QuillAudits

Audit Report
April, 2022

For

Pandora

Contents

Scope of the Audit

The scope of this audit was to analyze and document the Pandora Finance smart contract codebase for quality, security, and correctness.

Checked Vulnerabilities

We have scanned the smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered:

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level

Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis
In this step, we have analysed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis
Static analysis of smart contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis
Manual analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analysed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption
In this step, we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit
Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis, Theo.

Issue Categories

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below.

Risk-level	Description
High	A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.
Medium	The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.
Low	Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.
Informational	These are severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Number of issues per severity

Type	High	Medium	Low	Informational
Open	0	0	0	0
Acknowledged	1	1	2	5
Closed	6	7	7	8

Introduction

On **Feb 3, 2022 - Mar 28, 2022** - QuillAudits Team performed a security audit for Pandora Finance smart contracts.

The code for the audit was taken from following the official link:
https://github.com/Pandora-Finance/Modular-contract/tree/main

V	Date	Commit ID
1	3rd Feb	909a567e961130e72214fde4dd2b68dc22b9ce02
2	28th Mar	d7cf31449b1c109a867bbd28d0830078155a9e9d

Issues Found – Code Review / Manual Testing

A. NFT Factory contract

High severity issues

No issues were found.

Medium severity issues

A.1 Missing zero check

Line	Function - SellNFT()

37	`function SellNFT_byBid(address _collectionAddress, uint256 _tokenId, uint256 _price, uint`
	`public`
	`nonReentrant`
	`{`
	`uint256 bal = ERC1155(_collectionAddress).balanceOf(msg.sender, _tokenId);`
	`require(bal >= _amount);`

Description

Missing zero address check for _collectionAddress parameter in the SellNFT_byBid() function. Also there is missing zero value check for _price and _bidTime parameters in the same function.

Remediation

Add a require check for the same.

Status: Acknowledged

Client's Comment: We have added require check for the collection address parameter. We can put _price & _bidTime check from frontend.

Low severity issues

A.2 Missing Error messages

Line	Function - BuyNFT

51-54, 31 and 107	`require(meta.status == true);`
	`require(msg.sender != address(0) && msg.sender != meta.currentOwner);`
	`require(meta.bidSale == false);`
	`require(msg.value >= meta.price);`

```
modifier onlyOwnerOfToken(address _collectionAddress, uint256 _tokenId) {
    require(msg.sender == ERC721(_collectionAddress).ownerOf(_tokenId));
    _;
}
```

```
107        require(msg.sender == _tokenMeta[_saleId].currentOwner);
```

Description

There are no error messages in the “require” statements that may lead to confusion on the client side.

Remediation

It is advised to add appropriate error messages.

Status: Acknowledged

Client's Comment: Due to constraints in contract size, we are going to add specific numeric code as error messages which points to error statements on the readme error table.

A.3 For Loop over Dynamic Array

Line	Code/Function

63	`for(uint256 i = 0; i < royalties.length; i ++) {`
	`  uint256 amount = (royalties[i].value * val) / 10000;`
	`  address payable receiver = royalties[i].account;`
	`  receiver.transfer(amount);`
	`  sum = sum - amount;`
	`}`

Description

Programming patterns that are harmless in centralized applications can lead to Denial of Service conditions in smart contracts when the cost of executing a function exceeds the block gas limit.

Remediation

Consider adding a require check to keep a check on the maximum size over which the for loop will run over.
Refer - https://swcregistry.io/docs/SWC-128

Status: Fixed

Client's Comment: We’ve put a require check to limit the royalties array size to max 10.

Informational Issues

A.4 Comparison with Boolean constant

Line	Code/Function

51 & 53	`require(meta.status == true);`
	`require(msg.sender != address(0) && msg.sender != meta.currentOwner);`
	`require(meta.bidSale == false);`
	`require(msg.value >= meta.price);`

Description
Comparison with Boolean constants is not required.

Remediation
The variable itself can be directly used instead of comparing with a boolean constant.
Refer - https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Status: Fixed

A.5 Transfer function used

Line	Code/Function

66, 70, 71	`receiver.transfer(amount);`
	`sum = sum - amount;`
	`}`
	
	`payable(meta.currentOwner).transfer(sum - fee);`
	`payable(feeAddress).transfer(fee);`

Description
The transfer() and send() functions forward a fixed amount of 2300 gas. Historically, it has often been recommended to use these functions for value transfers to guard against reentrancy attacks. However, the gas cost of EVM instructions may change significantly during hard forks which may break already deployed contract systems that make fixed assumptions about gas costs.

Remediation
Use call instead of transfer. **Refer** - https://swcregistry.io/docs/SWC-134

Status: Fixed

B. Bid contract

High severity issues

B.1 Equality operator use instead of Assignment Operator

Line	Code/Function
88	`Bids[_saleId][_bidOrderID].withdrawn == true;`

Description
Double equality sign has been used instead of single equality sign(i.e. assignment operator). Doing so would result in the value of that variable to be the same as before leading to unintended results such as withdrawing the amount again even though it had already been withdrawn.

Remediation
It is recommended to use assignment operator properly

Status: Fixed

B.2 Equality operator use instead of Assignment Operator

Line	Function - Bid
34	`_tokenMeta[_saleId].price == msg.value;`

Description
Double equality sign has been used instead of single equality sign(i.e. assignment operator). Doing so would result in the value of that variable to be the same as before leading to unintended results such as the price of the token never being set during bid.

