BLOCKHAT
SECURITY

NFT MP

Smart Contract Security Audit

Prepared by BlockHat

December 11th, 2022 - December 15th, 2022

BlockHat.io

contact@blockhat.io

Document Properties

Client	Drchinp
Version	1.0
Classification	Public

Scope

The NFT MP Contracts links

Link	Address
https://mumbai.polygonscan.com/address/0xfcf4DC0C6D4d54D4C2fbFAe7B5D7ab5c51Ed7564#code	0xfcf4DC0C6D4d54D4C2fbFAe7B5D7ab5c51Ed7564
https://mumbai.polygonscan.com/address/0xd943919f005d111f5cc6fd1aca48276ddfddef1f#code	0xD943919f005d111f5cC6fd1aCA48276dDfDDeF1F
https://mumbai.polygonscan.com/address/0x6012ac49df14996cfa98c23e3bff498ba330596c#code	0x6012aC49Df14996CfA98C23e3bff498BA330596c

Files	MD5 Hash
/Lazy1155.sol	c75f522720531a489e2afb4e9afd292f
/Lazy721.sol	632b0a150d8baf44fc39735b270734eb
/utils/Address.sol	0a2130cd7d0e073af01bfe78d9d595be
/utils/StorageSlot.sol	ef62b135881364d2498a7a362064be81
/Proxy.sol	35cbc74ad5eb59d110aa507643e135ef

/TransparentUpgradeableProxy.sol	8daeafe2af87633e94f5aefb44451f03
/ERC1967/ERC1967Proxy.sol	fb05c3e1916442e15adada6d9e311fc5
/ERC1967/ERC1967Upgrade.sol	3bbf0f0512fd0dd4535f8d8aff3c6b3e
/beacon/IBeacon.sol	90a6de6fc5371995115492e9f65194ee

Contacts

COMPANY	CONTACT
BlockHat	contact@blockhat.io

Contents

1 Introduction

NFT MP engaged BlockHat to conduct a security assessment on the NFT MP beginning on December 11th, 2022 and ending December 15th, 2022. In this report, we detail our methodical approach to evaluate potential security issues associated with the implementation of smart contracts, by exposing possible semantic discrepancies between the smart contract code and design document, and by recommending additional ideas to optimize the existing code. Our findings indicate that the current version of smart contracts can still be enhanced further due to the presence of many security and performance concerns.

This document summarizes the findings of our audit.

1.1 About NFT MP

Nft marketplace utility for business

Issuer	Drchinp
Website	--
Type	Solidity Smart Contract
Audit Method	Whitebox

1.2 Approach & Methodology

BlockHat used a combination of manual and automated security testing to achieve a balance between efficiency, timeliness, practicability, and correctness within the audit's scope. While manual testing is advised for identifying problems in logic, procedure, and implementation, automated testing techniques help to expand the coverage of smart contracts and can quickly detect code that does not comply with security best practices.

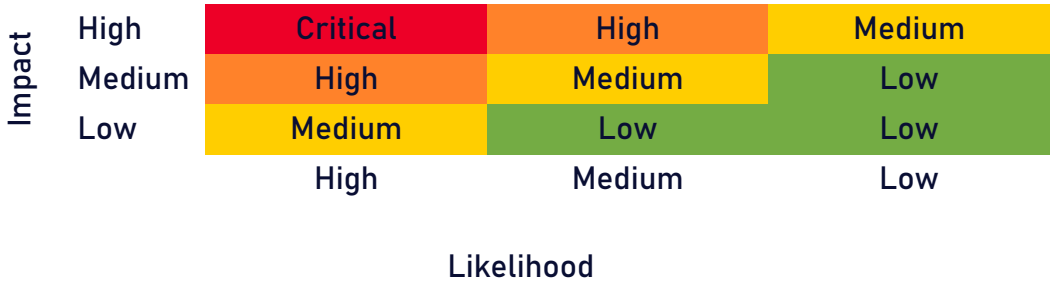
1.2.1 Risk Methodology

Vulnerabilities or bugs identified by BlockHat are ranked using a risk assessment technique that considers both the LIKELIHOOD and IMPACT of a security incident. This framework is effective at conveying the features and consequences of technological vulnerabilities.

Its quantitative paradigm enables repeatable and precise measurement, while also revealing the underlying susceptibility characteristics that were used to calculate the Risk scores. A risk level will be assigned to each vulnerability on a scale of 5 to 1, with 5 indicating the greatest possibility or impact.

- Likelihood quantifies the probability of a certain vulnerability being discovered and exploited in the untamed.
- Impact quantifies the technical and economic costs of a successful attack.
- Severity indicates the risk's overall criticality.

Probability and impact are classified into three categories: H, M, and L, which correspond to high, medium, and low, respectively. Severity is determined by probability and impact and is categorized into four levels, namely Critical, High, Medium, and Low.



2 Findings Overview

2.1 Summary

The following is a synopsis of our conclusions from our analysis of the NFT MP implementation. During the first part of our audit, we examine the smart contract source code and run the codebase via a static code analyzer. The objective here is to find known coding problems statically and then manually check (reject or confirm) issues highlighted by the tool. Additionally, we check business logics, system processes, and DeFi-related components manually to identify potential hazards and/or defects.

2.2 Key Findings

In general, these smart contracts are well-designed and constructed, but their implementation might be improved by addressing the discovered flaws, which include 2 critical-severity, 2 high-severity, 11 low-severity vulnerabilities.

Vulnerabilities	Severity	Status
The royality fee is controlled by the owner	CRITICAL	Acknowledged
The royality fee is controlled by the owner	CRITICAL	Acknowledged
LazyMint implementation	HIGH	Acknowledged
LazyMint implementation	HIGH	Acknowledged
For Loop Over Dynamic Array	LOW	Acknowledged
Floating Pragma	LOW	Acknowledged
For Loop Over Dynamic Array	LOW	Acknowledged
Floating Pragma	LOW	Acknowledged
Floating Pragma	LOW	Acknowledged
Floating Pragma	LOW	Acknowledged
Floating Pragma	LOW	Acknowledged
Floating Pragma	LOW	Acknowledged
Floating Pragma	LOW	Acknowledged
Floating Pragma	LOW	Acknowledged
Floating Pragma	LOW	Acknowledged

Floating Pragma	LOW	Acknowledged

3 Finding Details

A Lazy721.sol

A.1 The royality fee is controlled by the owner [CRITICAL]

Description:

The owner can set any value in fee variable 'feeNumerator'..This means that the owner may not change it in accordance with what was agreed on with the community.This represent a big risk on the user side.

Code:

Listing 1: lazy721.sol

```
1567    function _setTokenRoyalty(
1568        uint256 tokenId,
1569        address receiver,
1570        uint96 feeNumerator
1571    ) internal virtual {
1572        require(feeNumerator <= _feeDenominator(), "ERC2981: royalty fee
            ↪ will exceed salePrice");
1573        require(receiver != address(0), "ERC2981: Invalid parameters");

1575        _tokenRoyaltyInfo[tokenId] = RoyaltyInfo(receiver, feeNumerator);
1576    }
```

Listing 2: lazy721.sol

```
1673    function mint(string memory ipfsmetadata, address from, address to,
            ↪ uint royal, uint256 id_, string memory status) public {
1674        require(msg.sender == owner, "Public Mint Not Available");
1675        if(keccak256(abi.encodePacked((status))) ==
1676            keccak256(abi.encodePacked(("lazy")))){
1677                _lazyMint(from, to, id_);
```

```
1678            }
1679            else{
1680                _safeMint(to, id_);
1681            }
1682            _setTokenURI(id_, ipfsmetadata);
1683            _setTokenRoyalty(id_, from, uint96(royal.div(1e16)));
1684            _creator[id_] = from;
1685            _royal[id_]=royal;
1686        }
```

Risk Level:

Likelihood – 5
Impact - 5

Recommendation:

We recommend to limit the fee value by adding a require statement.

Status - Acknowledged

A.2 LazyMint implementation [HIGH]

Description:

Lazy Minting is a way to defer to the normal minting until right before the NFT is sold. This way, buyers pay the minting fee after their NFT is sold, making NFT creation affordable and equitable for creators. The _lazymint function in this contract is the same as _mint.

