Jul 15, 2024

ZAN

Security Assessment MintSwap404NFT

—

Professional Service

Table of Contents

1. Overview

1.1. Executive Summary

1.2. Project Summary

1.3. Assessment Summary

1.4. Assessment Scope

2. Checklist

3. Findings

H-01: Return Value of 0 From ecrecover Is Not Checked

H-02: Missing the Returning of the Residual msg.value

M-03: Centralization Risk in Off-Chain Benefit Assessments

I-04: Use the Checks-Effects-Interactions Pattern to Update the State

I-05: No Check of Address Params with Zero Address

I-06: Function Visibility Can Be External

I-07: Parameters Should Be Declared as Calldata

I-08: Floating Pragma

4. Disclaimer

5. Appendix

1. Overview

1.1. Executive Summary

MintSwap404NFT is a project about ERC-404 tokens. Users can earn ETH rewards by staking specific ERC-404 tokens, and receive ERC-404 token rewards through liquidity mining. This report has been prepared for NFTScan to discover issues and vulnerabilities in the source code of this project as well as any contract dependencies that were not part of an officially recognized library.

Conducted by Static Analysis, Formal Verification and Manual Review, we have identified 2 High, 1 Medium and 5 Informational issues in commit 078b308.

The project team resolved security vulnerability described in H-01 and all the informational issues in commit 1df4c9e. About issues described in H-02 and M-03, the project team has confirmed that the implementation in commit 078b308 complies with the project design and decided to keep no change.

1.2. Project Summary

Project Name	MintSwap404NFT
Platform	Mint
Language	Solidity
Codebase	Audit 1: • https://github.com/MintSwapFinance/Contract-MintSwap404NFT/tree/078b308234acc26dfa4e2e57a1788d9c3fc5b593 Final Audit: • https://github.com/MintSwapFinance/Contract-MintSwap404NFT/tree/1df4c9e70ff8b89b8b390b7680468f84e151896c

1.3. Assessment Summary

Delivery Date	Jul 15, 2024
Audit Methodology	Static Analysis, Formal Verification, Manual Review

1.4. Assessment Scope

ID	File	File Hash
1	/src/lib/DoubleEndedQueue.sol	55cf2d2aff9dab91d1a82002638e69c4
2	/src/erc404/MintSwap404NFT.sol	d9a30f81a528cf875684744e94a8d208
3	/src/rewards/MintSwap404NFTRewards.sol	4f0b9273d81715c942c9c78f6d9f0588

ID	File	File Hash
4	/src/metadata/MetadataRenderer.sol	3d95845e45ebc66f619770c481ef558d
5	/src/erc404/IERC404.sol	f4d24eb72af5902eaf34818ef1a1a46d
6	/src/metadata/IMetadataRenderer.sol	f803ae26c529388183633636c58fd6ec
7	/src/erc404/ERC404.sol	7ecf9031109c3b4710608194af50e00b
8	/src/lib/ERC20Events.sol	4f7c1dae60d959fdf6bd8d803a84dda6
9	/src/stake/MintSwap404NFTStake.sol	535185d841e46dfc188e8513ad29c11f
10	/src/lib/ERC721Events.sol	c466d406bc52c2f09e1de1c8f17b1d13

2. Checklist

2.1. Code Security

Reentrancy	DelegateCall	Integer Overflow
Input Validation	Unchecked this.call	Frozen Money
Arbitrary External Call	Unchecked Owner Transfer	Do-while Continue
Right-To-Left-Override Character	Unauthenticated Storage Access	Risk For Weak Randomness
TxOrigin	Missing Checks for Return Values	Diamond Inheritance
ThisBalance	VarType Deduction	Array Length Manipulation
Uninitialized Variable	Shadow Variable	Divide Before Multiply
Affected by Compiler Bug		

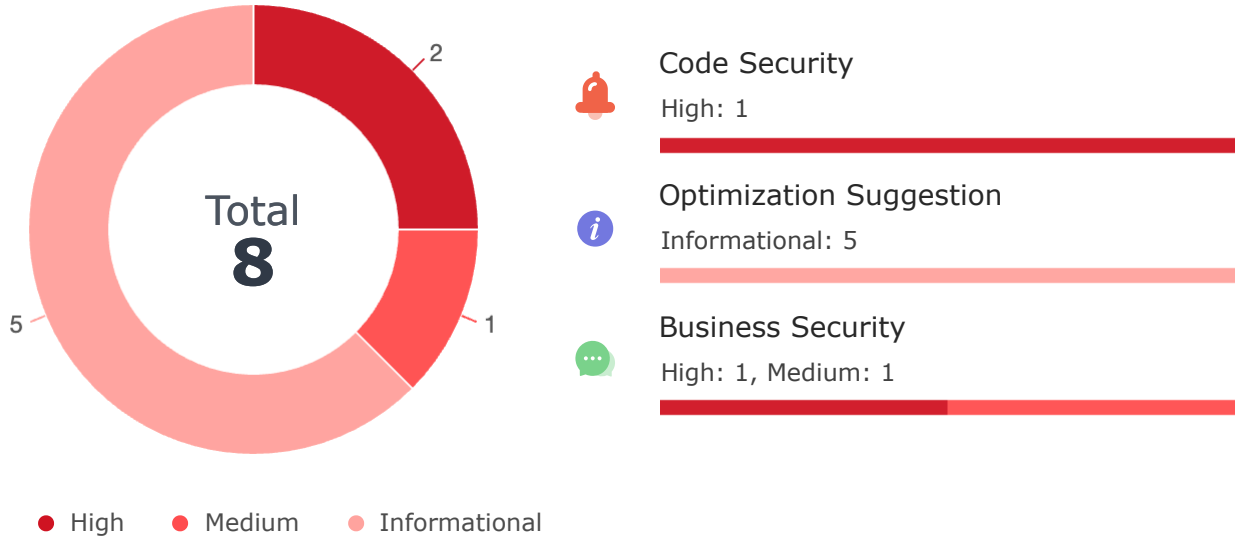
2.2. Optimization Suggestion

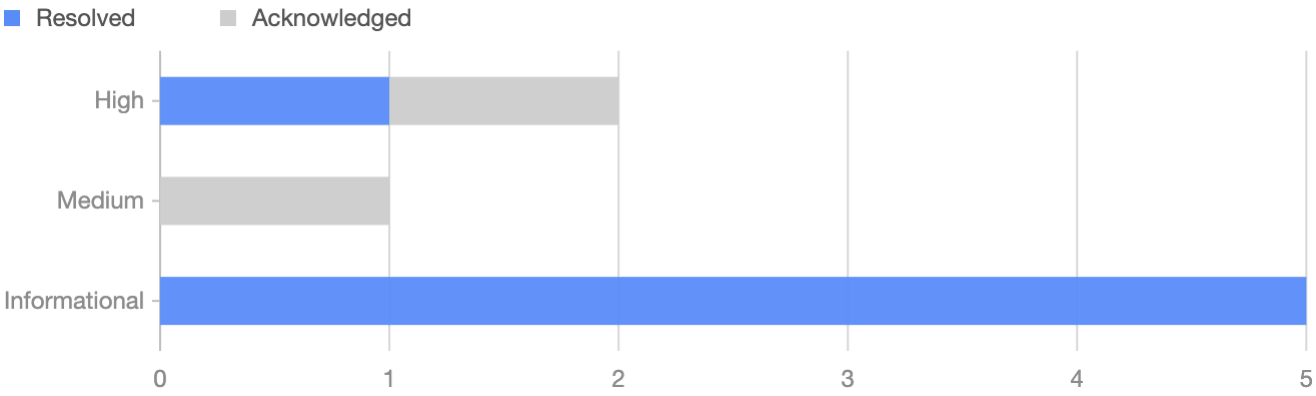
Compiler Version	Improper State Variable Modification
Function Visibility	Deprecated Function
Externally Controlled Variables	Code Style
Constant Specific	Event Specific
Return Value Unspecified	Inexistent Error Message
State Variable Defined Without Storage Location	Import Issue
Compare With Timestamp/Block Number/Blockhash	Constructor in Base Contract Not Implemented
Delete Struct Containing the Mapping Type	Usage of '=+'
Paths in the Modifier Not End with "_" or Revert	Non-payable Public Functions Use msg.value
Lack of SafeMath	Compiler Error/Warning
Tautology Issue	Loop Depends on Array Length
Redundant/Duplicated/Dead Code	Code Complexity/Code Inefficiency
Undeclared Resource	Optimizable Return Statement
Unused Resource	

