MythX

REPORT 62658015F862E80019DD7F23

Created	Sun Apr 24 2022 16:51:33 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	6265706d5ec4940334c82dc0

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
4e3b45c1-dbcd-4d88-8e41-e2b310354e72	NFT_flat.sol	13

Analysis 4e3b45c1-dbcd-4d88-8e41-e2b310354e72

MythX

Started	Sun Apr 24 2022 16:51:43 GMT+0000 (Coordinated Universal Time)
Finished	Sun Apr 24 2022 17:37:47 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Remythx
Main Source File	`NFT_flat.Sol`

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	13

ISSUES

LOW SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

`NFT_flat.sol`

Locations

```
// OpenZeppelin Contracts v4.4.1 (utils/Strings.sol)

pragma solidity ^0.8.0;

/**
```

LOW SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

`NFT_flat.sol`

Locations

```
// OpenZeppelin Contracts v4.4.1 (utils/Context.sol)

pragma solidity ^0.8.0;

/**
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.1"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

`NFT_flat.sol`

Locations

```
// OpenZeppelin Contracts (last updated v4.5.0) (utils/Address.sol)

pragma solidity ^0.8.1;

/**
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

`NFT_flat.sol`

Locations

```
// OpenZeppelin Contracts v4.4.1 (token/ERC721/IERC721Receiver.sol)

pragma solidity ^0.8.0;

/**
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

`NFT_flat.sol`

Locations

```
// OpenZeppelin Contracts v4.4.1 (utils/introspection/IERC165.sol)

pragma solidity ^0.8.0;

/**
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

`NFT_flat.sol`

Locations

```
// OpenZeppelin Contracts v4.4.1 (utils/introspection/ERC165.sol)

pragma solidity ^0.8.0;

```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

`NFT_flat.sol`

Locations

```
// OpenZeppelin Contracts v4.4.1 (token/ERC721/IERC721.sol)

pragma solidity ^0.8.0;

```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

`NFT_flat.sol`

Locations

```
// OpenZeppelin Contracts v4.4.1 (token/ERC721/extensions/IERC721Metadata.sol)

pragma solidity ^0.8.0;

```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

`NFT_flat.sol`

Locations

```
// OpenZeppelin Contracts (last updated v4.5.0) (token/ERC721/ERC721.sol)

pragma solidity ^0.8.0;

```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

NFT_flat.sol

Locations

```
// OpenZeppelin Contracts v4.4.1 (token/ERC721/extensions/ERC721URIStorage.sol)

pragma solidity ^0.8.0;

```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.4"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

NFT_flat.sol

Locations

```


pragma solidity ^0.8.4; //Similar to hardhat's version

```

LOW

SWC-107

A call to a user-supplied address is executed.

An external message call to an address specified by the caller is executed. Note that the callee account might contain arbitrary code and could re-enter any function within this contract. Reentering the contract in an intermediate state may lead to unexpected behaviour. Make sure that no state modifications are executed after this call and/or reentrancy guards are in place.

Source file

NFT_flat.sol

Locations

```
) private returns (bool) {
if (to.isContract()) {
try IERC721Receiver(to).onERC721Received(_msgSender(), from, tokenId, _data) returns (bytes4 retval) {
return retval == IERC721Receiver.onERC721Received.selector;
} catch (bytes memory reason) {
```

LOW

SWC-123

Requirement violation.

A requirement was violated in a nested call and the call was reverted as a result. Make sure valid inputs are provided to the nested call (for instance, via passed arguments).

Source file

`NFT_flat.sol`

Locations

```
) private returns (bool) {
if (to.isContract()) {
try IERC721Receiver(to).onERC721Received(_msgSender(), from, tokenId, _data) returns (bytes4 retval) {
return retval == IERC721Receiver.onERC721Received.selector;
} catch (bytes memory reason) {
```

Source file

`NFT_flat.sol`

Locations

```


contract NFT is ERC721URIStorage {
uint public tokenCount;
constructor() ERC721("Healthcare Products NFTs", "HCP"){} //The constructor of openzeppelin smart contract is used
function mint(string memory _tokenURI) external returns(uint) { //tokenURI is the metadata of the NFT (IPFS hash)
tokenCount ++;
_safeMint(msg.sender, tokenCount);
_setTokenURI(tokenCount, _tokenURI);
return(tokenCount);
}
}
```