Code:

Listing 3: lazy721.sol

```
1288    function _lazyMint(
1289        address from,
1290        address to,
```

```
1291          uint256 tokenId
1292      ) internal virtual {
1293          require(to != address(0) && from != address(0), "ERC721: mint to
                  ↪ the zero address");
1294          require(!_exists(tokenId), "ERC721: token already minted");
1295
1296          _beforeTokenTransfer(address(0), to, tokenId);
1297
1298          _balances[to] += 1;
1299          _owners[tokenId] = to;
1300          emit Transfer(address(0), to, tokenId);
1301          emit Transfer(from, to, tokenId);
1302
1303          _afterTokenTransfer(address(0), to, tokenId);
1304      }
```

Risk Level:

Likelihood – 4
Impact - 5

Recommendation:

We recommend to modify this function to go with the Lazy mint logic.

Status - Acknowledged

The Dev team Acknowledged the Risk because they need Lazymint for seperate emit event.

A.3 For Loop Over Dynamic Array [LOW]

Description:

When smart contracts are deployed or their associated functions are invoked, the execution of these operations always consumes a certain quantity of gas, according to the amount of

computation required to accomplish them. Modifying an unknown-size array that grows in size over time can result in a Denial of Service attack. Simply by having an excessively huge array, users can exceed the gas limit, therefore preventing the transaction from ever succeeding.

Code:

Listing 4: lazy721.sol

```
1689    function mintBatch(string[] memory ipfsmetadata, address[] memory
            ↪ from, address[] memory to, uint256 count, uint royal) public {
1690        require(msg.sender == owner, "Public Mint Not Available");
1691        for (uint256 i = 0; i < ipfsmetadata.length; i++) {
1692            count = count.add(1);
1693            uint256 id_ = count.add(block.timestamp);
1694            _safeMint(to[i], id_);
1695            _setTokenURI(id_, ipfsmetadata[i]);
1696            _creator[id_] = from[i];
1697            _royal[id_]=royal;
1698        }
1699    }
```

Risk Level:

Likelihood – 2
Impact - 2

Recommendation:

Avoid actions that involve looping across the entire data structure. If you really must loop over an array, arrange for it to consume many blocs and thus multiple transactions.

Status - Acknowledged

The Dev team Acknowledged the Risk ; For loop dynamic array is a Standard function so they haven't change it, but they restricted in UIUX.

A.4 Floating Pragma [LOW]

Description:

The contract makes use of the floating-point pragma 0.8.0. Contracts should be deployed using the same compiler version and flags that were used during the testing process.Locking the pragma helps ensure that contracts are not unintentionally deployed using another pragma, such as an obsolete version, that may introduce issues in the contract system

Code:

Listing 5: lazy721.sol

```
14 pragma solidity ^0.8.0;
```

Risk Level:

Likelihood – 1
Impact - 2

Recommendation:

Consider locking the pragma version. It is advised that floating pragma should not be used in production. Both truffle-config.js and hardhat.config.js support locking the pragma version.

Status - Acknowledged

B Lazy1155.sol

B.1 The royality fee is controlled by the owner [CRITICAL]

Description:

The owner can set any value in fee variable 'feeNumerator'.This means that the owner may not change it in accordance with what was agreed on with the community.This represent a

big risk on the user side.

Code:

Listing 6: lazy1155.sol

```
1530      function _setTokenRoyalty(
1531          uint256 tokenId,
1532          address receiver,
1533          uint96 feeNumerator
1534      ) internal virtual {
1535          require(feeNumerator <= _feeDenominator(), "ERC2981: royalty fee
                  ↪ will exceed salePrice");
1536          require(receiver != address(0), "ERC2981: Invalid parameters");

1538          _tokenRoyaltyInfo[tokenId] = RoyaltyInfo(receiver, feeNumerator);
1539      }
```

Risk Level:

Likelihood – 5
Impact - 5

Recommendation:

We recommend to limit the fee value by adding a require statement.

Status - Acknowledged

B.2 LazyMint implementation [HIGH]

Description:

Lazy Minting is a way to defer to the normal minting until right before the NFT is sold. This way, buyers pay the minting fee after their NFT is sold, making NFT creation affordable and equitable for creators. The _lazymint function in this contract is the same as _mint.

Code:

Listing 7: lazy1155.sol

```
1547    function _lazyMint(
1548        address from,
1549        address to,
1550        uint256 id,
1551        uint256 amount,
1552        uint256 total,
1553        bytes memory data
1554    ) internal virtual {
1555        require(from != address(0), "ERC1155: mint to the zero address");

1557        address operator = _msgSender();
1558        uint256[] memory ids = _asSingletonArray(id);
1559        uint256[] memory amounts = _asSingletonArray(amount);

1561        _beforeTokenTransfer(operator, address(0), from, ids, amounts,
               ↪ data);

1563        _balances[id][msg.sender] += (total - amount);
1564        _balances[id][to] += amount;
1565        emit TransferSingle(operator, address(0), from, id, total);
1566        emit TransferSingle(operator, from, to, id, amount);

1568        //_afterTokenTransfer(operator, address(0), to, ids, amounts,
               ↪ data);

1570        _doSafeTransferAcceptanceCheck(
1571            operator,
1572            address(0),
1573            from,
1574            id,
1575            total,
```

```
1576                 data
1577             );
1578         }
```

Risk Level:

Likelihood – 4
Impact - 5

Recommendation:

We recommend to modify this function to go with the Lazy mint logic.

Status - Acknowledged

The Dev team Acknowledged the Risk because they need Lazymint for seperate emit event.

B.3 For Loop Over Dynamic Array [LOW]

Description:

When smart contracts are deployed or their associated functions are invoked, the execution of these operations always consumes a certain quantity of gas, according to the amount of computation required to accomplish them. Modifying an unknown-size array that grows in size over time can result in a Denial of Service attack. Simply by having an excessively huge array, users can exceed the gas limit, therefore preventing the transaction from ever succeeding.

Code:

Listing 8: lazy1155.sol

```
973     function balanceOfBatch(address[] memory accounts, uint256[] memory
            ↪ ids)
974         public
975         view
```

```
976         virtual
977         override
978         returns (uint256[] memory)
979     {
980         require(
981             accounts.length == ids.length,
982             "ERC1155: accounts and ids length mismatch"
983         );
985         uint256[] memory batchBalances = new uint256[](accounts.length);
987         for (uint256 i = 0; i < accounts.length; ++i) {
988             batchBalances[i] = balanceOf(accounts[i], ids[i]);
989         }
991         return batchBalances;
992     }
```

Listing 9: lazy1155.sol

```
1109    function _safeBatchTransferFrom(
1110        address from,
1111        address to,
1112        uint256[] memory ids,
1113        uint256[] memory amounts,
1114        bytes memory data
1115    ) internal virtual {
1116        require(
1117            ids.length == amounts.length,
1118            "ERC1155: ids and amounts length mismatch"
1119        );
1120        require(to != address(0), "ERC1155: transfer to the zero address
            ↪ ");
1122        address operator = _msgSender();
```

```
1124        _beforeTokenTransfer(operator, from, to, ids, amounts, data);

1126        for (uint256 i = 0; i < ids.length; ++i) {
1127            uint256 id = ids[i];
1128            uint256 amount = amounts[i];

1130            uint256 fromBalance = _balances[id][from];
1131            require(
1132                fromBalance >= amount,
1133                "ERC1155: insufficient balance for transfer"
1134            );
1135            unchecked {
1136                _balances[id][from] = fromBalance - amount;
1137            }
1138            _balances[id][to] += amount;
1139        }
```

Listing 10: lazy1155.sol

```
1230    function _mintBatch(
1231        address to,
1232        uint256[] memory ids,
1233        uint256[] memory amounts,
1234        bytes memory data
1235    ) internal virtual {
1236        require(to != address(0), "ERC1155: mint to the zero address");
1237        require(
1238            ids.length == amounts.length,
1239            "ERC1155: ids and amounts length mismatch"
1240        );

1242        address operator = _msgSender();

1244        _beforeTokenTransfer(operator, address(0), to, ids, amounts, data
```

```
            ↪ );

1246        for (uint256 i = 0; i < ids.length; i++) {
1247            _balances[ids[i]][to] += amounts[i];
1248        }

1250        emit TransferBatch(operator, address(0), to, ids, amounts);

1252        //_afterTokenTransfer(operator, address(0), to, ids, amounts,
                ↪ data);

1254        _doSafeBatchTransferAcceptanceCheck(
1255            operator,
1256            address(0),
1257            to,
1258            ids,
1259            amounts,
1260            data
1261        );
1262    }
```