2.3. Business Security

The Code Implementation is Consistent With Comments, Project White Papers and Other Materials
Permission Check
Address Check

3. Findings





ID	Title	Category	Severity	Status
H-01	Return Value of 0 From ecrecover Is Not Checked	Code Security	High	Resolved
H-02	Missing the Returning of the Residual msg.value	Business Security	High	Acknowledged
M-03	Centralization Risk in Off-Chain Benefit Assessments	Business Security	Medium	Acknowledged
I-04	Use the Checks-Effects-Interactions Pattern to Update the State	Optimization Suggestion	Informational	Resolved
I-05	No Check of Address Params with Zero Address	Optimization Suggestion	Informational	Resolved
I-06	Function Visibility Can Be External	Optimization Suggestion	Informational	Resolved
I-07	Parameters Should Be Declared as Calldata	Optimization Suggestion	Informational	Resolved
I-08	Floating Pragma	Optimization Suggestion	Informational	Resolved

H-01: Return Value of 0 From ecrecover Is Not Checked

High: Code Security

File Location:
/src/rewards/MintSwap404NFTRewards.sol:71
/src/stake/MintSwap404NFTStake.sol:108

Description

The `_verfySigner` function is designed to authenticate the legitimacy of user-provided signatures, thereby enabling the withdrawal of mining and staking rewards. It utilizes the built-in `ecrecover` function to recover the address that signed the message.

However, should the signature be invalid, ecrecover merely returns a zero address, rather than reverting (as detailed in the *Solidity* documentation https://docs.soliditylang.org/en/latest/units-and-global-variables.html#mathematical-and-cryptographic-functions).

Consequently, if the signer address (the state variable `signer`) is either unset or erroneously set to zero, any arbitrary signature could pass this validity check, thus posing a significant security risk where all rewards could potentially be stolen.

/src/rewards/MintSwap404NFTRewards.sol

```
69          uint8 _v
70      ) internal view returns (address _signer) {
71          _signer = ecrecover(
72              keccak256(
73                  abi.encodePacked(
```

/src/stake/MintSwap404NFTStake.sol

```
106         uint8 _v
107     ) internal view returns (address _signer) {
108         _signer = ecrecover(
109             keccak256(
110                 abi.encodePacked(
```

Recommendation

Ensure that the return value of `ecrecover` is not zero to confirm the validity of the signature. Alternatively, consider utilizing the `ECDSA` contract from *OpenZeppelin* (https://docs.openzeppelin.com/contracts/5.x/api/utils#ECDSA) to more securely recover the address that signed the message.

Alleviation

Resolved in commit 1df4c9e. The project team addressed the issue by utilizing the `ECDSA` library to recover the address that signed the message.

H-02: Missing the Returning of the Residual msg.value

High: Business Security

File Location: /src/erc404/MintSwap404NFT.sol:106

Description

The `publicSale` function facilitates the public sale of the ERC-404 tokens. However, it currently lacks the capability to refund any excess Ether sent by buyers beyond the token's selling price, thereby rendering the residual `msg.value` to be lost.

/src/erc404/MintSwap404NFT.sol

```
106   function publicSale(uint numberOfTokens) external payable
      isPublicSaleTime {
107       require(numberOfTokens > 0 && _publicMintedCount + numberOfTokens <=
          SALE_TOTAL_COUNT, "Mint numberOfTokens exceeds limit");
108       require(PUBLIC_SALE_PRICE * numberOfTokens <= msg.value, "Not Enough
          ETH value to mint tokens");
109
110       _mintERC20(_msgSender(), numberOfTokens * units);
111       _publicMintedCount += numberOfTokens;
112   }
```

Recommendation

It is recommended to incorporate logic within the `publicSale` function to facilitate the return of any residual `msg.value` to the buyer, ensuring equitable transaction processing.

Alleviation

The project team has confirmed that this implementation complies with the project design and decided to keep no change.

M-03: Centralization Risk in Off-Chain Benefit Assessments

Medium: Business Security

File Location:
/src/rewards/MintSwap404NFTRewards.sol:34
/src/stake/MintSwap404NFTStake.sol:73

Description

The functions `withdrawBenefits` in the `MintSwap404NFTRewards` contract and `withdrawStakeBenefits` in the `MintSwap404NFTStake` contract enable users to withdraw their mining and staking rewards with messages signed by the protocol.

Yet, the computation of these rewards occurs entirely off-chain, presenting a significant centralization threat to the protocol. Should the off-chain service calculating these benefits contains bugs or be compromised, it could break the entire integrity of the protocol.

/src/rewards/MintSwap404NFTRewards.sol

```
34  function withdrawBenefits(
35      uint256 _amount,
36      bytes32 _r,
37      bytes32 _s,
38      uint8 _v
39  ) external {
40      address sender = msg.sender;
41      require(_verfySigner(sender, _amount, _r, _s, _v) == signer,
        "Invalid signer");
42      require(_amount > alreadyClaim[sender], "Invalid withdraw amount");
43
44      uint256 canClaimAmount = _amount - alreadyClaim[sender];
45      IERC404(mintswap404NFT).transferFrom(rewardsAccount, sender,
        canClaimAmount);
46      alreadyClaim[sender] = _amount;
47      emit WithdrawRewardsBenefits(sender, canClaimAmount);
48  }
```

/src/stake/MintSwap404NFTStake.sol

```
73  function withdrawStakeBenefits(
74      uint256 _amount,
75      bytes32 _r,
76      bytes32 _s,
77      uint8 _v
78  ) external nonReentrant {
79      address payable sender = payable(msg.sender);
80      require(_verfySigner(sender, _amount, _r, _s, _v) == signer,
        "Invalid signer");
81      require(_amount > alreadyClaim[sender], "Invalid withdraw amount");
82
```

```
83        uint256 canClaimAmount = _amount - alreadyClaim[sender];
84        (bool success, ) = sender.call{value: canClaimAmount}(new bytes(0));
85        require(success, 'ETH transfer failed');
86
87        alreadyClaim[sender] = _amount;
88        emit WithdrawStakeBenefits(sender, canClaimAmount);
89    }
```

Recommendation

Consider implementing the benefit calculation on-chain to enhance security and transparency.

Alleviation

The project team acknowledged the issue but has decided not to make any changes due to the complexity of the reward distribution logic.

I-04: Use the Checks-Effects-Interactions Pattern to Update the State

Informational: Optimization Suggestion

File Location:
/src/erc404/MintSwap404NFT.sol:100,101,102,103,110,111,118,119
/src/rewards/MintSwap404NFTRewards.sol:45,46
/src/stake/MintSwap404NFTStake.sol:63,64

Description

To enhance the security of the code, it is recommended to use the `Checks-Effects-Interactions` pattern to update the state. For example, in the `withdraw` function of the `MintSwap404NFTStake` contract, it is suggested to first delete `stakedTokens[tokenIds[i]]` and then transfer the ERC-404 NFT to the caller, in order to better comply with code safety standards and enhance code security.

/src/erc404/MintSwap404NFT.sol

```
91  function wlSale() external payable isWLSaleTime {
92      address account = _msgSender();
93      if (!whitelist[account]) revert UnauthorizedMinter(account);
94
95      require(!wlMinted[account], "This account has already WL minted");
96      require(_wlMintedCount + 1 <= WL_SALE_COUNT, "WL mint exceeds
        limit");
97      require(_publicMintedCount + 1 <= SALE_TOTAL_COUNT, "Mint exceeds
        limit");
98      require(WHITELIST_SALE_PRICE <= msg.value, "Not Enough ETH value to
        WL mint tokens");
99
100     _mintERC20(account, units);
101     wlMinted[account] = true;
102     _publicMintedCount++;
103     _wlMintedCount++;
104 }
105
106 function publicSale(uint numberOfTokens) external payable
    isPublicSaleTime {
107     require(numberOfTokens > 0 && _publicMintedCount + numberOfTokens <=
        SALE_TOTAL_COUNT, "Mint numberOfTokens exceeds limit");
108     require(PUBLIC_SALE_PRICE * numberOfTokens <= msg.value, "Not Enough
        ETH value to mint tokens");
109
110     _mintERC20(_msgSender(), numberOfTokens * units);
111     _publicMintedCount += numberOfTokens;
112 }
113
114 function mintRewards(address exemptAddress, uint256 amount) external
    onlyOwner {
115     require(erc721TransferExempt(exemptAddress), "The address is not
        erc721TransferExempt");
```

```
116         require(amount > 0 && _mintswapMintedCount + amount <=
            MINTSWAP_REWARDS_COUNT, "The maximum mint rewards quantity cannot
            exceed 7000");
117
118         _mintERC20(exemptAddress, amount * units);
119         _mintswapMintedCount += amount;
120     }
```

