QuillAudits

Audit Report
April, 2023

For

NFJ LABS.

Table of Content

Executive Summary

Project Name	NFJ Labs
Overview	NFJ labs contracts creates an atmosphere that allows users to mint, auction, sell and burn their nfts. The NFT contract is an ERC721 upgradable contract designed with royalty features. The NFT contract identifies three unique persons; Contract Owner, NFTOwner, and NFTCreator - each are given privileges such as setting platform fee derived on all trade on the NFJ platform, mint and burn nft and setting royalty fee respectively. In the marketplace contract, token holders can put nfts for auction, cancel nft after auction end time, and these nfts can also be put on sale for buyers.
Timeline	February 6, 2023 - February 10, 2023
Method	Manual Review, Functional Testing, Automated Testing etc.
Scope of Audit	The scope of this audit was to analyze NFJ Labs ArtisticJewellers and Marketplace codebase for quality, security, correctness, and exchange listing.
	https://github.com/ArtisticJewellers/nfjlabs-marketplace/tree/master/contracts/contractsVerify “master” branch with commit hash 8c260abe0693ec08f0a62bd3ed881b262dd6337a
Fixed In	196a4fcf0dc5477dcd525f68e7949bb5ce4a279c

Executive Summary

Number of security issues per severity

	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	2	1	7
Partially Resolved Issues	0	0	0	0
Resolved Issues	2	1	1	2

Types of Severities

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.

Types of Issues

Open

Security vulnerabilities identified that must be resolved and are currently unresolved.

Resolved

These are the issues identified in the initial audit and have been successfully fixed.

Acknowledged

Vulnerabilities which have been acknowledged but are yet to be resolved.

Partially Resolved

Considerable efforts have been invested to reduce the risk/impact of the security issue, but are not completely resolved.

Checked Vulnerabilities

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- Exception Disorder
- Gasless Send
- Use of tx.origin
- Compiler version not fixed
- Address hardcoded
- Divide before multiply
- Integer overflow/underflow
- Dangerous strict equalities
- Tautology or contradiction
- Return values of low-level calls
- Missing Zero Address Validation
- Private modifier
- Revert/require functions
- Using block.timestamp
- Multiple Sends
- Using SHA3
- Using suicide
- Using throw
- Using inline assembly

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.

Manual Testing

A. Contract - ArtisticJeweller.sol

High Severity Issues

No issues found

Medium Severity Issues

A.1: Improper Use of beforeTokenTransfer Hook with PausableUpgradable.sol WhenNotPaused Modifier

```
ftrace | funcSig
function _beforeTokenTransfer(
    address from,
    address to,
    uint256 tokenId
) internal override whenNotPaused {
    super._beforeTokenTransfer(from, to, tokenId);
}
```

During compilation, the codebase breaks at the level of the _beforeTokenTransfer hook and this makes it impossible to deploy the contract. Moreso, it was discovered that the hook was not implemented the proper way. The primary motive of the hook is to allow developers to make additional code implementation on what could happen before the move of the tokens from one address to another.
Although the override feature was added to the function and the whenNotPaused modifier from PausableUpgradable.sol, the override was not effective as the compiler throws an error of “function has override specified, it does not override anything”. This implies that the modification on the beforeTokenTransfer function can be effective when it is implemented within it. Also, parameters being passed to the super._beforeTokenTransfer expected four parameters but got three.

Recommendation

Look into applying the modification to the _beforeTokenTransfer within the function and ensure to pass exact expected parameters so the pause() and unpause() features will work.

