed.sol#L24
https://github.com/0chain/nft-dstorage-core/blob/main/contracts/FactoryModuleERC721Pack.sol#L41
https://github.com/0chain/nft-dstorage-core/blob/main/contracts/FactoryModuleERC721Random.sol#L50

[L-03] Unused Event Declaration

Severity: Low

Location : DStorageERC721.sol

Description: The Test event is declared but never used in the contract DStorageERC721.

Impact
Unused code increases the contract size unnecessarily and may confuse developers.

Recommendation
Remove the unused event

POC :

```
event Test(uint256 price);
```

Github Permalink :
https://github.com/0chain/nft-dstorage-core/blob/main/contracts/DStorageERC721.sol#L178

[L-04] Incorrect Payment Amount Check in mintOwner function

Severity: Low

Location : DStorageERC721Pack.sol

Description: The mintOwner function checks if msg.value is exactly equal to **amount * p**. This check should allow for **msg.value** to be greater than or equal to amount * p to handle edge cases where more Ether is sent.

Impact
The current check could cause valid transactions to revert if a user sends slightly more Ether than required.

Recommendation
Change the check to **msg.value >= amount * p**.

POC:

```
    require(
    msg.value >= amount * p,  "DStorageERC721: invalid owner payment
amount"
);
```

Github Permalink :
https://github.com/0chain/nft-dstorage-core/blob/7f71263a966fc01605d21c2d320dc04f293e0dfb/contracts/DStorageERC721.sol#L200

https://github.com/0chain/nft-dstorage-core/blob/7f71263a966fc01605d21c2d320dc04f293e0dfb/contracts/DStorageERC721Pack.sol#L88

https://github.com/0chain/nft-dstorage-core/blob/7f71263a966fc01605d21c2d320dc04f293e0dfb/contracts/DStorageERC721Random.sol#L110

https://github.com/0chain/nft-dstorage-core/blob/7f71263a966fc01605d21c2d320dc04f293e0dfb/contracts/DStorageERC721Fixed.sol#L72

[G-01] Caching msg.value in Local Variables can help save gas

Severity: Gas

Description
The contracts do not cache msg.value in local variables, leading to higher gas costs.

Impact
Not caching msg.value results in unnecessary gas consumption as msg.value is accessed multiple times.

Recommendation
Cache msg.value in a local variable within functions where it is used multiple times.

```
uint256 value = msg.value;
emit RoyaltyReceived(owner(), value);

if (plan == 0) return;
```

```
uint256 rate = IConfig(config).getUint256(
    keccak256(abi.encodePacked("0chain.0nft.fee.royalty.", plan))
);

require(rate <= 1e18, "DStorageERC721: invalid fee rate");

uint256 fee = (rate * value) / 1e18;
fees += fee;

emit RoyaltyFeesPending(owner(), fee);
emit RoyaltyWithFeesReceived(owner(), value - fee);
```

POC:

```
    /// @notice A global burn nonce. Its first value should be 1, not 0
    uint256 private burnNonce = 0;
    uint256 private userAddressesCounter = 0;
```

Github Permalink :
https://github.com/0chain/nft-dstorage-core/blob/7f71263a966fc01605d21c2d320dc04f293e0dfb/contracts/DStorageERC721.sol#L130-L147

https://github.com/0chain/nft-dstorage-core/blob/7f71263a966fc01605d21c2d320dc04f293e0dfb/contracts/DStorageERC721.sol#L200-L220

https://github.com/0chain/nft-dstorage-core/blob/7f71263a966fc01605d21c2d320dc04f293e0dfb/contracts/DStorageERC721Fixed.sol#L53-L100

https://github.com/0chain/nft-dstorage-core/blob/7f71263a966fc01605d21c2d320dc04f293e0dfb/contracts/DStorageERC721Pack.sol#L105-L136

[G-02] Use Assembly for Zero Address Checks

Severity: Gas

Description
Contracts perform zero address checks using standard Solidity code, which can be optimized using assembly.

Recommendation
Implement assembly code for zero address checks.

POC
https://github.com/0chain/nft-dstorage-core/blob/7f71263a966fc01605d21c2d320dc04f293e0dfb/contracts/Factory.sol#L83

https://github.com/0chain/nft-dstorage-core/blob/7f71263a966fc01605d21c2d320dc04f293e0dfb/contracts/FactoryModuleERC721Pack.sol#L32

https://github.com/0chain/nft-dstorage-core/blob/7f71263a966fc01605d21c2d320dc04f293e0dfb/contracts/FactoryModuleERC721Pack.sol#L33

[G-03] Inlining Internal Functions can help save gas

Severity: Gas

Location : DStorageERC721.sol

Description
Internal functions that are called only once can be inlined to save gas.