/src/rewards/MintSwap404NFTRewards.sol

```
34  function withdrawBenefits(
35      uint256 _amount,
36      bytes32 _r,
37      bytes32 _s,
38      uint8 _v
39  ) external {
40      address sender = msg.sender;
41      require(_verfySigner(sender, _amount, _r, _s, _v) == signer,
        "Invalid signer");
42      require(_amount > alreadyClaim[sender], "Invalid withdraw amount");
43
44      uint256 canClaimAmount = _amount - alreadyClaim[sender];
45      IERC404(mintswap404NFT).transferFrom(rewardsAccount, sender,
        canClaimAmount);
46      alreadyClaim[sender] = _amount;
47      emit WithdrawRewardsBenefits(sender, canClaimAmount);
48  }
```

/src/stake/MintSwap404NFTStake.sol

```
56  function withdraw(uint256[] calldata tokenIds) external {
57      require(tokenIds.length > 0, "Withdrawing zero tokens");
58      address sender = msg.sender;
59
60      for (uint256 i = 0; i < tokenIds.length; ) {
61          address nftOwner = stakedTokens[tokenIds[i]];
62          require(sender == nftOwner, "Invalid tokenId");
63          IERC404(mintswap404NFT).transferFrom(address(this), sender,
            tokenIds[i]);
64          delete stakedTokens[tokenIds[i]];
65          unchecked {
66              ++i;
67          }
68      }
69
70      emit TokensWithdraw(sender, tokenIds);
71  }
```

Recommendation

Suggested modifications are as follows:

/src/erc404/MintSwap404NFT.sol:

```
function wlSale() external payable isWLSaleTime {
    address account = _msgSender();
    if (!whitelist[account]) revert UnauthorizedMinter(account);

    require(!wlMinted[account], "This account has already WL minted");
    require(_wlMintedCount + 1 <= WL_SALE_COUNT, "WL mint exceeds limit");
    require(WHITELIST_SALE_PRICE <= msg.value, "Not Enough ETH value to WL
mint tokens");

    wlMinted[account] = true;
    _publicMintedCount++;
    _wlMintedCount++;
    _mintERC20(account, units);
}

function publicSale(uint numberOfTokens) external payable isPublicSaleTime {
    require(numberOfTokens > 0 && _publicMintedCount + numberOfTokens <=
SALE_TOTAL_COUNT, "Mint numberOfTokens exceeds limit");
    require(PUBLIC_SALE_PRICE * numberOfTokens <= msg.value, "Not Enough
ETH value to mint tokens");

    _publicMintedCount += numberOfTokens;
    _mintERC20(_msgSender(), numberOfTokens * units);
}

function mintRewards(address exemptAddress, uint256 amount) external
onlyOwner {
    require(erc721TransferExempt(exemptAddress), "The address is not
erc721TransferExempt");
    require(amount > 0 && _mintswapMintedCount + amount <=
MINTSWAP_REWARDS_COUNT, "The maximum mint rewards quantity cannot exceed
7000");

    _mintswapMintedCount += amount;
    _mintERC20(exemptAddress, amount * units);
}
```

/src/stake/MintSwap404NFTStake.sol:

```
function withdraw(uint256[] calldata tokenIds) external {
    require(tokenIds.length > 0, "Withdrawing zero tokens");
    address sender = msg.sender;

    for (uint256 i = 0; i < tokenIds.length; ) {
        address nftOwner = stakedTokens[tokenIds[i]];
        require(sender == nftOwner, "Invalid tokenId");
        delete stakedTokens[tokenIds[i]];
        IERC404(mintswap404NFT).transferFrom(address(this), sender,
tokenIds[i]);
        unchecked {
            ++i;
```

```
            }
        }

        emit TokensWithdraw(sender, tokenIds);
    }
```

/src/rewards/MintSwap404NFTRewards.sol:

```
    function withdrawBenefits(
        uint256 _amount,
        bytes32 _r,
        bytes32 _s,
        uint8 _v
    ) external {
        address sender = msg.sender;
        require(_verfySigner(sender, _amount, _r, _s, _v) == signer, "Invalid
    signer");
        require(_amount > alreadyClaim[sender], "Invalid withdraw amount");

        uint256 canClaimAmount = _amount - alreadyClaim[sender];
        alreadyClaim[sender] = _amount;
        IERC404(mintswap404NFT).transferFrom(rewardsAccount, sender,
    canClaimAmount);
        emit WithdrawRewardsBenefits(sender, canClaimAmount);
    }
```

Alleviation

Resolved in commit 1df4c9e.

I-05: No Check of Address Params with Zero Address

Informational: Optimization Suggestion

File Location:
/src/erc404/MintSwap404NFT.sol:83,144
/src/rewards/MintSwap404NFTRewards.sol:27,50,54
/src/stake/MintSwap404NFTStake.sol:32,91

Description

The input parameter of the address type in the function does not use the zero address for verification.

/src/erc404/MintSwap404NFT.sol

```
81      }
82
83      function setMetadataRenderer(address _metadataRenderer) external
        onlyOwner {
84          metadataRenderer = _metadataRenderer;
85      }
```

/src/erc404/MintSwap404NFT.sol

```
142     {}
143
144     function withdrawETH(address _to, uint256 _amount) external
        onlyOwner {
145         (bool success, ) = _to.call{value: _amount}(new bytes(0));
146         require(success, 'ETH transfer failed');
```

/src/rewards/MintSwap404NFTRewards.sol

```
25      }
26
27      function initialize(address initialOwner, address _mintswap404NFT)
        initializer public {
28          __Ownable_init(initialOwner);
29          __UUPSUpgradeable_init();
```

/src/rewards/MintSwap404NFTRewards.sol

```
48      }
49
50      function setSigner(address _signer) external onlyOwner {
51          signer = _signer;
52      }
53
54      function setRewardsAccount(address _rewardsAccount) external
        onlyOwner {
55          rewardsAccount = _rewardsAccount;
```

```
56      }
```

/src/stake/MintSwap404NFTStake.sol

```
30      }
31
32      function initialize(address initialOwner, address _mintswap404NFT)
        initializer public {
33          __Ownable_init(initialOwner);
34          __ReentrancyGuard_init();
```

/src/stake/MintSwap404NFTStake.sol

```
89      }
90
91      function setSigner(address _signer) public onlyOwner {
92          signer = _signer;
93      }
```

Recommendation

It is recommended to perform zero address verification on the input parameters of the address type.

Alleviation

Resolved in commit 1df4c9e.

I-06: Function Visibility Can Be External

Informational: Optimization Suggestion

File Location:
/src/erc404/MintSwap404NFT.sol:58,78
/src/metadata/MetadataRenderer.sol:30
/src/rewards/MintSwap404NFTRewards.sol:27
/src/stake/MintSwap404NFTStake.sol:32,91

Description

Functions that are not called should be declared as external.

/src/erc404/MintSwap404NFT.sol

```
52      function initialize(
53          address initialOwner,
54          string memory name_,
55          string memory symbol_,
56          uint8 decimals_,
57          uint256 unitMultiplicator_
58      ) public initializer {
```

/src/erc404/MintSwap404NFT.sol

```
74      }
75
76      function tokenURI(
77          uint256 tokenId
78      ) public view override returns (string memory) {
```

/src/metadata/MetadataRenderer.sol

```
26      }
27
28      function tokenURI(
29          uint256 tokenID
30      ) public view override returns (string memory) {
```

/src/rewards/MintSwap404NFTRewards.sol

```
25      }
26
27      function initialize(address initialOwner, address _mintswap404NFT)
        initializer public {
28          __Ownable_init(initialOwner);
29          __UUPSUpgradeable_init();
```

/src/stake/MintSwap404NFTStake.sol

```
30      }
31
32      function initialize(address initialOwner, address _mintswap404NFT)
        initializer public {
33          __Ownable_init(initialOwner);
34          __ReentrancyGuard_init();
```

/src/stake/MintSwap404NFTStake.sol

```
89      }
90
91      function setSigner(address _signer) public onlyOwner {
92          signer = _signer;
93      }
```

Recommendation

Functions that are not called in the contract should be declared as external.

Alleviation

Resolved in commit 1df4c9e.

I-07: Parameters Should Be Declared as Calldata

Informational: Optimization Suggestion

File Location: /src/metadata/MetadataRenderer.sol:62,66,70

Description

When the compiler parses the external or public function, it can directly read the function parameters from calldata. Setting it to other storage locations may waste gas. About 300-400 gas can be saved with optimization turned off while 120-150 gas can be saved vice versa.