Listing 11: lazy1155.sol

```
1306    function _burnBatch(
1307        address from,
1308        uint256[] memory ids,
1309        uint256[] memory amounts
1310    ) internal virtual {
1311        require(from != address(0), "ERC1155: burn from the zero address
                ↪ ");
1312        require(
1313            ids.length == amounts.length,
1314            "ERC1155: ids and amounts length mismatch"
1315        );
```

```
1317        address operator = _msgSender();

1319        _beforeTokenTransfer(operator, from, address(0), ids, amounts,
                ↪ "");

1321        for (uint256 i = 0; i < ids.length; i++) {
1322            uint256 id = ids[i];
1323            uint256 amount = amounts[i];

1325            uint256 fromBalance = _balances[id][from];
1326            require(
1327                fromBalance >= amount,
1328                "ERC1155: burn amount exceeds balance"
1329            );
1330            unchecked {
1331                _balances[id][from] = fromBalance - amount;
1332            }
1333        }

1335        emit TransferBatch(operator, from, address(0), ids, amounts);

1337        //_afterTokenTransfer(operator, from, address(0), ids, amounts,
                ↪ "");
1338    }
```

Risk Level:

Likelihood – 2
Impact - 2

Recommendation:

Avoid actions that involve looping across the entire data structure. If you really must loop over an array, arrange for it to consume many blocs and thus multiple transactions.

Status - Acknowledged

The Dev team Acknowledged the Risk ; For loop dynamic array is a Standard function so they haven't change it, but they restricted in UIUX.

B.4 Floating Pragma [LOW]

Description:

The contract makes use of the floating-point pragma 0.8.0. Contracts should be deployed using the same compiler version and flags that were used during the testing process.Locking the pragma helps ensure that contracts are not unintentionally deployed using another pragma, such as an obsolete version, that may introduce issues in the contract system.

Code:

Listing 12: lazy1155.sol

```
6 pragma solidity ^0.8.0;
```

Risk Level:

Likelihood – 1
Impact - 2

Status - Acknowledged

Recommendation:

Consider locking the pragma version. It is advised that floating pragma should not be used in production. Both truffle-config.js and hardhat.config.js support locking the pragma version.

C ERC1967Upgrade.sol

C.1 Floating Pragma [LOW]

Description:

The contract makes use of the floating-point pragma 0.8.2. Contracts should be deployed using the same compiler version and flags that were used during the testing process.Locking the pragma helps ensure that contracts are not unintentionally deployed using another pragma, such as an obsolete version, that may introduce issues in the contract system.

Code:

Listing 13: ERC1967Upgrade.sol

```
3 pragma solidity ^0.8.2;
```

Risk Level:

Likelihood – 1
Impact - 2

Status - Acknowledged

Recommendation:

Consider locking the pragma version. It is advised that floating pragma should not be used in production. Both truffle-config.js and hardhat.config.js support locking the pragma version.

D Address.sol

D.1 Floating Pragma [LOW]

Description:

The contract makes use of the floating-point pragma 0.8.0. Contracts should be deployed using the same compiler version and flags that were used during the testing process.Locking the pragma helps ensure that contracts are not unintentionally deployed using another pragma, such as an obsolete version, that may introduce issues in the contract system.

Code:

Listing 14: Address.sol

```
3 pragma solidity ^0.8.0;
```

Risk Level:

Likelihood – 1
Impact - 2

Status - Acknowledged

Recommendation:

Consider locking the pragma version. It is advised that floating pragma should not be used in production. Both truffle-config.js and hardhat.config.js support locking the pragma version.

E StorageSlot.sol

E.1 Floating Pragma [LOW]

Description:

The contract makes use of the floating-point pragma 0.8.0. Contracts should be deployed using the same compiler version and flags that were used during the testing process.Locking the pragma helps ensure that contracts are not unintentionally deployed using another pragma, such as an obsolete version, that may introduce issues in the contract system.

Code:

Listing 15: StorageSlot.sol

```
3 pragma solidity ^0.8.0;
```

Risk Level:

Likelihood – 1
Impact - 2

Status - Acknowledged

Recommendation:

Consider locking the pragma version. It is advised that floating pragma should not be used in production. Both truffle-config.js and hardhat.config.js support locking the pragma version.

F Proxy.sol

F.1 Floating Pragma [LOW]

Description:

The contract makes use of the floating-point pragma 0.8.0. Contracts should be deployed using the same compiler version and flags that were used during the testing process.Locking the pragma helps ensure that contracts are not unintentionally deployed using another pragma, such as an obsolete version, that may introduce issues in the contract system.

Code:

Listing 16: Proxy.sol

```
3 pragma solidity ^0.8.0;
```

Risk Level:

Likelihood – 1
Impact - 2

Status - Acknowledged

Recommendation:

Consider locking the pragma version. It is advised that floating pragma should not be used in production. Both truffle-config.js and hardhat.config.js support locking the pragma version.

G TransparentUpgradeableProxy.sol

G.1 Floating Pragma [LOW]

Description:

The contract makes use of the floating-point pragma 0.8.0. Contracts should be deployed using the same compiler version and flags that were used during the testing process.Locking the pragma helps ensure that contracts are not unintentionally deployed using another pragma, such as an obsolete version, that may introduce issues in the contract system.

Code:

Listing 17: TransparentUpgradeableProxy.sol

```
3 pragma solidity ^0.8.0;
```

Risk Level:

Likelihood – 1
Impact - 2

Status - Acknowledged

Recommendation:

Consider locking the pragma version. It is advised that floating pragma should not be used in production. Both truffle-config.js and hardhat.config.js support locking the pragma version.

H ERC1967Proxy.sol

H.1 Floating Pragma [LOW]

Description:

The contract makes use of the floating-point pragma 0.8.0. Contracts should be deployed using the same compiler version and flags that were used during the testing process.Locking the pragma helps ensure that contracts are not unintentionally deployed using another pragma, such as an obsolete version, that may introduce issues in the contract system.

Code:

Listing 18: ERC1967Proxy.sol

```
3 pragma solidity ^0.8.0;
```

Risk Level:

Likelihood – 1
Impact - 2

Status - Acknowledged

Recommendation:

Consider locking the pragma version. It is advised that floating pragma should not be used in production. Both truffle-config.js and hardhat.config.js support locking the pragma version.

I IBeacon.sol

I.1 Floating Pragma [LOW]

Description:

The contract makes use of the floating-point pragma 0.8.0. Contracts should be deployed using the same compiler version and flags that were used during the testing process.Locking the pragma helps ensure that contracts are not unintentionally deployed using another pragma, such as an obsolete version, that may introduce issues in the contract system.

Code:

Listing 19: IBeacon.sol

```
3 pragma solidity ^0.8.0;
```

Risk Level:

Likelihood – 1
Impact - 2

Status - Acknowledged

Recommendation:

Consider locking the pragma version. It is advised that floating pragma should not be used in production. Both truffle-config.js and hardhat.config.js support locking the pragma version.