Remediation
It is recommended to use assignment operator properly

Status: Fixed

Medium severity issues

B.3 Missing zero check

Line	Function - SellNFT_byBid()
37	`function SellNFT_byBid(address _collectionAddress, uint256 _tokenId, uint256 _price, uint` `public` `nonReentrant` `{` `uint256 bal = ERC1155(_collectionAddress).balanceOf(msg.sender, _tokenId);` `require(bal >= _amount);`

Description
Missing zero address check for _collectionAddress parameter in the SellNFT_byBid() function. Also there is missing zero value check for _price and _bidTime parameters in the same function.

Remediation
Add a require check for the same.

Status: Partially Fixed

Low severity issues

B.4 External calls made inside a Loop

Line	Function - executeBidOrder()
99	`for(uint256 i = 0; i < royalties.length; i ++) {` `uint256 amount = (royalties[i].value * Bids[_saleId][_bidOrderID].price) / 10000;` `address payable receiver = royalties[i].account;`

Description
In executeBidOrder() there are external calls made inside a loop. This could lead to Denial of Service and out of gas errors.

Remediation
Consider adding a require check to keep a check on the maximum size over which the for loop will run over.

Status: Fixed
Client's Comment: The maximum number of royalties can be 10, there's no need to add a require statement for checking the max iteration of the loop

B.5 Usage of block.timestamp

Line	Function - SellNFT_byBid()
19	19 require(block.timestamp <= _tokenMeta[_saleId].bidEndTime); 20 require(

Description

It should be noted that block.timestamp can give you a sense of the current time or a time delta, however, they are not safe to use for most purposes. It can be manipulated and altered by miners by upto 15 seconds.

Remediation

Usage of block.timestamp can lead to unexpected results. It is advised to use time based oracles instead.

Status: Acknowledged
Client's Comment: We've acknowledged this issue, will update the status once fixed

Informational Issues

B.6 Transfer and send functions used

Line	Code/Function
102, 106, 107, 122	102 receiver.transfer(amount); 103 sum = sum - amount; 104 } 105 106 payable(msg.sender).transfer(sum - fee); 107 payable(feeAddress).transfer(fee); 122 if (payable(msg.sender).send(Bids[_saleId][_bidId].price)) {

Description

The transfer() and send() functions forward a fixed amount of 2300 gas. Historically, it has often been recommended to use these functions for value transfers to guard against reentrancy attacks. However, the gas cost of EVM instructions may change significantly during hard forks which may break already deployed contract systems that make fixed assumptions about gas costs.

Remediation
Use call instead of transfer.
Refer- https://swcregistry.io/docs/SWC-134

Status: Fixed

B.7 Comparison with Boolean constant

Line	Code/Function
17&18	15 function Bid(uint256 _saleId↑) external payable { 16 require(_tokenMeta[_saleId↑].currentOwner != _msgSender()); 17 require(_tokenMeta[_saleId↑].status == true); 18 require(_tokenMeta[_saleId↑].bidSale == true); 19 require(block.timestamp <= _tokenMeta[_saleId↑].bidEndTime); 20 require(21 _tokenMeta[_saleId↑].price + ((5 * _tokenMeta[_saleId↑].price) / 100) <= 22 msg.value 23);
80&81	74 function executeBidOrder(uint256 _saleId↑, uint256 _bidOrderID↑) 75 external 76 nonReentrant 77 { 78 LibBid.BidOrder memory bids = Bids[_saleId↑][_bidOrderID↑]; 79 require(msg.sender == _tokenMeta[_saleId↑].currentOwner); 80 require(bids.withdrawn == false); 81 require(_tokenMeta[_saleId↑].status == true); 82

126	118 function withdrawBidMoney(uint256 _saleId↑, uint256 _bidId↑)
	119 external
	120 nonReentrant
	121 {
	122 require(msg.sender != _tokenMeta[_saleId↑].currentOwner);
	123 // BidOrder[] memory bids = Bids[_tokenId];
	124 LibBid.BidOrder memory bids = Bids[_saleId↑][_bidId↑];
	125 require(bids.buyerAddress == msg.sender);
	126 require(bids.withdrawn == false);
	127 (bool success,) = payable(msg.sender).call{
	128 value: bids.price

Description
Comparison with Boolean constants is not required.

Remediation
The variable itself can be directly used instead of comparing with a boolean constant.
Refer - https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Status: Fixed

C. Bid1155 contract

High severity issues

C.1 Equality operator use instead of Assignment Operator

Line	Function - Bid
86	`86 Bids[_saleId][_bidOrderID].withdrawn == true;`

Description
Double equality sign has been used instead of single equality sign(i.e. assignment operator). Doing so would result in the value of that variable to be the same as before leading to unintended results such as withdrawing the amount again even though it had already been withdrawn.

Remediation
It is recommended to use assignment operator properly

Status: Fixed

Medium severity issues

C.2 Missing zero check

Line	Function - SellNFT_byBid()
37	`function SellNFT_byBid(address _collectionAddress, uint256 _tokenId, uint256 _price, uint` `public` `nonReentrant` `{` `uint256 bal = ERC1155(_collectionAddress).balanceOf(msg.sender, _tokenId);` `require(bal >= _amount);`

Description
Missing zero address check for _collectionAddress parameter in the SellNFT_byBid() function. Also there are missing zero value checks for _price and _amount parameters in the same function.

Remediation
Add a require check for the same.