/src/metadata/MetadataRenderer.sol

```
60      }
61
62      function setName(string memory _newName) external onlyOwner {
63          name = _newName;
64      }
65
66      function setImageUri(string memory _newURI) external onlyOwner {
67          imageURI = _newURI;
68      }
69
70      function setDescription(string memory _description) external
        onlyOwner {
71          description = _description;
72      }
```

Recommendation

In external or public functions, the storage location of function parameters should be set to calldata to save gas.

Alleviation

Resolved in commit 1df4c9e.

I-08: Floating Pragma

Informational: Optimization Suggestion

File Location:
/src/erc404/ERC404.sol:2
/src/erc404/IERC404.sol:2
/src/erc404/MintSwap404NFT.sol:2
/src/lib/DoubleEndedQueue.sol:5
/src/lib/ERC20Events.sol:2
/src/lib/ERC721Events.sol:2
/src/metadata/IMetadataRenderer.sol:2
/src/metadata/MetadataRenderer.sol:2
/src/rewards/MintSwap404NFTRewards.sol:2
/src/stake/MintSwap404NFTStake.sol:2

Description

Contracts should be deployed with fixed compiler version which has been tested thoroughly or make sure to lock the contract compiler version in the project configuration. Locked compiler version ensures that contracts will not be compiled by untested compiler version.

/src/erc404/ERC404.sol

```
1  //SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.26;
3
4  import "@openzeppelin/contracts/interfaces/IERC721Receiver.sol";
```

/src/erc404/IERC404.sol

```
1  //SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.26;
3
4  import "@openzeppelin/contracts/interfaces/IERC165.sol";
```

/src/erc404/MintSwap404NFT.sol

```
1  //SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.26;
3
4  import "@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.
   sol";
```

/src/lib/DoubleEndedQueue.sol

```
3  // OpenZeppelin Contracts (last updated v5.0.0) (utils/structs/
   DoubleEndedQueue.sol)
4  // Modified by Pandora Labs to support native uint256 operations
5  pragma solidity ^0.8.26;
6
7  /**
```

/src/lib/ERC20Events.sol

```
1  // SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.26;
3
4  library ERC20Events {
```

/src/lib/ERC721Events.sol

```
1  // SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.26;
3
4  library ERC721Events {
```

/src/metadata/IMetadataRenderer.sol

```
1  // SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.26;
3
4  interface IMetadataRenderer {
```

/src/metadata/MetadataRenderer.sol

```
1  // SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.26;
3
4  import "@openzeppelin/contracts/access/Ownable.sol";
```

/src/rewards/MintSwap404NFTRewards.sol

```
1  // SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.26;
3
4  import "@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.
   sol";
```

/src/stake/MintSwap404NFTStake.sol

```
1  // SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.26;
3
4  import "@openzeppelin/contracts-upgradeable/utils/
   ReentrancyGuardUpgradeable.sol";
```

Recommendation

Use a fixed compiler version, and consider whether the bugs in the selected compiler version (https://github.com/ethereum/solidity/releases) will affect the contract.

Alleviation

Resolved in commit 1df4c9e.

4. Disclaimer

No description, statement, recommendation or conclusion in this report shall be construed as endorsement, affirmation or confirmation of the project. The security assessment is limited to the scope of work as stipulated in the Statement of Work.

This report is prepared in response to source code, and based on the attacks and vulnerabilities in the source code that already existed or occurred before the date of this report, excluding any new attacks or vulnerabilities that exist or occur after the date of this report. The security assessment are solely based on the documents and materials provided by the customer, and the customer represents and warrants documents and materials are true, accurate and complete.

CONSULTANT DOES NOT MAKE AND HEREBY DISCLAIMS ANY REPRESENTATIONS OR WARRANTIES OF ANY KIND, WHETHER EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, REGARDING THE SERVICES, DELIVERABLES, OR ANY OTHER MATTER PERTAINING TO THIS REPORT.

CONSULTANT SHALL NOT BE RESPONSIBLE FOR AND HEREBY DISCLAIMS MERCHANTABILITY, FITNESS FOR PURPOSE, TITLE, NON-INFRINGEMENT OR NON-APPROPRIATION OF INTELLECTUAL PROPERTY RIGHTS OF A THIRD PARTY, SATISFACTORY QUALITY, ACCURACY, QUALITY, COMPLETENESS, TIMELINESS, RESPONSIVENESS, OR PRODUCTIVITY OF THE SERVICES OR DELIVERABLES.

CONSULTANT EXCLUDES ANY WARRANTY THAT THE SERVICES AND DELIVERABLES WILL BE UNINTERRUPTED, ERROR FREE, FREE OF SECURITY DEFECTS OR HARMFUL COMPONENTS, REVEAL ALL SECURITY VULNERABILITIES, OR THAT ANY DATA WILL NOT BE LOST OR CORRUPTED.

CONSULTANT SHALL NOT BE RESPONSIBLE FOR (A) ANY REPRESENTATIONS MADE BY ANY PERSON REGARDING THE SUFFICIENCY OR SUITABILITY OF SERVICES AND DELIVERABLES IN ANY ACTUAL APPLICATION, OR (B) WHETHER ANY SUCH USE WOULD VIOLATE OR INFRINGE THE APPLICABLE LAWS, OR (C) REVIEWING THE CUSTOMER MATERIALS FOR ACCURACY.

5. Appendix

5.1 Visibility

Contract	FuncName	Visibility	Mutability	Modifiers
MintSwap404NFT	_CTOR_	public	Y	
MintSwap404NFT	initialize	public	Y	initializer
MintSwap404NFT	tokenURI	public	N	
MintSwap404NFT	setMetadataRender er	external	Y	onlyOwner
MintSwap404NFT	setERC721TransferE xempt	external	Y	onlyOwner
MintSwap404NFT	wlSale	external	Y	isWLSaleTime
MintSwap404NFT	publicSale	external	Y	isPublicSaleTime
MintSwap404NFT	mintRewards	external	Y	onlyOwner
MintSwap404NFT	setMintConfig	external	Y	onlyOwner
MintSwap404NFT	setWLConfig	external	Y	onlyOwner
MintSwap404NFT	_authorizeUpgrade	internal	N	onlyOwner
MintSwap404NFT	withdrawETH	external	N	onlyOwner
MintSwap404NFT	addWhitelist	external	Y	onlyOwner
MintSwap404NFTR ewards	_CTOR_	public	Y	
MintSwap404NFTR ewards	initialize	public	Y	initializer
MintSwap404NFTR ewards	withdrawBenefits	external	Y	
MintSwap404NFTR ewards	setSigner	external	Y	onlyOwner