4 Best Practices

BP.1 _upgradeToAndCallSecure function

Description:

In openzeppelin-contracts, the function _upgradeToAndCallSecure was renamed to _upgradeToAndCallUUPS, along with the change in security mechanism, the implementation was changed too We recommend to change this function with _upgradeToAndCallUUPS

Code:

Listing 20: ERC1967Upgrade.sol

```
76      function _upgradeToAndCallSecure(address newImplementation, bytes
            ↪ memory data, bool forceCall) internal {
77          address oldImplementation = _getImplementation();
78
79          // Initial upgrade and setup call
80          _setImplementation(newImplementation);
81          if (data.length > 0 || forceCall) {
82              Address.functionDelegateCall(newImplementation, data);
83          }
84
85          // Perform rollback test if not already in progress
86          StorageSlot.BooleanSlot storage rollbackTesting = StorageSlot.
                ↪ getBooleanSlot(_ROLLBACK_SLOT);
87          if (!rollbackTesting.value) {
88              // Trigger rollback using upgradeTo from the new
                    ↪ implementation
89              rollbackTesting.value = true;
90              Address.functionDelegateCall(
91                  newImplementation,
92                  abi.encodeWithSignature(
93                      "upgradeTo(address)",
```

```
94                      oldImplementation
95                  )
96              );
97              rollbackTesting.value = false;
98              // Check rollback was effective
99              require(oldImplementation == _getImplementation(), "
                    ↪ ERC1967Upgrade: upgrade breaks further upgrades");
100             // Finally reset to the new implementation and log the
                    ↪ upgrade
101             _setImplementation(newImplementation);
102             emit Upgraded(newImplementation);
103         }
104     }
```

BP.2 Public Function Can Be Called External

Description:

Functions with a public scope that are not called inside the contract should be declared external to reduce the gas fees

Code:

Listing 21: Lazy721.sol

```
1673    function mint(string memory ipfsmetadata, address from, address to,
            ↪ uint royal, uint256 id_, string memory status) public {
1674        require(msg.sender == owner, "Public Mint Not Available");
1675        if(keccak256(abi.encodePacked((status))) ==
1676            keccak256(abi.encodePacked(("lazy")))){
1677                _lazyMint(from, to, id_);
1678        }
1679        else{
1680            _safeMint(to, id_);
1681        }z
```

```
1682          _setTokenURI(id_, ipfsmetadata);
1683          _setTokenRoyalty(id_, from, uint96(royal.div(1e16)));
1684          _creator[id_] = from;
1685          _royal[id_]=royal;
1686      }
```

Listing 22: Lazy721.sol

```
1687      function mintBatch(string[] memory ipfsmetadata, address[] memory
              ↪ from, address[] memory to, uint256 count, uint royal) public {
1688          require(msg.sender == owner, "Public Mint Not Available");
1689          for (uint256 i = 0; i < ipfsmetadata.length; i++) {
1690              count = count.add(1);
1691              uint256 id_ = count.add(block.timestamp);
1692              _safeMint(to[i], id_);
1693              _setTokenURI(id_, ipfsmetadata[i]);
1694              _creator[id_] = from[i];
1695              _royal[id_]=royal;
1696          }
1697      }
```

Listing 23: Lazy721.sol

```
1698      function getCreatorsAndRoyalty(uint256 tokenid) public view returns(
              ↪ address, uint256) {
1699          return (_creator[tokenid], _royal[tokenid]);
1700      }
```

Listing 24: Lazy721.sol

```
1701      function TransferNFT(address to, uint256 tokenid) public {
1702          safeTransferFrom(msg.sender, to, tokenid);
1703      }
```

Listing 25: Lazy721.sol

```
1704      function burnNFT(uint256 tokenId) public{
1705          require(ownerOf(tokenId) == msg.sender, "Not a NFT Owner");
```

```
        _burn(tokenId);
    }
```

Listing 26: Lazy721.sol

```
    function changeCollectionOwner(address to) public {
        transferOwnership(payable(to));
    }
```

Listing 27: Lazy1155.sol

```
    function mint(string memory ipfsmetadata, address from, address to,
        ↪ uint supply, uint total, uint royal, uint256 id_) public {
        require(msg.sender == owner, "Public Mint Not Available");
        if(supply == 0){
            _mint(from, to, id_, total, "");
        }
        else{
            _lazyMint(from, to, id_, supply, total, "");
        }
        _setTokenURI(id_, ipfsmetadata);
        _setTokenRoyalty(id_, from, uint96(royal.div(1e16)));
        _creator[id_] = from;
        _royal[id_]=royal;
    }
```

Listing 28: Lazy1155.sol

```
    function getCreatorsAndRoyalty(uint256 tokenid) public view returns(
        ↪ address, uint256) {
        return (_creator[tokenid], _royal[tokenid]);
    }
```

Listing 29: Lazy1155.sol

```
    function TransferNFT(address to, uint256 tokenid, uint256 count)
        ↪ public {
        safeTransferFrom(msg.sender, to, tokenid, count, '');
```

```
1632     }
```

Listing 30: Lazy1155.sol

```
1633     function _openUri(bool open) public onlyOwner{
1634         openUri = open;
1635     }
```

Listing 31: Lazy1155.sol

```
1635     function burnNFT(uint256 tokenId, uint256 amount) public{
1636         require(balanceOf(msg.sender, tokenId) == amount, "Not a Owner or
             ↪  balance Mismatch");
1637         _burn(msg.sender, tokenId, amount);
1638     }
```

Listing 32: Lazy1155.sol

```
1640     function changeCollectionOwner(address to) public {
1641         transferOwnership(payable(to));
1642     }
```

BP.3 Presence of unused code

Description:

The program contains code that is not essential for execution, i.e, makes no state changes and has no side effects that alter data or control flow, such that removal of the code would have no impact on functionality or correctness , Context._msgData() is never used and should be removed

Code:

Listing 33: Lazy721.sol

```
493     function _msgData() internal view virtual returns (bytes calldata) {
494         return msg.data;
495     }
```

Listing 34: Lazy1155.sol

```
23      function _msgData() internal view virtual returns (bytes calldata) {
24          return msg.data;
25      }
```

5 Static Analysis (Slither)

Description:

Block Hat expanded the coverage of the specific contract areas using automated testing methodologies. Slither, a Solidity static analysis framework, was one of the tools used. Slither was run on all-scoped contracts in both text and binary formats. This tool can be used to test mathematical relationships between Solidity instances statically and variables that allow for the detection of errors or inconsistent usage of the contracts' APIs throughout the entire codebase.