Recommendation
Inline internal functions that are called only once.

POC
https://github.com/0chain/nft-dstorage-core/blob/7f71263a966fc01605d21c2d320dc04f293e0dfb/contracts/DStorageERC721.sol#L425

[G-04] Using `calldata` instead of `memory` for read-only arguments saves gas

Severity: Gas

Description
Use calldata for arguments that are not changed while the function calls.

Recommendation
Change memory to calldata in the function parameters.

POC :
https://github.com/0chain/nft-dstorage-core/blob/7f71263a966fc01605d21c2d320dc04f293e0dfb/contracts/Factory.sol#L44-L48

https://github.com/0chain/nft-dstorage-core/blob/7f71263a966fc01605d21c2d320dc04f293e0dfb/contracts/FactoryModuleERC721.sol#L26-L28

https://github.com/0chain/nft-dstorage-core/blob/7f71263a966fc01605d21c2d320dc04f293e0dfb/contracts/FactoryModuleERC721Fixed.sol#L26-L28

https://github.com/0chain/nft-dstorage-core/blob/7f71263a966fc01605d21c2d320dc04f293e0dfb/contracts/FactoryModuleERC721Pack.sol#L43-L45

[G-05] For Loop can be optimized

Severity: Gas

Description
The functions for example the minting functions use a for-loop to iterate over the whole amount of tokens to mint and minting them to the user one by one. This loop accesses the storage for each iteration, which is costly in terms of gas.

POC :
```
        for (uint256 i; i < size; i++) {
            r.transferFrom(address(this), msg.sender, c + i);
        }
```

Recommendation

Don’t initialize ‘i’ as its by default initialized to 0. Also increment ‘i’ in an unchecked block as that also saves gas and overflowing i is virtually impossible. Consider changing the loop to something like this :-

```
       uint256 i=0;
        for (; i < amount; ) {
            t += 1;
            _mint(msg.sender, t);
            unchecked {
                ++i;
            }
        }
```

Github Permalink :

https://github.com/0chain/nft-dstorage-core/blob/7f71263a966fc01605d21c2d320dc04f293e0dfb/contracts/DStorageERC721.sol#L210-L213

https://github.com/0chain/nft-dstorage-core/blob/7f71263a966fc01605d21c2d320dc04f293e0dfb/contracts/DStorageERC721Fixed.sol#L88-L92

https://github.com/0chain/nft-dstorage-core/blob/7f71263a966fc01605d21c2d320dc04f293e0dfb/contracts/DStorageERC721Pack.sol#L156-L162

https://github.com/0chain/nft-dstorage-core/blob/7f71263a966fc01605d21c2d320dc04f293e0dfb/contracts/DStorageERC721Pack.sol#L182-L184

https://github.com/0chain/nft-dstorage-core/blob/7f71263a966fc01605d21c2d320dc04f293e0dfb/contracts/DStorageERC721Pack.sol#L219

https://github.com/0chain/nft-dstorage-core/blob/7f71263a966fc01605d21c2d320dc04f293e0dfb/contracts/DStorageERC721Random.sol#L123-L127

https://github.com/0chain/nft-dstorage-core/blob/7f71263a966fc01605d21c2d320dc04f293e0dfb/contracts/DStorageERC721Random.sol#L175-L178 https://github.com/0chain/nft-dstorage-core/blob/7f71263a966fc01605d21c2d320dc04f293e0dfb/contracts/DStorageERC721Random.sol#L205-L224

https://github.com/0chain/nft-dstorage-core/blob/7f71263a966fc01605d21c2d320dc04f293e0dfb/contracts/DStorageERC721Random.sol#L335-L341

https://github.com/0chain/nft-dstorage-core/blob/7f71263a966fc01605d21c2d320dc04f293e0dfb/contracts/DStorageERC721Random.sol#L375-L388

[I-01] Consider using custom revert messages

Severity: Informational

Description
Since Solidity v0.8.4, the more gas-efficient custom-errors have been introduced. They allow for passing dynamic data in the error and remove costly and repeated string error messages.

Recommendation
Consider replacing required statements with custom errors.
https://blog.soliditylang.org/2021/04/21/custom-errors/