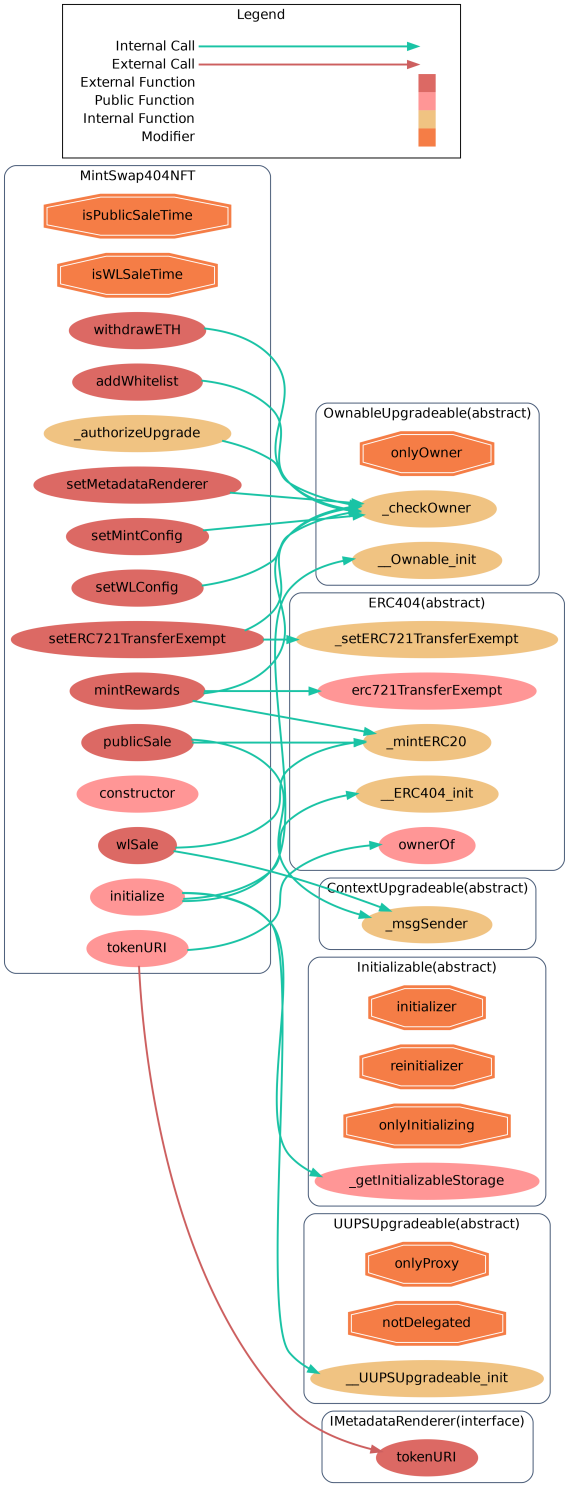
Contract	FuncName	Visibility	Mutability	Modifiers
MintSwap404NFTRewards	setRewardsAccount	external	Y	onlyOwner
MintSwap404NFTRewards	_authorizeUpgrade	internal	N	onlyOwner
MintSwap404NFTRewards	_verfySigner	internal	N	
MintSwap404NFTRewards	_computeDomainSeparator	internal	N	
MetadataRenderer	_CTOR_	public	Y	
MetadataRenderer	tokenURI	public	N	
MetadataRenderer	tokenURIJSON	public	N	
MetadataRenderer	setName	external	Y	onlyOwner
MetadataRenderer	setImageUri	external	Y	onlyOwner
MetadataRenderer	setDescription	external	Y	onlyOwner
MintSwap404NFTStake	_CTOR_	public	Y	
MintSwap404NFTStake	initialize	public	Y	initializer
MintSwap404NFTStake	stake	external	Y	
MintSwap404NFTStake	withdraw	external	N	
MintSwap404NFTStake	withdrawStakeBenefits	external	Y	nonReentrant
MintSwap404NFTStake	setSigner	public	Y	onlyOwner
MintSwap404NFTStake	_authorizeUpgrade	internal	N	onlyOwner

Contract	FuncName	Visibility	Mutability	Modifiers
MintSwap404NFTStake	_verfySigner	internal	N	
MintSwap404NFTStake	_computeDomainSeparator	internal	N	
MintSwap404NFTStake	receive	external	N	