Results:

```
ERC1967Upgrade._upgradeToAndCall(address,bytes,bool) (ERC1967Upgrade.sol
   ↪ #63-69) ignores return value by Address.functionDelegateCall(
   ↪ newImplementation,data) (ERC1967Upgrade.sol#67)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (
   ↪ ERC1967Upgrade.sol#76-104) ignores return value by Address.
   ↪ functionDelegateCall(newImplementation,data) (ERC1967Upgrade.sol
   ↪ #82)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (
   ↪ ERC1967Upgrade.sol#76-104) ignores return value by Address.
   ↪ functionDelegateCall(newImplementation,abi.encodeWithSignature(
   ↪ upgradeTo(address),oldImplementation)) (ERC1967Upgrade.sol#90-96)
ERC1967Upgrade._upgradeBeaconToAndCall(address,bytes,bool) (
   ↪ ERC1967Upgrade.sol#112-118) ignores return value by Address.
   ↪ functionDelegateCall(IBeacon(newBeacon).implementation(),data) (
   ↪ ERC1967Upgrade.sol#116)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #unused-return

Modifier TransparentUpgradeableProxy.ifAdmin() (
   ↪ TransparentUpgradeableProxy.sol#41-47) does not always execute _;
   ↪  or revertReference: https://github.com/crytic/slither/wiki/
```

```
    ↪ Detector-Documentation#incorrect-modifier

Reentrancy in ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool)
    ↪  (ERC1967Upgrade.sol#76-104):
        External calls:
        - Address.functionDelegateCall(newImplementation,data) (
            ↪ ERC1967Upgrade.sol#82)
        - Address.functionDelegateCall(newImplementation,abi.
            ↪ encodeWithSignature(upgradeTo(address),oldImplementation))
            ↪  (ERC1967Upgrade.sol#90-96)
        Event emitted after the call(s):
        - Upgraded(newImplementation) (ERC1967Upgrade.sol#102)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #reentrancy-vulnerabilities-3

Address.isContract(address) (Address.sol#26-35) uses assembly
        - INLINE ASM (Address.sol#33)
Address._verifyCallResult(bool,bytes,string) (Address.sol#171-188) uses
    ↪ assembly
        - INLINE ASM (Address.sol#180-183)
Proxy._delegate(address) (Proxy.sol#21-41) uses assembly
        - INLINE ASM (Proxy.sol#23-40)
StorageSlot.getAddressSlot(bytes32) (StorageSlot.sol#51-55) uses
    ↪ assembly
        - INLINE ASM (StorageSlot.sol#52-54)
StorageSlot.getBooleanSlot(bytes32) (StorageSlot.sol#60-64) uses
    ↪ assembly
        - INLINE ASM (StorageSlot.sol#61-63)
StorageSlot.getBytes32Slot(bytes32) (StorageSlot.sol#69-73) uses
    ↪ assembly
        - INLINE ASM (StorageSlot.sol#70-72)
StorageSlot.getUint256Slot(bytes32) (StorageSlot.sol#78-82) uses
    ↪ assembly
        - INLINE ASM (StorageSlot.sol#79-81)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #assembly-usage

Different versions of Solidity are used:
        - Version used: ['^0.8.0', '^0.8.2']
        - ^0.8.0 (Address.sol#3)
        - ^0.8.0 (ERC1967Proxy.sol#3)
        - ^0.8.2 (ERC1967Upgrade.sol#3)
        - ^0.8.0 (IBeacon.sol#3)
        - ^0.8.0 (Proxy.sol#3)
        - ^0.8.0 (StorageSlot.sol#3)
        - ^0.8.0 (TransparentUpgradeableProxy.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #different-pragma-directives-are-used

Address.functionCall(address,bytes) (Address.sol#79-81) is never used
    ↪ and should be removed
Address.functionCall(address,bytes,string) (Address.sol#89-91) is never
    ↪ used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (Address.sol
    ↪ #104-106) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (Address.sol
    ↪ #114-121) is never used and should be removed
Address.functionStaticCall(address,bytes) (Address.sol#129-131) is never
    ↪  used and should be removed
Address.functionStaticCall(address,bytes,string) (Address.sol#139-145)
    ↪ is never used and should be removed
Address.sendValue(address,uint256) (Address.sol#53-59) is never used and
    ↪  should be removed
ERC1967Upgrade._getBeacon() (ERC1967Upgrade.sol#171-173) is never used
    ↪ and should be removed
ERC1967Upgrade._setBeacon(address) (ERC1967Upgrade.sol#178-188) is never
    ↪  used and should be removed
```

```
ERC1967Upgrade._upgradeBeaconToAndCall(address,bytes,bool) (
   ↪ ERC1967Upgrade.sol#112-118) is never used and should be removed
ERC1967Upgrade._upgradeTo(address) (ERC1967Upgrade.sol#53-56) is never
   ↪ used and should be removed
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (
   ↪ ERC1967Upgrade.sol#76-104) is never used and should be removed
StorageSlot.getBooleanSlot(bytes32) (StorageSlot.sol#60-64) is never
   ↪ used and should be removed
StorageSlot.getBytes32Slot(bytes32) (StorageSlot.sol#69-73) is never
   ↪ used and should be removed
StorageSlot.getUint256Slot(bytes32) (StorageSlot.sol#78-82) is never
   ↪ used and should be removed
TransparentUpgradeableProxy._admin() (TransparentUpgradeableProxy.sol
   ↪ #109-111) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #dead-code

Pragma version^0.8.0 (Address.sol#3) allows old versions
Pragma version^0.8.0 (ERC1967Proxy.sol#3) allows old versions
Pragma version^0.8.2 (ERC1967Upgrade.sol#3) allows old versions
Pragma version^0.8.0 (IBeacon.sol#3) allows old versions
Pragma version^0.8.0 (Proxy.sol#3) allows old versions
Pragma version^0.8.0 (StorageSlot.sol#3) allows old versions
Pragma version^0.8.0 (TransparentUpgradeableProxy.sol#3) allows old
   ↪ versions
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (Address.sol#53-59)
   ↪ :
       - (success) = recipient.call{value: amount}() (Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,
   ↪ string) (Address.sol#114-121):
```

```
        - (success,returndata) = target.call{value: value}(data) (Address
            ↪ .sol#119)
Low level call in Address.functionStaticCall(address,bytes,string) (
    ↪ Address.sol#139-145):
        - (success,returndata) = target.staticcall(data) (Address.sol
            ↪ #143)
Low level call in Address.functionDelegateCall(address,bytes,string) (
    ↪ Address.sol#163-169):
        - (success,returndata) = target.delegatecall(data) (Address.sol
            ↪ #167)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #low-level-calls

TransparentUpgradeableProxy (TransparentUpgradeableProxy.sol#28-120)
    ↪ should inherit from IBeacon (IBeacon.sol#8-16)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #missing-inheritance

Address.isContract(address) (Address.sol#26-35) uses assembly
        - INLINE ASM (Address.sol#33)
Address._verifyCallResult(bool,bytes,string) (Address.sol#171-188) uses
    ↪ assembly
        - INLINE ASM (Address.sol#180-183)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #assembly-usage

Address._verifyCallResult(bool,bytes,string) (Address.sol#171-188) is
    ↪ never used and should be removed
Address.functionCall(address,bytes) (Address.sol#79-81) is never used
    ↪ and should be removed
Address.functionCall(address,bytes,string) (Address.sol#89-91) is never
    ↪ used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (Address.sol
    ↪ #104-106) is never used and should be removed
```

```
Address.functionCallWithValue(address,bytes,uint256,string) (Address.sol
    ↪ #114-121) is never used and should be removed
Address.functionDelegateCall(address,bytes) (Address.sol#153-155) is
    ↪ never used and should be removed
Address.functionDelegateCall(address,bytes,string) (Address.sol#163-169)
    ↪  is never used and should be removed
Address.functionStaticCall(address,bytes) (Address.sol#129-131) is never
    ↪  used and should be removed
Address.functionStaticCall(address,bytes,string) (Address.sol#139-145)
    ↪ is never used and should be removed
Address.isContract(address) (Address.sol#26-35) is never used and should
    ↪  be removed
Address.sendValue(address,uint256) (Address.sol#53-59) is never used and
    ↪  should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #dead-code

Pragma version^0.8.0 (Address.sol#3) allows old versions
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (Address.sol#53-59)
    ↪ :
        - (success) = recipient.call{value: amount}() (Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,
    ↪ string) (Address.sol#114-121):
        - (success,returndata) = target.call{value: value}(data) (Address
            ↪ .sol#119)
Low level call in Address.functionStaticCall(address,bytes,string) (
    ↪ Address.sol#139-145):
        - (success,returndata) = target.staticcall(data) (Address.sol
            ↪ #143)
```

```
Low level call in Address.functionDelegateCall(address,bytes,string) (