Status: Fixed

Low severity issues

C.3 External calls made inside a Loop

Line	Function - executeBidOrder()

100	`for(uint256 i = 0; i < royalties.length; i ++) {`
	`  uint256 amount = (royalties[i].value * Bids[_saleId][_bidOrderID].price) / 10000;`
	`  royalties[i].account.transfer(amount);`

Description
In executeBidOrder() there are external calls made inside a loop. This could lead to Denial of Service and out of gas errors.

Remediation
Consider adding a require check to keep a check on the maximum size over which the for loop will run over.

Status: Fixed
Client's Comment: The maximum number of royalties can be 10, there's no need to add a require statement for checking the max iteration of the loop.

Informational Issues

C.4 Transfer and send functions used

Line	Code/Function

100, 104, 105,115	`101  royalties[i].account.transfer(amount);`
	`102  sum = sum - amount;`
	`103  }`
	`104`
	`105  payable(msg.sender).transfer(sum - fee);`
	`106  payable(feeAddress).transfer(fee);`
	`107`
	`122  if (payable(msg.sender).send(Bids[_saleId][_bidId].price)) {`

Description

The transfer() and send() functions forward a fixed amount of 2300 gas. Historically, it has often been recommended to use these functions for value transfers to guard against reentrancy attacks. However, the gas cost of EVM instructions may change significantly during hard forks which may break already deployed contract systems that make fixed assumptions about gas costs.

Remediation

Use call instead of transfer. **Refer** - https://swcregistry.io/docs/SWC-134

Status: Fixed

C.5 Missing Revert statement

Line	Function - withdrawBidMoney

```
function withdrawBidMoney(uint256 _saleId, uint256 _bidId) public nonReentrant{
    require(
        Bids[_saleId][_bidId].buyerAddress == msg.sender
    );
    require(Bids[_saleId][_bidId].withdrawn == false);
    if (payable(msg.sender).send(Bids[_saleId][_bidId].price)) {
        Bids[_saleId][_bidId].withdrawn = true;
    }
}
```

Description

On failure of send in the withdrawBidMoney() function in NFTBid contract, there is revert statement but on the failure of send in the NFTBid1155 withdrawBidMoney() function, there is no revert statement.

Remediation

It is advised to add the corresponding revert statement for withdrawBidMoney() function in NFTBid1155 as well.

Status: Fixed

C.6 Comparison with Boolean constant

Line	Code/Function
15&16	13 function Bid(uint256 _saleId↑, uint256 _amount↑) external payable { 14 require(_tokenMeta[_saleId↑].currentOwner != msg.sender); 15 require(_tokenMeta[_saleId↑].status == true); 16 require(_tokenMeta[_saleId↑].bidSale == true); 17 require(msg.value % _amount↑ == 0); 18 require(msg.value / _amount↑ >= _tokenMeta[_saleId↑].price); 19 require(_tokenMeta[_saleId↑].numberOfTokens >= _amount↑); 20 21 LibBid1155.BidOrder memory bid = LibBid1155.BidOrder(
72&73	66 function executeBidOrder(uint256 _saleId↑, uint256 _bidOrderID↑) 67 external 68 nonReentrant 69 { 70 LibBid1155.BidOrder memory bids = Bids[_saleId↑][_bidOrderID↑]; 71 require(msg.sender == _tokenMeta[_saleId↑].currentOwner); 72 require(bids.withdrawn == false); 73 require(_tokenMeta[_saleId↑].status == true); 74 require(_tokenMeta[_saleId↑].numberOfTokens >= bids.numberOfTokens); 75

116	111 function withdrawBidMoney(uint256 _saleId↑, uint256 _bidId↑) external nonReentrant{
	112 LibBid1155.BidOrder memory bids = Bids[_saleId↑][_bidId↑];
	113 require(
	114 bids.buyerAddress == msg.sender
	115);
	116 require(bids.withdrawn == false);
	⚠ 117 (bool success,) = payable(msg.sender).call{
	118 value: bids.price

Description
Comparison with Boolean constants is not required.

Remediation
The variable itself can be directly used instead of comparing with a boolean constant.
Refer - https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Status: Fixed

D. NFTFactoryContract1155

High severity issues

No issues were found.

Medium severity issues

D.1 Missing zero check

Line	Function - sellNFT

75	`function sellNFT(address _collectionAddress, uint256 _tokenId, uint256 _price, uint2`
	`public`
	`nonReentrant`
	`{`
	`  uint256 bal = ERC1155(_collectionAddress).balanceOf(msg.sender, _tokenId);`
	`  require(bal >= _amount);`
	
	`  _tokenIdTracker.increment();`

Description

Missing zero address check for _collectionAddress parameter in the sellNFT() function. Also there is missing zero value check for _price and _amount parameters in the same function.

Remediation

Add a require check for the same

Status: Partially Fixed

Low severity issues

No issues were found.

Informational Issues

No issues were found.

E. Token Factory contract

High severity issues

No issues were found.

Medium severity issues

E.1 Missing zero check

Line	Function - initialize

15	`function initialize(address _address, address _feeAddress) initializer public {`
	`  PNDCAddress = _address;`
	`  NFTFactoryContract.initialize();`
	`  __UUPSUpgradeable_init();`
	`  feeAddress = _feeAddress;`
	`}`

Description

Missing zero address check for _address and _feeAddress parameters in the initialize() function.

Remediation

Add a require check for the same

Status: Fixed

E.2 Function failed to run

Line	Function - initialize

15	`function initialize(address _address, address _feeAddress) initializer public {`
	`  PNDCAddress = _address;`
	`  NFTFactoryContract.initialize();`
	`  __UUPSUpgradeable_init();`
	`  feeAddress = _feeAddress;`
	`}`

Description

initialize() function defined in the TokenFactory contract fails to run and the PNDC address and fee Address cannot be set.