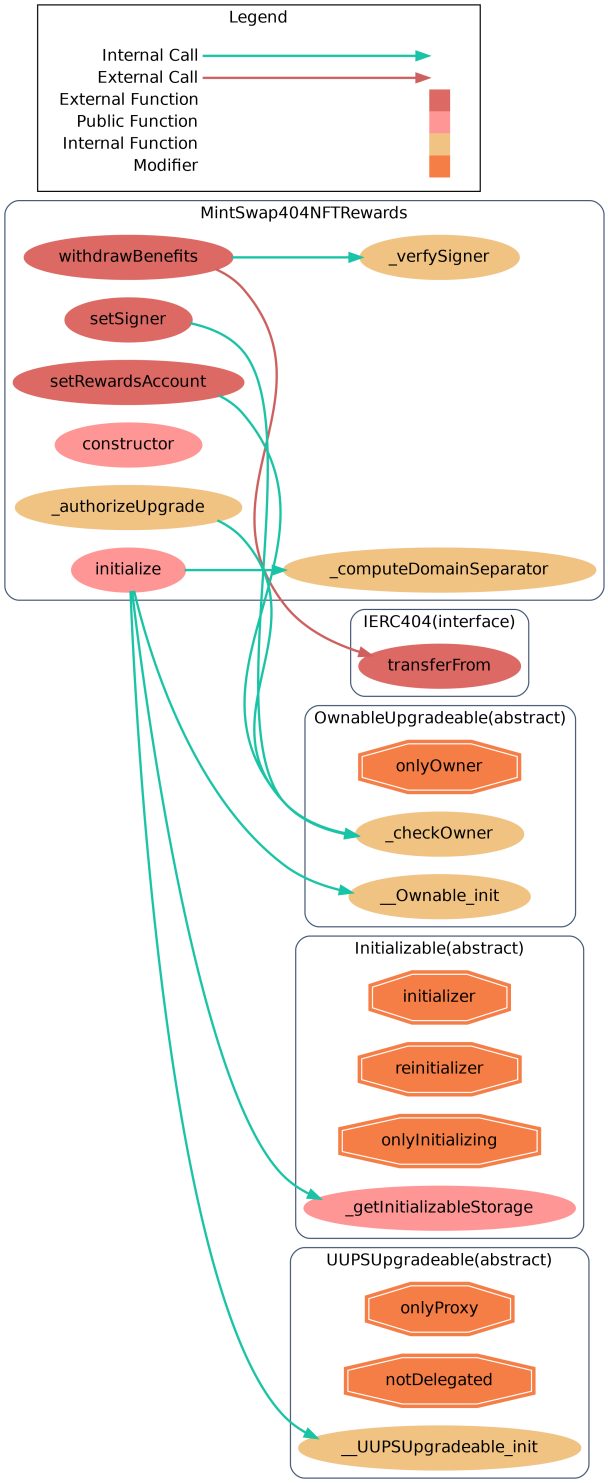
5. Appendix

5.2 Call Graph

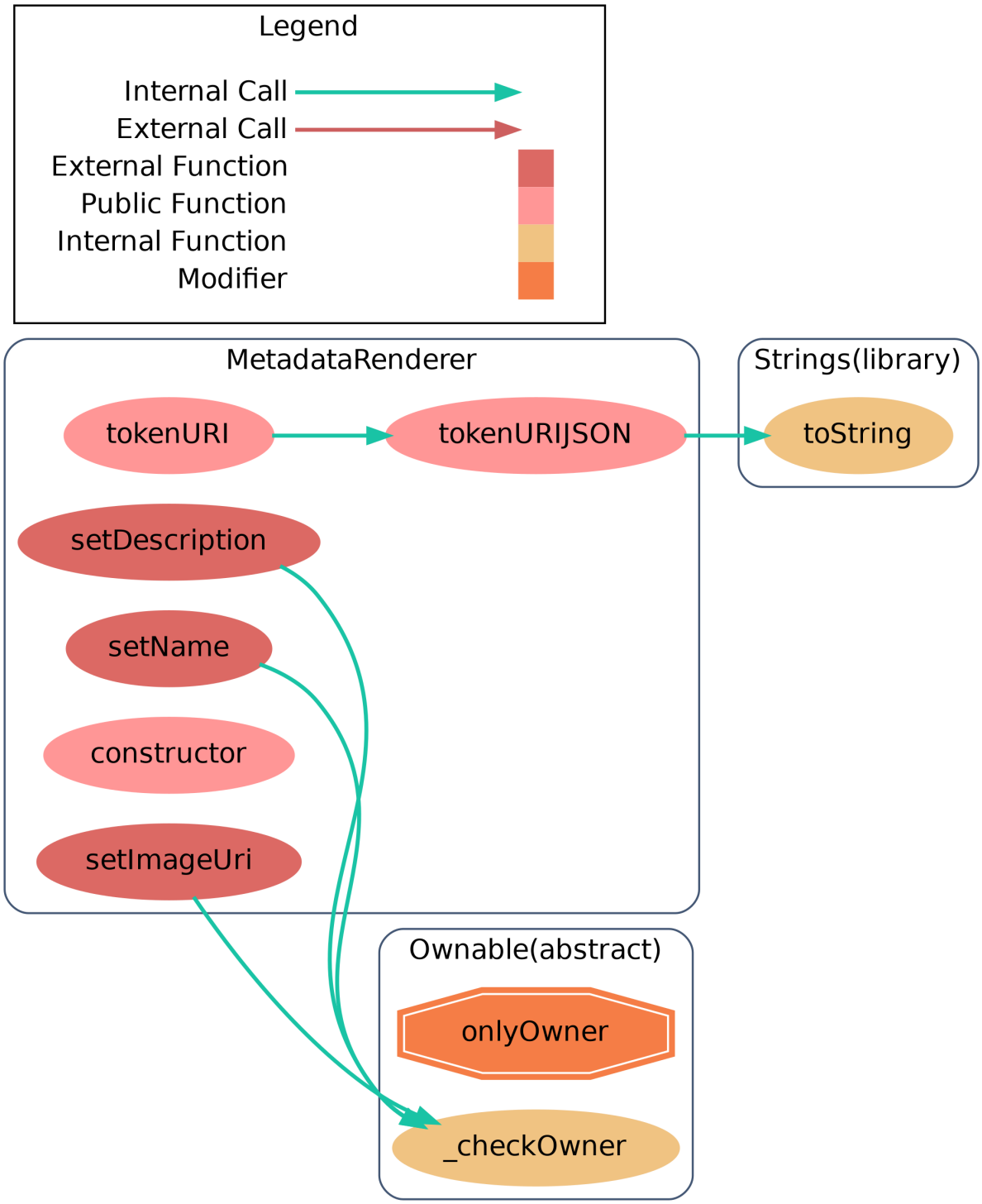
MintSwap404NFT



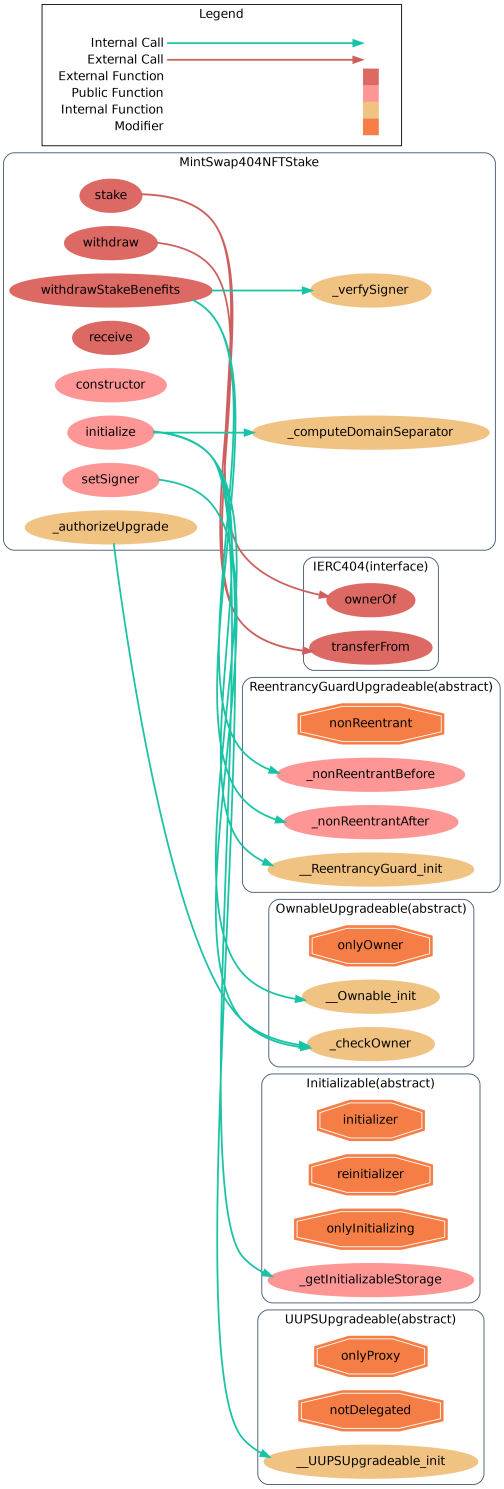
MintSwap404NFTRewards



MetadataRenderer



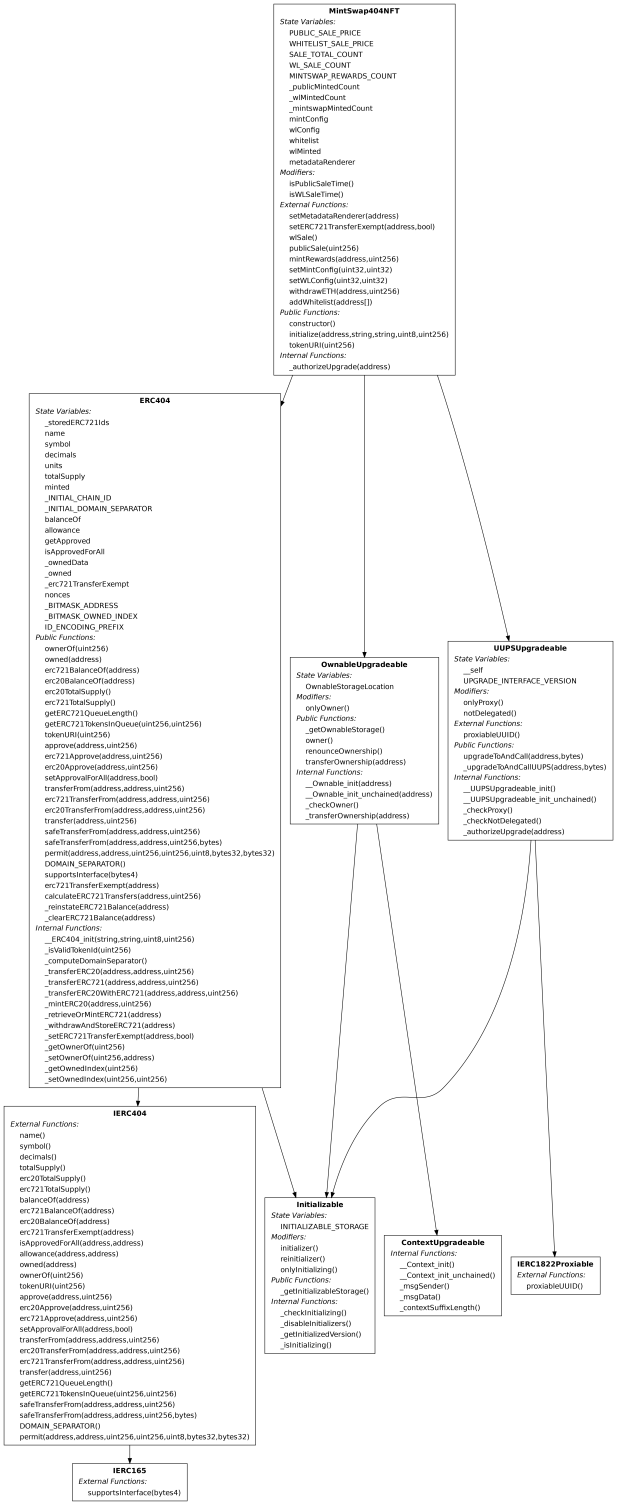
MintSwap404NFTStake



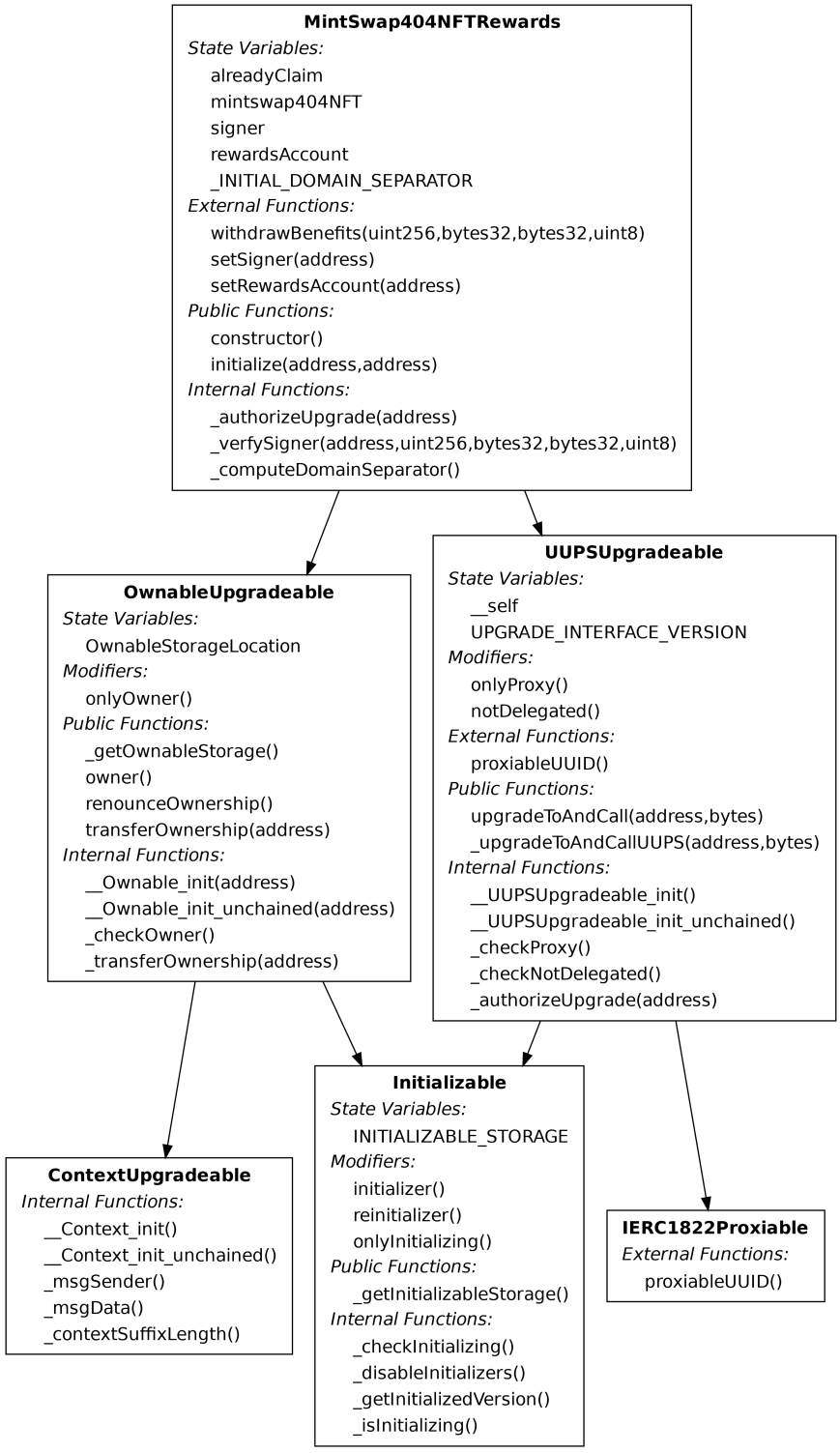
5. Appendix

5.3 Inheritance Graph

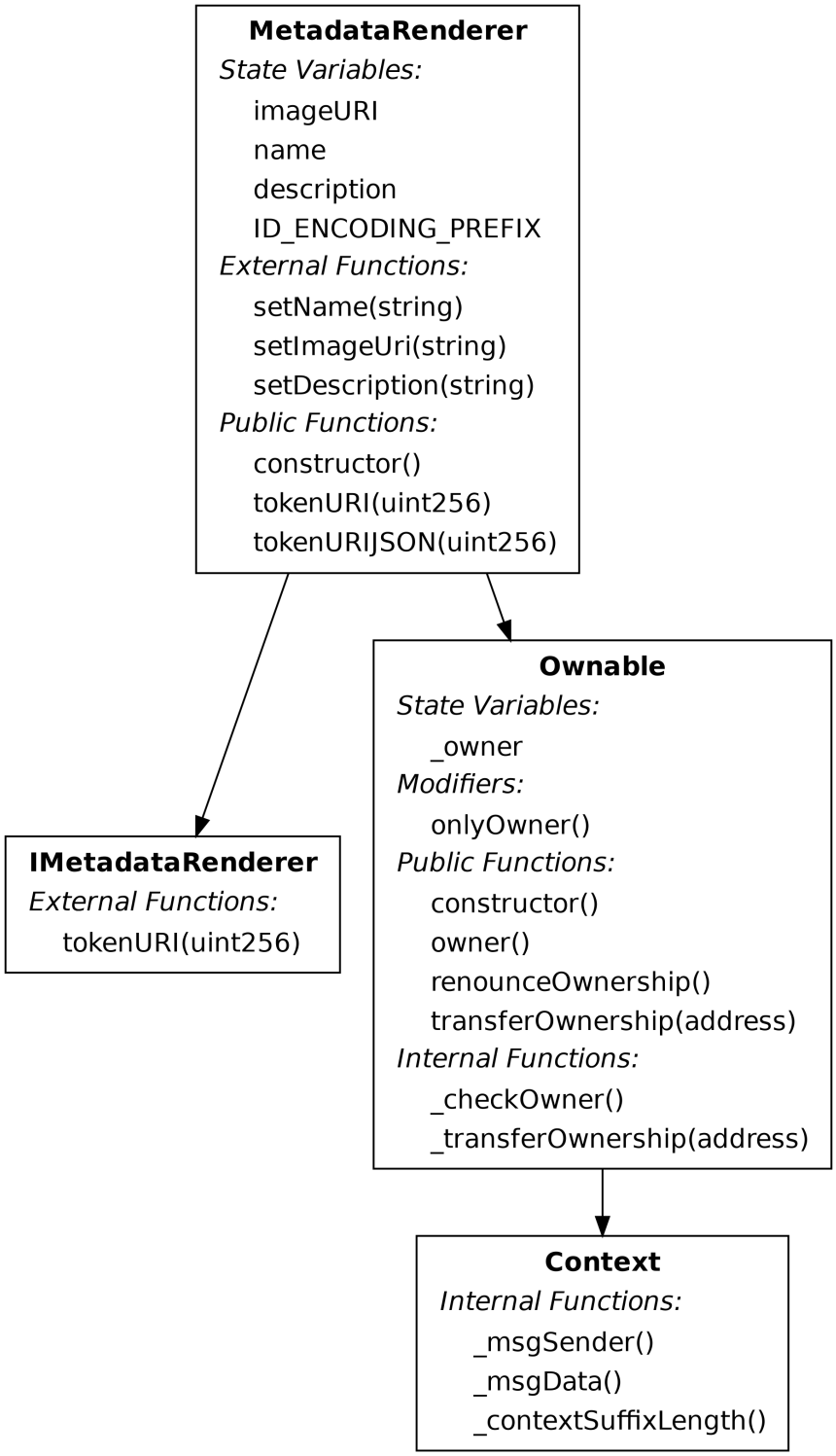
MintSwap404NFT



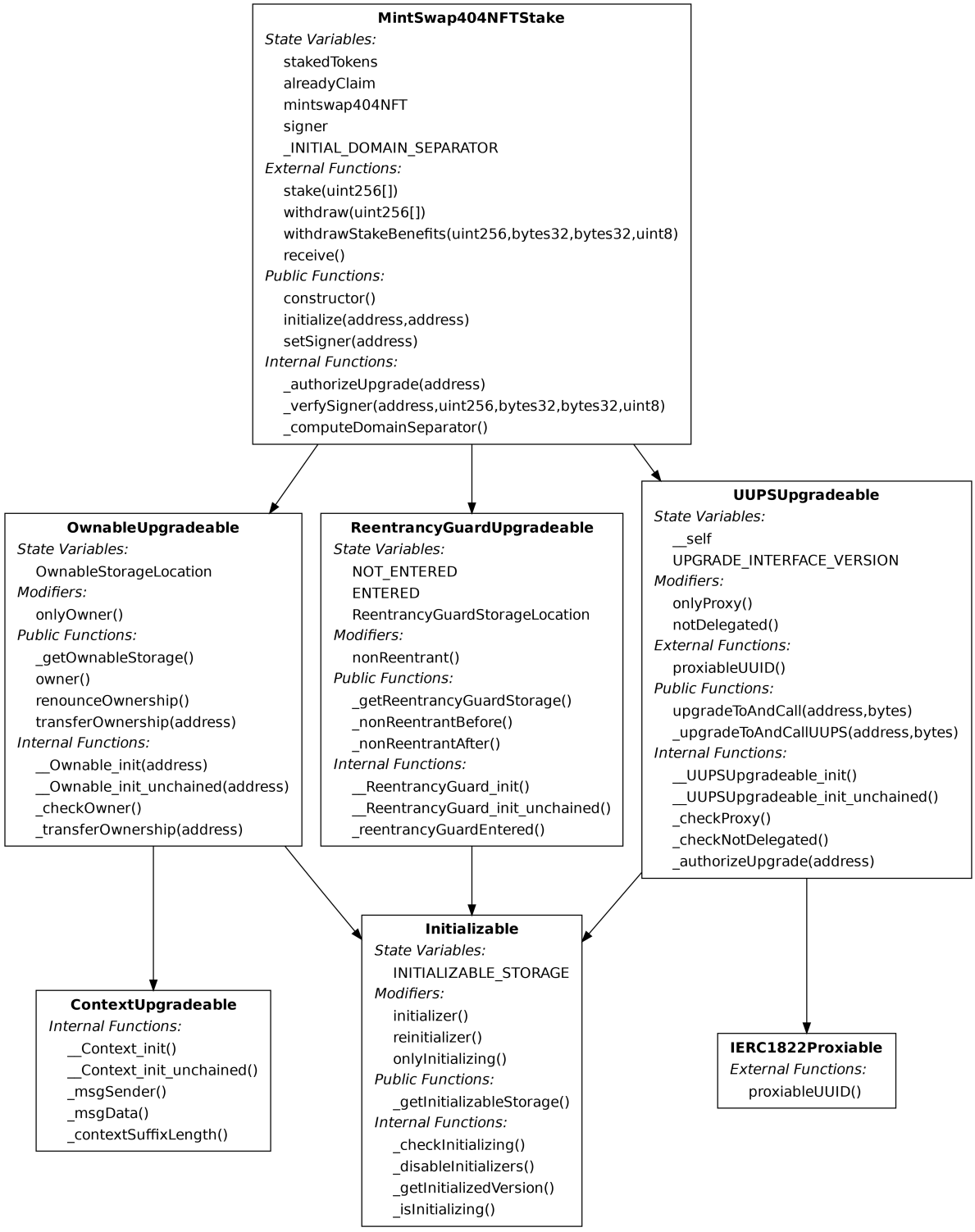
MintSwap404NFTRewards



MetadataRenderer



MintSwap404NFTStake



5.4 Formal Verification Metadata

1. The balance of this contract is anticipated to rise following a successful public sale.

```
/// #if_succeeds {:msg "The balance of this contract is anticipated to rise
following a successful public sale."} address(this).balance ==
old(address(this).balance) + PUBLIC_SALE_PRICE * numberOfTokens;
function publicSale(uint numberOfTokens) external payable isPublicSaleTime {
```

Violated. Please refer to H-02.

2. Upon successful completion of the publicSale function, the number of minted tokens should not exceed the PUBLIC_SALE_COUNT.

```
/// #if_succeeds {:msg "Upon successful completion of the publicSale
function, the number of minted tokens should not exceed the
PUBLIC_SALE_COUNT."} _publicMintedCount <= PUBLIC_SALE_COUNT;
function publicSale(uint numberOfTokens) external payable isPublicSaleTime {
```

Passed.

3. If the publicSale function completes successfully, the number of tokens minted in this transaction is correctly added to the total number of public minted tokens.

```
/// #if_succeeds {:msg "If the publicSale function completes successfully,
the number of tokens minted in this transaction is correctly added to the
total number of public minted tokens."} _publicMintedCount ==
old(_publicMintedCount) + numberOfTokens;
function publicSale(uint numberOfTokens) external payable isPublicSaleTime {
```

Passed.

4. Upon the successful execution of the function mintRewards, the total count of minted tokens under the MintSwap rewards category should increment by the exact amount specified.

```
/// #if_succeeds {:msg "Upon the successful execution of the function
mintRewards, the total count of minted tokens under the MintSwap rewards
category should increment by the exact amount specified."}
_mintswapMintedCount == old(_mintswapMintedCount) + amount;
function mintRewards(address exemptAddress, uint256 amount) external
onlyOwner {
```

Passed.

5. The function mintRewards should only succeed if the total minted tokens after the operation do not exceed the MINTSWAP_REWARDS_COUNT limit.

```
/// #if_succeeds {:msg "The function mintRewards should only succeed if the