    ↪ Address.sol#163-169):
        - (success,returndata) = target.delegatecall(data) (Address.sol
            ↪ #167)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #low-level-calls

ERC721._checkOnERC721Received(address,address,uint256,bytes) (Lazy721.
    ↪ sol#1436-1467) ignores return value by IERC721Receiver(to).
    ↪ onERC721Received(_msgSender(),from,tokenId,data) (Lazy721.sol
    ↪ #1443-1463)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #unused-return

Owned.constructor(address)._OwnerAddress (Lazy721.sol#957) lacks a zero-
    ↪ check on :
                - owner = address(_OwnerAddress) (Lazy721.sol#958)
Owned.transferOwnership(address)._newOwner (Lazy721.sol#966) lacks a
    ↪ zero-check on :
                - owner = _newOwner (Lazy721.sol#967)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #missing-zero-address-validation

ERC721._checkOnERC721Received(address,address,uint256,bytes) (Lazy721.
    ↪ sol#1436-1467) has external calls inside a loop: IERC721Receiver(
    ↪ to).onERC721Received(_msgSender(),from,tokenId,data) (Lazy721.sol
    ↪ #1443-1463)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ /#calls-inside-a-loop

Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).
    ↪ retval (Lazy721.sol#1450)' in ERC721._checkOnERC721Received(
    ↪ address,address,uint256,bytes) (Lazy721.sol#1436-1467)
    ↪ potentially used before declaration: retval == IERC721Receiver.
```

```
    ↪ onERC721Received.selector (Lazy721.sol#1451)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).
    ↪ reason (Lazy721.sol#1452)' in ERC721._checkOnERC721Received(
    ↪ address,address,uint256,bytes) (Lazy721.sol#1436-1467)
    ↪ potentially used before declaration: reason.length == 0 (Lazy721.
    ↪ sol#1453)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).
    ↪ reason (Lazy721.sol#1452)' in ERC721._checkOnERC721Received(
    ↪ address,address,uint256,bytes) (Lazy721.sol#1436-1467)
    ↪ potentially used before declaration: revert(uint256,uint256)(32 +
    ↪  reason,mload(uint256)(reason)) (Lazy721.sol#1460)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #pre-declaration-usage-of-local-variables

Reentrancy in Lazy721.mint(string,address,address,uint256,uint256,string
    ↪ ) (Lazy721.sol#1673-1686):
        External calls:
        - _safeMint(to,id_) (Lazy721.sol#1680)
                - IERC721Receiver(to).onERC721Received(_msgSender(),from,
                    ↪ tokenId,data) (Lazy721.sol#1443-1463)
        State variables written after the call(s):
        - _creator[id_] = from (Lazy721.sol#1684)
        - _royal[id_] = royal (Lazy721.sol#1685)
        - _setTokenRoyalty(id_,from,uint96(royal.div(1e16))) (Lazy721.sol
            ↪ #1683)
                - _tokenRoyaltyInfo[tokenId] = RoyaltyInfo(receiver,
                    ↪ feeNumerator) (Lazy721.sol#1575)
        - _setTokenURI(id_,ipfsmetadata) (Lazy721.sol#1682)
                - _tokenURIs[tokenId] = _tokenURI (Lazy721.sol#1639)
Reentrancy in Lazy721.mintBatch(string[],address[],address[],uint256,
    ↪ uint256) (Lazy721.sol#1687-1697):
        External calls:
        - _safeMint(to[i],id_) (Lazy721.sol#1692)
```

```
                - IERC721Receiver(to).onERC721Received(_msgSender(),from,
                    ↪ tokenId,data) (Lazy721.sol#1443-1463)
        State variables written after the call(s):
        - _creator[id_] = from[i] (Lazy721.sol#1694)
        - _royal[id_] = royal (Lazy721.sol#1695)
        - _setTokenURI(id_,ipfsmetadata[i]) (Lazy721.sol#1693)
                - _tokenURIs[tokenId] = _tokenURI (Lazy721.sol#1639)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #reentrancy-vulnerabilities-2

Address.verifyCallResult(bool,bytes,string) (Lazy721.sol#730-750) uses
    ↪ assembly
        - INLINE ASM (Lazy721.sol#742-745)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (Lazy721.
    ↪ sol#1436-1467) uses assembly
        - INLINE ASM (Lazy721.sol#1459-1461)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #assembly-usage

ERC721.tokenURI(uint256) (Lazy721.sol#1073-1088) compares to a boolean
    ↪ constant:
        -require(bool,string)(openUri == true,Admin Not Approved) (
            ↪ Lazy721.sol#1080)
ERC721URIStorage.tokenURI(uint256) (Lazy721.sol#1600-1622) compares to a
    ↪  boolean constant:
        -require(bool,string)(openUri == true,Admin Not Approved) (
            ↪ Lazy721.sol#1607)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #boolean-equality

Address.functionCall(address,bytes) (Lazy721.sol#583-588) is never used
    ↪ and should be removed
Address.functionCall(address,bytes,string) (Lazy721.sol#596-602) is
    ↪ never used and should be removed
```

```
Address.functionCallWithValue(address,bytes,uint256) (Lazy721.sol
   ↪ #615-627) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (Lazy721.sol
   ↪ #635-651) is never used and should be removed
Address.functionDelegateCall(address,bytes) (Lazy721.sol#695-705) is
   ↪ never used and should be removed
Address.functionDelegateCall(address,bytes,string) (Lazy721.sol#713-722)
   ↪  is never used and should be removed
Address.functionStaticCall(address,bytes) (Lazy721.sol#659-670) is never
   ↪  used and should be removed
Address.functionStaticCall(address,bytes,string) (Lazy721.sol#678-687)
   ↪ is never used and should be removed
Address.sendValue(address,uint256) (Lazy721.sol#552-563) is never used
   ↪ and should be removed
Address.verifyCallResult(bool,bytes,string) (Lazy721.sol#730-750) is
   ↪ never used and should be removed
Context._msgData() (Lazy721.sol#493-495) is never used and should be
   ↪ removed
ERC721._deleteDefaultRoyalty() (Lazy721.sol#1555-1557) is never used and
   ↪  should be removed
ERC721._resetTokenRoyalty(uint256) (Lazy721.sol#1581-1583) is never used
   ↪  and should be removed
ERC721._setDefaultRoyalty(address,uint96) (Lazy721.sol#1545-1550) is
   ↪ never used and should be removed
SafeMath.div(uint256,uint256,string) (Lazy721.sol#440-449) is never used
   ↪  and should be removed
SafeMath.mod(uint256,uint256) (Lazy721.sol#400-402) is never used and
   ↪ should be removed
SafeMath.mod(uint256,uint256,string) (Lazy721.sol#466-475) is never used
   ↪  and should be removed
SafeMath.mul(uint256,uint256) (Lazy721.sol#370-372) is never used and
   ↪ should be removed
SafeMath.sub(uint256,uint256) (Lazy721.sol#356-358) is never used and
   ↪ should be removed
```

```
SafeMath.sub(uint256,uint256,string) (Lazy721.sol#417-426) is never used
   ↪  and should be removed
SafeMath.tryAdd(uint256,uint256) (Lazy721.sol#251-261) is never used and
   ↪  should be removed
SafeMath.tryDiv(uint256,uint256) (Lazy721.sol#305-314) is never used and
   ↪  should be removed
SafeMath.tryMod(uint256,uint256) (Lazy721.sol#321-330) is never used and
   ↪  should be removed
SafeMath.tryMul(uint256,uint256) (Lazy721.sol#284-298) is never used and
   ↪  should be removed
SafeMath.trySub(uint256,uint256) (Lazy721.sol#268-277) is never used and
   ↪  should be removed
Strings.toHexString(address) (Lazy721.sol#240-242) is never used and
   ↪ should be removed
Strings.toHexString(uint256) (Lazy721.sol#205-216) is never used and
   ↪ should be removed
Strings.toHexString(uint256,uint256) (Lazy721.sol#221-235) is never used
   ↪  and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #dead-code

Pragma version^0.8.0 (Lazy721.sol#14) allows old versions
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (Lazy721.sol
   ↪ #552-563):
       - (success) = recipient.call{value: amount}() (Lazy721.sol#558)
Low level call in Address.functionCallWithValue(address,bytes,uint256,
   ↪ string) (Lazy721.sol#635-651):
       - (success,returndata) = target.call{value: value}(data) (Lazy721
          ↪ .sol#647-649)
```

```
Low level call in Address.functionStaticCall(address,bytes,string) (
    ↪ Lazy721.sol#678-687):
        - (success,returndata) = target.staticcall(data) (Lazy721.sol
            ↪ #685)
Low level call in Address.functionDelegateCall(address,bytes,string) (
    ↪ Lazy721.sol#713-722):
        - (success,returndata) = target.delegatecall(data) (Lazy721.sol
            ↪ #720)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #low-level-calls

Parameter Owned.transferOwnership(address)._newOwner (Lazy721.sol#966)
    ↪ is not in mixedCase
Parameter ERC721.royaltyInfo(uint256,uint256)._tokenId (Lazy721.sol
    ↪ #1516) is not in mixedCase
Parameter ERC721.royaltyInfo(uint256,uint256)._salePrice (Lazy721.sol