Remediation
Refer - https://docs.openzeppelin.com/contracts/4.x/upgradeable#multiple-inheritance to resolve this issue.

Status: Fixed

Low severity issues

No issues were found.

Informational Issues

No issues were found.

F. TokenFactoryContract1155

High severity issues

No issues were found.

Medium severity issues

F.1 Missing zero check

Please refer to issue E1. The same exists in this contract on the same function - initialize()

Status: Fixed

F.2 Function failed to run

Line	Function - initialize

15	`function initialize(address _address, address _feeAddress) initializer public {`
	`  PNDC1155Address = _address;`
	`  NFTFactoryContract1155.initialize();`
	`  __UUPSUpgradeable_init();`
	`  feeAddress = _feeAddress;`
	`}`

Description
initialize() function defined in the TokenFactory contract fails to run and the PNDC1155 address and fee Address cannot be set.

Remediation
Refer - https://docs.openzeppelin.com/contracts/4.x/upgradeable#multiple-inheritance to resolve this issue.

Status: Fixed

Low severity issues

No issues were found.

Informational Issues

No issues were found.

G. TokenERC721 contract

High severity issues

G.1 Broken Access Control

Line	Code/Function

44	`function batchMint(`
	`  uint256 _totalNft,`
	`  string[] memory _uri,`
	`  RoyaltiesSet memory royaltiesSet`
	`) external {`
	`  require(_totalNft <= 15, "Minting more than 15 Nfts are not allowe");`
	`  require(`
	`  _totalNft == _uri.length,`
	`  "uri array length should be equal to _totalNFT"`
	`  );`
	`  for (uint256 i = 0; i < _totalNft; i++) {`
	`  safeMint(msg.sender, _uri[i], royaltiesSet);`
	`  }`
	`}`

Description

Anyone can call batchMint() function as there has been no access control modifier used.

Remediation

Use appropriate access control modifiers such as Openzeppelin onlyOwner or Roles

Status: Fixed

Medium severity issues

No issues were found.

Low severity issues

G.2 For loop over dynamic array

Line	Code/Function
100	`for (uint256 i = 0; i < royalties.length; i++) { require( royalties[i].account != address(0x0), "Royalty recipient should be present" ); require(royalties[i].value != 0, "Royalty value should be > 0"); royaltiesArr.push(royalties[i]); sumRoyalties += royalties[i].value; }`

Description

Programming patterns that are harmless in centralized applications can lead to Denial of Service conditions in smart contracts when the cost of executing a function exceeds the block gas limit.

Remediation

Consider adding a require check to keep a check on the maximum size over which the for loop will run over.
Refer- https://swcregistry.io/docs/SWC-128

Status: Fixed

Informational Issues

G.3 TokenERC721 contract does not need to inherit from ERC721, because it inherits ERC721Enumerable which already inherits from ERC721.

Status: Fixed

H. TokenERC1155 contract

High severity issues

H.1 Broken Access Control

Line	Code/Function - setTokenUri() and burn()

29&44	`function setTokenUri(string memory _uri, uint256 _tokenId) public {`
	`  _uris[_tokenId] = _uri;`
	`}`
	
	`function burn(address _from, uint256 _id, uint256 _amount) public {`
	`  require(balanceOf(_from, _id) >= _amount);`
	
	`  _burn(_from, _id, _amount);`
	`}`

Description

Anyone can call setTokenUri() function as there has been no access control modifier used and the burn() function also. An attacker can burn any NFTs that they do not own.

Remediation

Use appropriate access control modifiers such as Openzeppelin onlyOwner or Roles

Status: Fixed

Medium severity issues

No issues were found.

Low severity issues

H.2 For loop over dynamic array

Please refer to H.2 for Proof-of-concept, Description, and Remediation as both of these contracts have the same issue in the same function _setRoyaltiesArray()

Status: Fixed

Informational Issues

H.3 TokenERC1155 contract does not need to inherit from ERC1155, because it inherits ERC1155Supply which already inherits from ERC1155.

Status: Fixed

I. PNDC_ERC721 contract

High severity issues

I.1 Broken Access Control

Line	Code/Function - safeMint() and batchMint()

26 & 39	`function safeMint(`
	`  address to,`
	`  string memory uri,`
	`  LibShare.Share[] memory royalties`
	`) public returns(uint256){`
	
	`function batchMint(`
	`  uint256 _totalNft,`
	`  string[] memory _uri,`
	`  LibShare.Share[][] memory royaltiesSet`
	`) external {`

Description

Anyone can do batchMint() and anyone can do safeMint() as there has been no access control modifier used.

Remediation

Use appropriate access control modifiers such as Openzeppelin onlyOwner or Roles

Status: Acknowledged

Medium severity issues

No issues were found.

Low severity issues

I.2 For loop over dynamic array

Line	Code/Function - _setRoyaltiesByTokenId()

100	`function _setRoyaltiesByTokenId(`
	`  uint256 _tokenId,`
	`  LibShare.Share[] memory royalties`
	`) internal {`
	`  delete royaltiesByTokenId[_tokenId];`
	`  uint256 sumRoyalties = 0;`
	`  for (uint256 i = 0; i < royalties.length; i++) {`
	`  require(`
	`  royalties[i].account != address(0x0),`
	`  "Royalty recipient should be present"`
	`  );`

Description

Programming patterns that are harmless in centralized applications can lead to Denial of Service conditions in smart contracts when the cost of executing a function exceeds the block gas limit.