total minted tokens after the operation do not exceed the
MINTSWAP_REWARDS_COUNT limit."} _mintswapMintedCount <=
MINTSWAP_REWARDS_COUNT;
function mintRewards(address exemptAddress, uint256 amount) external
onlyOwner {
```

Passed.

6. The mintRewards function should only succeed if the given address (exemptAddress) is marked as exempt for ERC721 transfers.

```
/// #if_succeeds {:msg "The mintRewards function should only succeed if the
given address (exemptAddress) is marked as exempt for ERC721 transfers."}
erc721TransferExempt(exemptAddress) == true;
function mintRewards(address exemptAddress, uint256 amount) external
onlyOwner {
```

Passed.

7. Upon successful execution of withdrawETH, the transferred amount of ETH should correctly be subtracted from the contract's balance.

```
/// #if_succeeds {:msg "Upon successful execution of withdrawETH, the
transferred amount of ETH should correctly be subtracted from the
contract's balance."} old(address(this).balance) == address(this).balance +
_amount;
function withdrawETH(address _to, uint256 _amount) external onlyOwner {
```

Passed.

8. The function withdrawETH should only succeed when the caller is the contract owner.

```
/// #if_succeeds {:msg "The function withdrawETH should only succeed when
the caller is the contract owner."} msg.sender == owner();
function withdrawETH(address _to, uint256 _amount) external onlyOwner {
```

Passed.

9. Upon the execution of withdrawETH, it's imperative that the `_to` address should not be address(0).

```
/// #if_succeeds {:msg "Upon the execution of withdrawETH, it's imperative
that the `_to` address should not be address(0)." } _to != address(0);
```

```
function withdrawETH(address _to, uint256 _amount) external onlyOwner {
```

Passed.

10. After successfully withdrawing benefits, the user's reward benefits balance must decrease by the amount withdrawn.

```
/// #if_succeeds {:msg "After successfully withdrawing benefits, the user's
reward benefits balance must decrease by the amount withdrawn."}
old(userRewardsBenefits[msg.sender]) == userRewardsBenefits[msg.sender] +
benefit;
function withdrawBenefits(
    uint256 _amount,
    bytes32 _r,
    bytes32 _s,
    uint8 _v
) external {
```

Passed.

11. The withdrawal should not succeed if the requested benefit amount is less than the minimum withdrawal amount.

```
/// #if_succeeds {:msg "The withdrawal should not succeed if the requested
benefit amount is less than the minimum withdrawal amount."} benefit >=
MIN_WITHDRAW_AMOUNT;
function withdrawBenefits(
    uint256 _amount,
    bytes32 _r,
    bytes32 _s,
    uint8 _v
) external {
```

Passed.

12. The withdrawal operation must ensure that the `rewardsAccount` has enough tokens to transfer.