    ↪ #1516) is not in mixedCase
Function ERC721URIStorage._openUri(bool) (Lazy721.sol#1654-1656) is not
    ↪ in mixedCase
Function Lazy721.TransferNFT(address,uint256) (Lazy721.sol#1701-1703) is
    ↪  not in mixedCase
Variable Lazy721._tid (Lazy721.sol#1662) is not in mixedCase
Variable Lazy721._creator (Lazy721.sol#1663) is not in mixedCase
Variable Lazy721._royal (Lazy721.sol#1664) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #conformance-to-solidity-naming-conventions

Lazy721._tid (Lazy721.sol#1662) should be constant
Lazy721.totalmint (Lazy721.sol#1661) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #state-variables-that-could-be-declared-constant

balanceOf(address) should be declared external:
        - ERC721.balanceOf(address) (Lazy721.sol#1027-1039)
```

```
name() should be declared external:
        - ERC721.name() (Lazy721.sol#1059-1061)
symbol() should be declared external:
        - ERC721.symbol() (Lazy721.sol#1066-1068)
approve(address,uint256) should be declared external:
        - ERC721.approve(address,uint256) (Lazy721.sol#1102-1112)
setApprovalForAll(address,bool) should be declared external:
        - ERC721.setApprovalForAll(address,bool) (Lazy721.sol#1132-1138)
transferFrom(address,address,uint256) should be declared external:
        - ERC721.transferFrom(address,address,uint256) (Lazy721.sol
            ↪ #1156-1168)
royaltyInfo(uint256,uint256) should be declared external:
        - ERC721.royaltyInfo(uint256,uint256) (Lazy721.sol#1516-1526)
_openUri(bool) should be declared external:
        - ERC721URIStorage._openUri(bool) (Lazy721.sol#1654-1656)
mint(string,address,address,uint256,uint256,string) should be declared
    ↪ external:
        - Lazy721.mint(string,address,address,uint256,uint256,string) (
            ↪ Lazy721.sol#1673-1686)
getCreatorsAndRoyalty(uint256) should be declared external:
        - Lazy721.getCreatorsAndRoyalty(uint256) (Lazy721.sol#1698-1700)
TransferNFT(address,uint256) should be declared external:
        - Lazy721.TransferNFT(address,uint256) (Lazy721.sol#1701-1703)
burnNFT(uint256) should be declared external:
        - Lazy721.burnNFT(uint256) (Lazy721.sol#1704-1707)
changeCollectionOwner(address) should be declared external:
        - Lazy721.changeCollectionOwner(address) (Lazy721.sol#1708-1710)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #public-function-that-could-be-declared-external

ERC1967Upgrade._upgradeToAndCall(address,bytes,bool) (ERC1967Upgrade.sol
    ↪ #63-69) ignores return value by Address.functionDelegateCall(
    ↪ newImplementation,data) (ERC1967Upgrade.sol#67)
```

```
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (
    ↪ ERC1967Upgrade.sol#76-104) ignores return value by Address.
    ↪ functionDelegateCall(newImplementation,data) (ERC1967Upgrade.sol
    ↪ #82)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (
    ↪ ERC1967Upgrade.sol#76-104) ignores return value by Address.
    ↪ functionDelegateCall(newImplementation,abi.encodeWithSignature(
    ↪ upgradeTo(address),oldImplementation)) (ERC1967Upgrade.sol#90-96)
ERC1967Upgrade._upgradeBeaconToAndCall(address,bytes,bool) (
    ↪ ERC1967Upgrade.sol#112-118) ignores return value by Address.
    ↪ functionDelegateCall(IBeacon(newBeacon).implementation(),data) (
    ↪ ERC1967Upgrade.sol#116)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #unused-return

Reentrancy in ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool)
    ↪  (ERC1967Upgrade.sol#76-104):
        External calls:
        - Address.functionDelegateCall(newImplementation,data) (
            ↪ ERC1967Upgrade.sol#82)
        - Address.functionDelegateCall(newImplementation,abi.
            ↪ encodeWithSignature(upgradeTo(address),oldImplementation))
            ↪  (ERC1967Upgrade.sol#90-96)
        Event emitted after the call(s):
        - Upgraded(newImplementation) (ERC1967Upgrade.sol#102)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #reentrancy-vulnerabilities-3

Address.isContract(address) (Address.sol#26-35) uses assembly
        - INLINE ASM (Address.sol#33)
Address._verifyCallResult(bool,bytes,string) (Address.sol#171-188) uses
    ↪ assembly
        - INLINE ASM (Address.sol#180-183)
Proxy._delegate(address) (Proxy.sol#21-41) uses assembly
```

```
        - INLINE ASM (Proxy.sol#23-40)
StorageSlot.getAddressSlot(bytes32) (StorageSlot.sol#51-55) uses
    ↪ assembly
        - INLINE ASM (StorageSlot.sol#52-54)
StorageSlot.getBooleanSlot(bytes32) (StorageSlot.sol#60-64) uses
    ↪ assembly
        - INLINE ASM (StorageSlot.sol#61-63)
StorageSlot.getBytes32Slot(bytes32) (StorageSlot.sol#69-73) uses
    ↪ assembly
        - INLINE ASM (StorageSlot.sol#70-72)
StorageSlot.getUint256Slot(bytes32) (StorageSlot.sol#78-82) uses
    ↪ assembly
        - INLINE ASM (StorageSlot.sol#79-81)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #assembly-usage

Different versions of Solidity are used:
        - Version used: ['^0.8.0', '^0.8.2']
        - ^0.8.0 (Address.sol#3)
        - ^0.8.0 (ERC1967Proxy.sol#3)
        - ^0.8.2 (ERC1967Upgrade.sol#3)
        - ^0.8.0 (IBeacon.sol#3)
        - ^0.8.0 (Proxy.sol#3)
        - ^0.8.0 (StorageSlot.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #different-pragma-directives-are-used

Address.functionCall(address,bytes) (Address.sol#79-81) is never used
    ↪ and should be removed
Address.functionCall(address,bytes,string) (Address.sol#89-91) is never
    ↪ used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (Address.sol
    ↪ #104-106) is never used and should be removed
```

```
Address.functionCallWithValue(address,bytes,uint256,string) (Address.sol
   ↪ #114-121) is never used and should be removed
Address.functionStaticCall(address,bytes) (Address.sol#129-131) is never
   ↪  used and should be removed
Address.functionStaticCall(address,bytes,string) (Address.sol#139-145)
   ↪ is never used and should be removed
Address.sendValue(address,uint256) (Address.sol#53-59) is never used and
   ↪  should be removed
ERC1967Upgrade._changeAdmin(address) (ERC1967Upgrade.sol#152-155) is
   ↪ never used and should be removed
ERC1967Upgrade._getAdmin() (ERC1967Upgrade.sol#135-137) is never used
   ↪ and should be removed
ERC1967Upgrade._getBeacon() (ERC1967Upgrade.sol#171-173) is never used
   ↪ and should be removed
ERC1967Upgrade._setAdmin(address) (ERC1967Upgrade.sol#142-145) is never
   ↪ used and should be removed
ERC1967Upgrade._setBeacon(address) (ERC1967Upgrade.sol#178-188) is never
   ↪  used and should be removed
ERC1967Upgrade._upgradeBeaconToAndCall(address,bytes,bool) (
   ↪ ERC1967Upgrade.sol#112-118) is never used and should be removed
ERC1967Upgrade._upgradeTo(address) (ERC1967Upgrade.sol#53-56) is never
   ↪ used and should be removed
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (
   ↪ ERC1967Upgrade.sol#76-104) is never used and should be removed
StorageSlot.getBooleanSlot(bytes32) (StorageSlot.sol#60-64) is never
   ↪ used and should be removed
StorageSlot.getBytes32Slot(bytes32) (StorageSlot.sol#69-73) is never
   ↪ used and should be removed
StorageSlot.getUint256Slot(bytes32) (StorageSlot.sol#78-82) is never
   ↪ used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #dead-code

Pragma version^0.8.0 (Address.sol#3) allows old versions
```

```
Pragma version^0.8.0 (ERC1967Proxy.sol#3) allows old versions
Pragma version^0.8.2 (ERC1967Upgrade.sol#3) allows old versions
Pragma version^0.8.0 (IBeacon.sol#3) allows old versions
Pragma version^0.8.0 (Proxy.sol#3) allows old versions
Pragma version^0.8.0 (StorageSlot.sol#3) allows old versions
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (Address.sol#53-59)
    ↪ :
        - (success) = recipient.call{value: amount}() (Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,
    ↪ string) (Address.sol#114-121):
        - (success,returndata) = target.call{value: value}(data) (Address
            ↪ .sol#119)
Low level call in Address.functionStaticCall(address,bytes,string) (
    ↪ Address.sol#139-145):
        - (success,returndata) = target.staticcall(data) (Address.sol
            ↪ #143)
Low level call in Address.functionDelegateCall(address,bytes,string) (
    ↪ Address.sol#163-169):
        - (success,returndata) = target.delegatecall(data) (Address.sol
            ↪ #167)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #low-level-calls