Remediation

Consider adding a require check to keep a check on the maximum size over which the for loop will run over.

Status: Fixed

Informational Issues

No issues were found.

J. PNDC_ERC1155 contract

High severity issues

J.1 Broken Access Control

Line	Code/Function - setTokenUri() and burn()

29 & 48	`function setTokenUri(string memory _uri, uint256 _tokenId) public {`
	`  _uris[_tokenId] = _uri;`
	`}`
	
	`function burn(address _from, uint256 _id, uint256 _amount) public {`
	`  require(balanceOf(_from, _id) >= _amount);`
	
	`  _burn(_from, _id, _amount);`
	`}`

Description

Anyone can do batchMint() and anyone can do safeMint() as there has been no access control modifier used.

Remediation

Use appropriate access control modifiers such as Openzeppelin onlyOwner or Roles

Status: Acknowledged

Medium severity issues

No issues were found.

Low severity issues

J.2 For loop over dynamic array

Line	Code/Function - _setRoyaltiesByTokenId()
100	`for (uint256 i = 0; i < royalties.length; i++) { require( royalties[i].account != address(0x0), "Royalty recipient should be present" );`

Description

Programming patterns that are harmless in centralized applications can lead to Denial of Service conditions in smart contracts when the cost of executing a function exceeds the block gas limit.

Remediation

Consider adding a require check to keep a check on the maximum size over which the for loop will run over.

Status: Fixed

Informational Issues

No issues were found.

K. Common issues for NFTFactoryContract, NFTFactoryContract1155, PNDC_ERC721, PNDC_ERC1155, TokenERC721 and TokenERC1155

High severity issues

No issues were found.

Medium severity issues

No issues were found.

Low severity issues

No issues were found.

Informational Issues

K.1 Transfer Ownership to undesired address

Description
Owner can accidently transfer ownership to an undesired address.

Remediation
It is advised to make transfer Ownership a two step process. Refer this post for additional info- Here

Status: Acknowledged

K.2 Renounce Ownership

Description
Owner can accidently renounce ownership and immediately lose control of all the privileged role functions

Remediation
It is advised that the Owner cannot call renounceOwnership without first transferring ownership to a different address. Alternatively, the Renounce Ownership functionality can be disabled by overriding it. Refer this post for additional info- Here

Status: Acknowledged

K.3 Upgradeable contracts are complex to maintain security wise. It can be seen that multiple inheritance has been used in contracts with a complex structure. It is advised to do thorough testing of the contracts including unit testing by the team.

Status: Acknowledged

K.4 Floating pragma is used across the codebase. It is advised to lock the solidity compiler version instead.

Status: Acknowledged

K.5 Do not leave an implementation contract uninitialized during final deployment. An uninitialized implementation contract can be taken over by an attacker, which may impact the proxy. You can either invoke the initializer manually, or you can include a constructor to automatically mark it as initialized when it is deployed:

```
/// @custom:oz-upgrades-unsafe-allow constructor
constructor() initializer {}
```