```
/// #if_succeeds {:msg "The withdrawal operation must ensure that the
`rewardsAccount` has enough tokens to transfer."}
IERC404(mintswap404NFT).balanceOf(rewardsAccount) >= benefit;
function withdrawBenefits(
    uint256 _amount,
    bytes32 _r,
    bytes32 _s,
    uint8 _v
) external {
```

Passed.

13. After the `stake` function executes, the sender should be the owner of all tokens provided in the `tokenIds` array.

```
/// #if_succeeds {:msg "After the `stake` function executes, the sender
should be the owner of all tokens provided in the `tokenIds` array."}
tokenIds.length > 0 ==> forall (uint i in 0...tokenIds.length-1)
stakedTokens[tokenIds[i]] == msg.sender;
function stake(uint256[] calldata tokenIds) external {
```

Passed.

14. After withdrawing, the owner of each token in `tokenIds` should still be the correct `"msg.sender"`.

```
/// #if_succeeds {:msg "After withdrawing, the owner of each token in
`tokenIds` should still be the correct `"msg.sender"`."} tokenIds.length >
0 ==> forall (uint i in 0...tokenIds.length-1)
IERC404(mintswap404NFT).ownerOf(tokenIds[i]) == msg.sender;
function withdraw(uint256[] calldata tokenIds) external {
```

Passed.

15. The `withdrawStakeBenefits` function must ensure that the benefit withdrawn must be greater than or equal to MIN_WITHDRAW_AMOUNT.

```
/// #if_succeeds {:msg "The `withdrawStakeBenefits` function must ensure
that the benefit withdrawn must be greater than or equal to
MIN_WITHDRAW_AMOUNT."} benefit >= MIN_WITHDRAW_AMOUNT;
function withdrawStakeBenefits(
    uint256 _amount,
    bytes32 _r,
    bytes32 _s,
    uint8 _v
) external nonReentrant {
```

Passed.

16. The `withdrawStakeBenefits` function must verify that the request benefit amount does not exceed the user's available stake benefits.

```
/// #if_succeeds {:msg "The `withdrawStakeBenefits` function must verify
that the request benefit amount does not exceed the user's available stake
benefits."} old(userStakeBenefits[msg.sender]) >= benefit;
function withdrawStakeBenefits(
    uint256 _amount,
    bytes32 _r,
    bytes32 _s,
```

```
    uint8 _v
) external nonReentrant {
```

Passed.

17. After successful execution of `withdrawStakeBenefits`, the user's stake benefits should decrease by the amount of benefit.

```
/// #if_succeeds {:msg "After successful execution of
`withdrawStakeBenefits`, the user's stake benefits should decrease by the
amount of benefit."} old(userStakeBenefits[msg.sender]) ==
userStakeBenefits[msg.sender] + benefit;
function withdrawStakeBenefits(
    uint256 _amount,
    bytes32 _r,
    bytes32 _s,
    uint8 _v
) external nonReentrant {
```

Passed.