ERC1967Upgrade._upgradeToAndCall(address,bytes,bool) (ERC1967Upgrade.sol
    ↪ #63-69) ignores return value by Address.functionDelegateCall(
    ↪ newImplementation,data) (ERC1967Upgrade.sol#67)
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (
    ↪ ERC1967Upgrade.sol#76-104) ignores return value by Address.
    ↪ functionDelegateCall(newImplementation,data) (ERC1967Upgrade.sol
    ↪ #82)
```

```
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (
    ↪ ERC1967Upgrade.sol#76-104) ignores return value by Address.
    ↪ functionDelegateCall(newImplementation,abi.encodeWithSignature(
    ↪ upgradeTo(address),oldImplementation)) (ERC1967Upgrade.sol#90-96)
ERC1967Upgrade._upgradeBeaconToAndCall(address,bytes,bool) (
    ↪ ERC1967Upgrade.sol#112-118) ignores return value by Address.
    ↪ functionDelegateCall(IBeacon(newBeacon).implementation(),data) (
    ↪ ERC1967Upgrade.sol#116)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #unused-return

Reentrancy in ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool)
    ↪  (ERC1967Upgrade.sol#76-104):
        External calls:
        - Address.functionDelegateCall(newImplementation,data) (
            ↪ ERC1967Upgrade.sol#82)
        - Address.functionDelegateCall(newImplementation,abi.
            ↪ encodeWithSignature(upgradeTo(address),oldImplementation))
            ↪  (ERC1967Upgrade.sol#90-96)
        Event emitted after the call(s):
        - Upgraded(newImplementation) (ERC1967Upgrade.sol#102)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #reentrancy-vulnerabilities-3

Address.isContract(address) (Address.sol#26-35) uses assembly
        - INLINE ASM (Address.sol#33)
Address._verifyCallResult(bool,bytes,string) (Address.sol#171-188) uses
    ↪ assembly
        - INLINE ASM (Address.sol#180-183)
StorageSlot.getAddressSlot(bytes32) (StorageSlot.sol#51-55) uses
    ↪ assembly
        - INLINE ASM (StorageSlot.sol#52-54)
StorageSlot.getBooleanSlot(bytes32) (StorageSlot.sol#60-64) uses
    ↪ assembly
```

```
        - INLINE ASM (StorageSlot.sol#61-63)
StorageSlot.getBytes32Slot(bytes32) (StorageSlot.sol#69-73) uses
    ↪ assembly
        - INLINE ASM (StorageSlot.sol#70-72)
StorageSlot.getUint256Slot(bytes32) (StorageSlot.sol#78-82) uses
    ↪ assembly
        - INLINE ASM (StorageSlot.sol#79-81)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #assembly-usage

Different versions of Solidity are used:
        - Version used: ['^0.8.0', '^0.8.2']
        - ^0.8.0 (Address.sol#3)
        - ^0.8.2 (ERC1967Upgrade.sol#3)
        - ^0.8.0 (IBeacon.sol#3)
        - ^0.8.0 (StorageSlot.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #different-pragma-directives-are-used

Address._verifyCallResult(bool,bytes,string) (Address.sol#171-188) is
    ↪ never used and should be removed
Address.functionCall(address,bytes) (Address.sol#79-81) is never used
    ↪ and should be removed
Address.functionCall(address,bytes,string) (Address.sol#89-91) is never
    ↪ used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (Address.sol
    ↪ #104-106) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (Address.sol
    ↪ #114-121) is never used and should be removed
Address.functionDelegateCall(address,bytes) (Address.sol#153-155) is
    ↪ never used and should be removed
Address.functionDelegateCall(address,bytes,string) (Address.sol#163-169)
    ↪  is never used and should be removed
```

```
Address.functionStaticCall(address,bytes) (Address.sol#129-131) is never
   ↪  used and should be removed
Address.functionStaticCall(address,bytes,string) (Address.sol#139-145)
   ↪ is never used and should be removed
Address.isContract(address) (Address.sol#26-35) is never used and should
   ↪  be removed
Address.sendValue(address,uint256) (Address.sol#53-59) is never used and
   ↪  should be removed
ERC1967Upgrade._changeAdmin(address) (ERC1967Upgrade.sol#152-155) is
   ↪ never used and should be removed
ERC1967Upgrade._getAdmin() (ERC1967Upgrade.sol#135-137) is never used
   ↪ and should be removed
ERC1967Upgrade._getBeacon() (ERC1967Upgrade.sol#171-173) is never used
   ↪ and should be removed
ERC1967Upgrade._getImplementation() (ERC1967Upgrade.sol#36-38) is never
   ↪ used and should be removed
ERC1967Upgrade._setAdmin(address) (ERC1967Upgrade.sol#142-145) is never
   ↪ used and should be removed
ERC1967Upgrade._setBeacon(address) (ERC1967Upgrade.sol#178-188) is never
   ↪  used and should be removed
ERC1967Upgrade._setImplementation(address) (ERC1967Upgrade.sol#43-46) is
   ↪  never used and should be removed
ERC1967Upgrade._upgradeBeaconToAndCall(address,bytes,bool) (
   ↪ ERC1967Upgrade.sol#112-118) is never used and should be removed
ERC1967Upgrade._upgradeTo(address) (ERC1967Upgrade.sol#53-56) is never
   ↪ used and should be removed
ERC1967Upgrade._upgradeToAndCall(address,bytes,bool) (ERC1967Upgrade.sol
   ↪ #63-69) is never used and should be removed
ERC1967Upgrade._upgradeToAndCallSecure(address,bytes,bool) (
   ↪ ERC1967Upgrade.sol#76-104) is never used and should be removed
StorageSlot.getAddressSlot(bytes32) (StorageSlot.sol#51-55) is never
   ↪ used and should be removed
StorageSlot.getBooleanSlot(bytes32) (StorageSlot.sol#60-64) is never
   ↪ used and should be removed
```

```
StorageSlot.getBytes32Slot(bytes32) (StorageSlot.sol#69-73) is never
   ↪ used and should be removed
StorageSlot.getUint256Slot(bytes32) (StorageSlot.sol#78-82) is never
   ↪ used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #dead-code

Pragma version^0.8.0 (Address.sol#3) allows old versions
Pragma version^0.8.2 (ERC1967Upgrade.sol#3) allows old versions
Pragma version^0.8.0 (IBeacon.sol#3) allows old versions
Pragma version^0.8.0 (StorageSlot.sol#3) allows old versions
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (Address.sol#53-59)
   ↪ :
       - (success) = recipient.call{value: amount}() (Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,
   ↪ string) (Address.sol#114-121):
       - (success,returndata) = target.call{value: value}(data) (Address
          ↪ .sol#119)
Low level call in Address.functionStaticCall(address,bytes,string) (
   ↪ Address.sol#139-145):
       - (success,returndata) = target.staticcall(data) (Address.sol
          ↪ #143)
Low level call in Address.functionDelegateCall(address,bytes,string) (
   ↪ Address.sol#163-169):
       - (success,returndata) = target.delegatecall(data) (Address.sol
          ↪ #167)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #low-level-calls

Pragma version^0.8.0 (IBeacon.sol#3) allows old versions
```

```
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #incorrect-versions-of-solidity

StorageSlot.getAddressSlot(bytes32) (StorageSlot.sol#51-55) uses
    ↪ assembly
        - INLINE ASM (StorageSlot.sol#52-54)
StorageSlot.getBooleanSlot(bytes32) (StorageSlot.sol#60-64) uses
    ↪ assembly
        - INLINE ASM (StorageSlot.sol#61-63)
StorageSlot.getBytes32Slot(bytes32) (StorageSlot.sol#69-73) uses
    ↪ assembly
        - INLINE ASM (StorageSlot.sol#70-72)
StorageSlot.getUint256Slot(bytes32) (StorageSlot.sol#78-82) uses
    ↪ assembly
        - INLINE ASM (StorageSlot.sol#79-81)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #assembly-usage

StorageSlot.getAddressSlot(bytes32) (StorageSlot.sol#51-55) is never
    ↪ used and should be removed
StorageSlot.getBooleanSlot(bytes32) (StorageSlot.sol#60-64) is never
    ↪ used and should be removed
StorageSlot.getBytes32Slot(bytes32) (StorageSlot.sol#69-73) is never
    ↪ used and should be removed
StorageSlot.getUint256Slot(bytes32) (StorageSlot.sol#78-82) is never
    ↪ used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #dead-code

Pragma version^0.8.0 (StorageSlot.sol#3) allows old versions
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #incorrect-versions-of-solidity
```

Conclusion:

Most of the vulnerabilities found by the analysis have already been addressed by the smart contract code review.

6 Conclusion

In this audit, we examined the design and implementation of NFT MP contract and discovered several issues of varying severity. Drchinp team addressed 0 issues raised in the initial report and implemented the necessary fixes, while classifying the rest as a risk with low-probability of occurrence. BlockHat' auditors advised Drchinp Team to maintain a high level of vigilance and to keep those findings in mind in order to avoid any future complications.

BLOCKHAT
SECURITY

For a Contract Audit, contact us at contact@blockhat.io