Status: Acknowledged

Automated Tests

Slither

```
ERC1967UpgradeUpgradeable._functionDelegateCall(address,bytes) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#208-214) uses delegatecall
 to a input-controlled function id
        - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#212)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#controlled-delegatecall

OwnableUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#82) shadows:
        - ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#31)
UUPSUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol#81) shadows:
        - ERC1967UpgradeUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#215)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing

NFTV1Storage.PNDCAddress (contracts/NFTStorage.sol#19) is never initialized. It is used in:
        - NFTFactoryContract.BuyNFT(uint256) (contracts/NFTFactoryContract.sol#38-74)
        - NFTBid.executeBidOrder(uint256,uint256) (contracts/Auction/Bid.sol#69-110)
NFTV1Storage.feeAddress (contracts/NFTStorage.sol#20) is never initialized. It is used in:
        - NFTFactoryContract.BuyNFT(uint256) (contracts/NFTFactoryContract.sol#38-74)
        - NFTBid.executeBidOrder(uint256,uint256) (contracts/Auction/Bid.sol#69-110)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables

ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#382-403) ignores return value by IERC721Receiver(to).onERC72
1Received(_msgSender(),from,tokenId,_data) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#389-399)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

PNDC_ERC721.constructor(string,string).name (contracts/PNDC_ERC721.sol#22) shadows:
        - ERC721.name() (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#79-81) (function)
        - IERC721Metadata.name() (node_modules/@openzeppelin/contracts/token/ERC721/extensions/IERC721Metadata.sol#16) (function)
PNDC_ERC721.constructor(string,string).symbol (contracts/PNDC_ERC721.sol#22) shadows:
        - ERC721.symbol() (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#86-88) (function)
        - IERC721Metadata.symbol() (node_modules/@openzeppelin/contracts/token/ERC721/extensions/IERC721Metadata.sol#21) (function)
TokenERC721.constructor(string,string).name (contracts/TokenERC721.sol#27) shadows:
        - ERC721.name() (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#79-81) (function)
        - IERC721Metadata.name() (node_modules/@openzeppelin/contracts/token/ERC721/extensions/IERC721Metadata.sol#16) (function)
TokenERC721.constructor(string,string).symbol (contracts/TokenERC721.sol#27) shadows:
        - ERC721.symbol() (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#86-88) (function)
        - IERC721Metadata.symbol() (node_modules/@openzeppelin/contracts/token/ERC721/extensions/IERC721Metadata.sol#21) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

NFTBid.executeBidOrder(uint256,uint256) (contracts/Auction/Bid.sol#69-110) has external calls inside a loop: receiver.transfer(amount) (contracts/Auction/Bid.sol#102)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).retval (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#389)' in ERC721._checkOnERC721Received(addr
ess,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#382-403) potentially used before declaration: retval == IERC721Receiver.onERC721Received.selec
tor (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#390)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#391)' in ERC721._checkOnERC721Received(addr
ess,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#382-403) potentially used before declaration: reason.length == 0 (node_modules/@openzeppelin/c
ontracts/token/ERC721/ERC721.sol#392)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#391)' in ERC721._checkOnERC721Received(addr
ess,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#382-403) potentially used before declaration: revert(uint256,uint256)(32 + reason,mload(uint25
6)(reason)) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#396)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables
        - _setRoyaltiesByTokenId(tokenId,royalties) (contracts/PNDC_ERC721.sol#35)
                - delete royaltiesByTokenId[_tokenId] (contracts/PNDC_ERC721.sol#64)
                - i < royalties.length (contracts/PNDC_ERC721.sol#66)
                - royaltiesByTokenId[_tokenId].push(royalties[i]) (contracts/PNDC_ERC721.sol#72)
Reentrancy in TokenERC721.safeMint(address,string,TokenERC721.RoyaltiesSet) (contracts/TokenERC721.sol#31-42):
        External calls:
        - _safeMint(to,tokenId) (contracts/TokenERC721.sol#38)
                - IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,_data) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#389-399)
        State variables written after the call(s):
        - _setTokenURI(tokenId,uri) (contracts/TokenERC721.sol#39)
                - _tokenURIs[tokenId] = _tokenURI (node_modules/@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol#47)
        - setRoyaltiesByTokenId(tokenId,royaltiesSet) (contracts/TokenERC721.sol#40)
                - delete royaltiesByTokenId[_tokenId] (contracts/TokenERC721.sol#69)
                - royaltiesByTokenId[_tokenId].set = royaltiesSet.set (contracts/TokenERC721.sol#70)
Reentrancy in NFTFactoryContract.sellNFT(address,uint256,uint256) (contracts/NFTFactoryContract.sol#76-103):
        External calls:
        - ERC721(_collectionAddress).safeTransferFrom(msg.sender,address(this),_tokenId) (contracts/NFTFactoryContract.sol#84)
        State variables written after the call(s):
        - _tokenMeta[_tokenIdTracker.current()] = meta (contracts/NFTFactoryContract.sol#99)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in NFTBid.SellNFT_byBid(address,uint256,uint256,uint256) (contracts/Auction/Bid.sol#39-67):
        External calls:
        - ERC721(_collectionAddress).safeTransferFrom(msg.sender,address(this),_tokenId) (contracts/Auction/Bid.sol#48)
        Event emitted after the call(s):
        - TokenMetaReturn(meta,_tokenIdTracker.current()) (contracts/Auction/Bid.sol#65)
Reentrancy in ERC1967UpgradeUpgradeable._upgradeToAndCallSecure(address,bytes,bool) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#87-11
6):
        External calls:
        - _functionDelegateCall(newImplementation,data) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#97)
                - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#212)
        - _functionDelegateCall(newImplementation,abi.encodeWithSignature(upgradeTo(address),oldImplementation)) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC196
7UpgradeUpgradeable.sol#105-108)
                - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#212)
        Event emitted after the call(s):
        - Upgraded(newImplementation) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#63)
                - _upgradeTo(newImplementation) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#113)
Reentrancy in NFTBid.executeBidOrder(uint256,uint256) (contracts/Auction/Bid.sol#69-110):
        External calls:
        - ERC721(_tokenMeta[_saleId].collectionAddress).safeTransferFrom(address(this),Bids[_saleId][_bidOrderID].buyerAddress,_tokenMeta[_saleId].tokenId) (contracts/Auction/Bid.sol#
90-94)
        External calls sending eth:
        - receiver.transfer(amount) (contracts/Auction/Bid.sol#102)
        - address(msg.sender).transfer(sum - fee) (contracts/Auction/Bid.sol#106)
        - address(feeAddress).transfer(fee) (contracts/Auction/Bid.sol#107)
        Event emitted after the call(s):
        - BidExecuted(Bids[_saleId][_bidOrderID].price) (contracts/Auction/Bid.sol#109)
Reentrancy in PNDC_ERC721.safeMint(address,string,LibShare.Share[]) (contracts/PNDC_ERC721.sol#26-37):
        External calls:
        - _safeMint(to,tokenId) (contracts/PNDC_ERC721.sol#33)
                - IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,_data) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#389-399)
        Event emitted after the call(s):
        - RoyaltiesSetForTokenId(_tokenId,royalties) (contracts/PNDC_ERC721.sol#77)
```

```
AddressUpgradeable.isContract(address) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#27-37) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#33-35)
AddressUpgradeable.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#169-189) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#181-184)
StorageSlotUpgradeable.getAddressSlot(bytes32) (node_modules/@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#52-56) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#53-55)
StorageSlotUpgradeable.getBooleanSlot(bytes32) (node_modules/@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#61-65) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#62-64)
StorageSlotUpgradeable.getBytes32Slot(bytes32) (node_modules/@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#70-74) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#71-73)
StorageSlotUpgradeable.getUint256Slot(bytes32) (node_modules/@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#79-83) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#80-82)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#382-403) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#395-397)
Address.isContract(address) (node_modules/@openzeppelin/contracts/utils/Address.sol#27-37) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#33-35)
Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#196-216) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#208-211)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

NFTBid.Bid(uint256) (contracts/Auction/Bid.sol#15-37) compares to a boolean constant:
        -require(bool)(_tokenMeta[_saleId].bidSale == true) (contracts/Auction/Bid.sol#18)
NFTBid.Bid(uint256) (contracts/Auction/Bid.sol#15-37) compares to a boolean constant:
        -require(bool)(_tokenMeta[_saleId].status == true) (contracts/Auction/Bid.sol#17)
NFTBid.executeBidOrder(uint256,uint256) (contracts/Auction/Bid.sol#69-110) compares to a boolean constant:
        -Bids[_saleId][_bidOrderID].withdrawn == true (contracts/Auction/Bid.sol#88)
NFTBid.executeBidOrder(uint256,uint256) (contracts/Auction/Bid.sol#69-110) compares to a boolean constant:
        -require(bool)(_tokenMeta[_saleId].status == true) (contracts/Auction/Bid.sol#75)
NFTBid.executeBidOrder(uint256,uint256) (contracts/Auction/Bid.sol#69-110) compares to a boolean constant:
        -require(bool)(Bids[_saleId][_bidOrderID].withdrawn == false) (contracts/Auction/Bid.sol#74)
NFTBid.withdrawBidMoney(uint256,uint256) (contracts/Auction/Bid.sol#112-127) compares to a boolean constant:
        -require(bool)(Bids[_saleId][_bidId].withdrawn == false) (contracts/Auction/Bid.sol#121)
NFTFactoryContract.BuyNFT(uint256) (contracts/NFTFactoryContract.sol#38-74) compares to a boolean constant:
        -require(bool)(meta.status == true) (contracts/NFTFactoryContract.sol#51)
NFTFactoryContract.BuyNFT(uint256) (contracts/NFTFactoryContract.sol#38-74) compares to a boolean constant:
        -require(bool)(meta.bidSale == false) (contracts/NFTFactoryContract.sol#53)
NFTFactoryContract.cancelSale(uint256) (contracts/NFTFactoryContract.sol#105-118) compares to a boolean constant:
        -require(bool)(_tokenMeta[_saleId].status == true) (contracts/NFTFactoryContract.sol#108)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Different versions of Solidity is used:
        - Version used: ['^0.8.0', '^0.8.2']
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#4)
        - ^0.8.2 (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/beacon/IBeaconUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/IERC721ReceiverUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/utils/ERC721HolderUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#4)
```

```
PNDC_ERC1155.constructor(string).uri (contracts/PNDC_ERC1155.sol#23) shadows:
        - ERC1155.uri(uint256) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#59-61) (function)
        - IERC1155MetadataURI.uri(uint256) (node_modules/@openzeppelin/contracts/token/ERC1155/extensions/IERC1155MetadataURI.sol#21) (function)
PNDC_ERC1155.setTokenUri(string,uint256)._uri (contracts/PNDC_ERC1155.sol#29) shadows:
        - ERC1155._uri (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#30) (state variable)
PNDC_ERC1155.mint(address,uint256,bytes,string,LibShare.Share[]).uri (contracts/PNDC_ERC1155.sol#37) shadows:
        - ERC1155.uri(uint256) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#59-61) (function)
        - IERC1155MetadataURI.uri(uint256) (node_modules/@openzeppelin/contracts/token/ERC1155/extensions/IERC1155MetadataURI.sol#21) (function)
TokenERC1155.constructor(string).uri (contracts/TokenERC1155.sol#22) shadows:
        - ERC1155.uri(uint256) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#59-61) (function)
        - IERC1155MetadataURI.uri(uint256) (node_modules/@openzeppelin/contracts/token/ERC1155/extensions/IERC1155MetadataURI.sol#21) (function)
TokenERC1155.setTokenUri(string,uint256)._uri (contracts/TokenERC1155.sol#29) shadows:
        - ERC1155._uri (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#30) (state variable)
TokenERC1155.mint(address,uint256,uint256,string,bytes)._uri (contracts/TokenERC1155.sol#37) shadows:
        - ERC1155._uri (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#30) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

NFTBid1155.executeBidOrder(uint256,uint256) (contracts/Auction/Bid1155.sol#65-108) has external calls inside a loop: royalties[i].account.transfer(amount) (contracts/Auction/Bid1155.s
ol#100)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Variable 'ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).response (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#423)' in ERC11
55._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#414-433) potentially used before decla
ration: response != IERC1155Receiver.onERC1155Received.selector (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#424)
Variable 'ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes).reason (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#427)' in ERC1155
._doSafeTransferAcceptanceCheck(address,address,address,uint256,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#414-433) potentially used before declara
tion: revert(string)(reason) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#428)
Variable 'ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).response (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#445)'
 in ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#435-456) potentially
 used before declaration: response != IERC1155Receiver.onERC1155BatchReceived.selector (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#447)
Variable 'ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes).reason (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#450)' i
n ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,uint256[],uint256[],bytes) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#435-456) potentially u
sed before declaration: revert(string)(reason) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#451)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

Reentrancy in NFTBid1155.SellNFT_byBid(address,uint256,uint256,uint256) (contracts/Auction/Bid1155.sol#36-63):
        External calls:
        - ERC1155(_collectionAddress).safeTransferFrom(msg.sender,address(this),_tokenId,_amount,) (contracts/Auction/Bid1155.sol#46)
        State variables written after the call(s):
        - _tokenMeta[_tokenIdTracker.current()] = meta (contracts/Auction/Bid1155.sol#59)
Reentrancy in PNDC_ERC1155.mint(address,uint256,bytes,string,LibShare.Share[]) (contracts/PNDC_ERC1155.sol#33-46):
        External calls:
        - _mint(account,tokenId,amount,data) (contracts/PNDC_ERC1155.sol#42)
                - IERC1155Receiver(to).onERC1155Received(operator,from,id,amount,data) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#423-431)
        State variables written after the call(s):
        - setTokenUri(uri,tokenId) (contracts/PNDC_ERC1155.sol#44)
                - _uris[_tokenId] = _uri (contracts/PNDC_ERC1155.sol#30)
        - setRoyaltiesByTokenId(tokenId,royalties) (contracts/PNDC_ERC1155.sol#43)
```

```
        Event emitted after the call(s):
        - Upgraded(newImplementation) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#63)
                - _upgradeTo(newImplementation) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#113)
Reentrancy in NFTBid1155.executeBidOrder(uint256,uint256) (contracts/Auction/Bid1155.sol#65-108):
        External calls:
        - ERC1155(_tokenMeta[_saleId].collectionAddress).safeTransferFrom(address(this),Bids[_saleId][_bidOrderID].buyerAddress,_tokenMeta[_saleId].tokenId,Bids[_saleId][_bidOrderID].
numberOfTokens,) (contracts/Auction/Bid1155.sol#87-93)
        External calls sending eth:
        - royalties[i].account.transfer(amount) (contracts/Auction/Bid1155.sol#100)
        - address(msg.sender).transfer(sum - fee) (contracts/Auction/Bid1155.sol#104)
        - address(feeAddress).transfer(fee) (contracts/Auction/Bid1155.sol#105)
        Event emitted after the call(s):
        - BidExecuted(Bids[_saleId][_bidOrderID].price) (contracts/Auction/Bid1155.sol#107)
Reentrancy in PNDC_ERC1155.mint(address,uint256,bytes,string,LibShare.Share[]) (contracts/PNDC_ERC1155.sol#33-46):
        External calls:
        - _mint(account,tokenId,amount,data) (contracts/PNDC_ERC1155.sol#42)
                - IERC1155Receiver(to).onERC1155Received(operator,from,id,amount,data) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol#423-431)
        Event emitted after the call(s):
        - RoyaltiesSetForTokenId(_tokenId,royalties) (contracts/PNDC_ERC1155.sol#71)
                - _setRoyaltiesByTokenId(tokenId,royalties) (contracts/PNDC_ERC1155.sol#43)
Reentrancy in NFTFactoryContract1155.sellNFT(address,uint256,uint256,uint256) (contracts/NFTFactoryContract1155.sol#75-102):
        External calls:
        - ERC1155(_collectionAddress).safeTransferFrom(msg.sender,address(this),_tokenId,_amount,) (contracts/NFTFactoryContract1155.sol#85)
        Event emitted after the call(s):
        - TokenMetaReturn(meta,_tokenIdTracker.current()) (contracts/NFTFactoryContract1155.sol#100)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

AddressUpgradeable.isContract(address) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#27-37) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#33-35)
AddressUpgradeable.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#169-189) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#181-184)
StorageSlotUpgradeable.getAddressSlot(bytes32) (node_modules/@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#52-56) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#53-55)
StorageSlotUpgradeable.getBooleanSlot(bytes32) (node_modules/@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#61-65) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#62-64)
StorageSlotUpgradeable.getBytes32Slot(bytes32) (node_modules/@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#70-74) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#71-73)
StorageSlotUpgradeable.getUint256Slot(bytes32) (node_modules/@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#79-83) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#80-82)
Address.isContract(address) (node_modules/@openzeppelin/contracts/utils/Address.sol#27-37) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#33-35)
Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#196-216) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol#208-211)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

NFTBid1155.Bid(uint256,uint256) (contracts/Auction/Bid1155.sol#15-34) compares to a boolean constant:
        -require(bool)(_tokenMeta[_saleId].status == true) (contracts/Auction/Bid1155.sol#17)
NFTBid1155.Bid(uint256,uint256) (contracts/Auction/Bid1155.sol#15-34) compares to a boolean constant:
        -require(bool)(_tokenMeta[_saleId].bidSale == true) (contracts/Auction/Bid1155.sol#18)
NFTBid1155.executeBidOrder(uint256,uint256) (contracts/Auction/Bid1155.sol#65-108) compares to a boolean constant:
        -Bids[_saleId][_bidOrderID].withdrawn == true (contracts/Auction/Bid1155.sol#85)
NFTBid1155.executeBidOrder(uint256,uint256) (contracts/Auction/Bid1155.sol#65-108) compares to a boolean constant:
        -require(bool)(_tokenMeta[_saleId].status == true) (contracts/Auction/Bid1155.sol#71)
NFTBid1155.executeBidOrder(uint256,uint256) (contracts/Auction/Bid1155.sol#65-108) compares to a boolean constant:
```

Mythril

No issues were reported by Mythril.

Closing Summary

Numerous issues of high,medium and low severity were discovered during the initial audit. At the end, most of the issues were Fixed, and some were acknowledged by the Pandora Finance team.

Disclaimer

QuillAudits smart contract audit is not a security warranty, investment advice, or an endorsement of the Pandora Finance platform. This audit does not provide a security or correctness guarantee of the audited smart contracts.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the Pandora Finance Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

Audit Report
April, 2022

For

Pandora

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com