Status

Resolved

Refer: *Link 1* | *Link 2* | *Link 3* | *Link 4*

A.2: Allow Setting Unchecked Arbitrary Values

```
ftrace | funcSig
39  function initialize(address _platformReciverAddr⬆, uint128 _platformCuts⬆)
40      public
41      initializer
42  {
43      __ERC721_init("ArtisticJeweller", "AJ");
44      __Pausable_init();
45      __Ownable_init();
46      __ERC721Burnable_init();
47      __UUPSUpgradeable_init();
48
49      PlatformFee = _platformCuts⬆;
50      PlatformAddress = payable(_platformReciverAddr⬆);
51  }
```

```
135 // only Admin can update platform details
    ftrace | funcSig
136 function updatePlatformDetails(
137     address _platformReciverAddr⬆,
138     uint128 _platformCuts⬆
139 ) public onlyOwner {
140     PlatformFee = _platformCuts⬆;
141     PlatformAddress = payable(_platformReciverAddr⬆);
142 }
143
144 // only NFT Creator can change NFT Royalty Detials
    ftrace | funcSig
145 function updateNFTRoyalty(
146     uint256 tokenIdCounter⬆,
147     uint128 _royaltyFee⬆,
148     address _royaltyAddress⬆
149 ) public NFTCreator(tokenIdCounter⬆) {
150     storeData[tokenIdCounter⬆].nftRoyalty = _royaltyFee⬆;
151     storeData[tokenIdCounter⬆].royaltyAddress = _royaltyAddress⬆;
152 }
```

Description

In the revenue sources section of NFJ labs whitepaper, it states that the intended royalty for every nft minted on their platform will be 2.5% derivation on a trade. The allocation that goes to the platform was also cited in the whitepaper to be 10% on all trades made on an NFT. However, this business logic is not exactly captured in the contract. The present design architecture allows platform owners and nft creators to set platformFee and royalties to arbitrary values. Malicious NFT creators can mint on the platform and hike royalties so high in order to get a proportion of the funds greater than 2.5%.

The following critical functions do not have input validation checks

- initialize(),
- mintNFT(),
- updatePlatformDetails(),
- updateNFTRoyalty()

Remediation

Consider mitigating these possible actions by setting a limit and this is possible using basis points. Before critical changes to the contract state, checks can be made; for example: to confirm the previous value is not the same as the new one, zero address checks, integer over or underflows. These checks should happen at the beginning of the functions to ensure reverts happen before excessive gas is consumed. It is necessary to sanitize input to avoid breaks in code because of unexpected values.

Status

Acknowledged

NFJ Team Comment: These sets of functions that give privileges to the NFT creators and the platform owner are handled from the front end interface to regulate setting arbitrary values for the NFT royalty.

Refer: *Link 1*

Low Severity Issues

No issues found

Informational Issues

A.3: Unnecessary Check In mintNFT Function

```
110        require(msg.sender != address(0), "Zero address");
```

Description

The require check on L98 is unnecessary because the zero address can't be used to sign transactions.

Remediation

It is recommended to remove this check since the null address cannot be associated with any signer for transaction.

Status

Resolved

A.4: Remove Unused Variable

```
21      uint128 public constant MAX_BPS = 10_000;
```

Description

To maintain clean code, it is required to remove unused variables in the contract. Although MAX_BPS being a constant variable, doesn't add up to the contract bytecode but this variable was never used within the contract.

Remediation

It is recommended to remove this kind of variable that was not used within the contract.

Status

Acknowledged

NFJ Team Comment: We want to keep this variable in our code

A.5: Change Function Visibility from Public to External

Description

In contract, some view functions were created to return some state variables but these functions were not called within the contract. The functions are as follows:
getTotalNFTCount
getRoyaltyAndPlatformFeeDetails
The visibility of the function is public. However, it is not being called in the contract. It is only called externally. Hence it is recommended to declare such functions as external in order to save gas.

Remediation

Consider changing the functions visibility label from public to external.

Status

Acknowledged

A.6: Lack of Comments

Description

The codebase could provide more detail with descriptions and comments. It is generally good practice to have inline comments to make code easily readable.

Remediation

Consider including more comments in the codebase, more specifically using the Natspec format is recommended. This model of code comment makes it easier to comprehend the motives of functions and its expected parameters.

Status

Acknowledged

A.7: Emit events for notable changes to state

Description

When important changes are made, it is advisable to log these changes by emitting events - usually with indexed parameters to enable easy searching. The following functions do not log changes updateNFTJsonData, updatePlatformDetails, and updateNFTRoyalty.

Remediation

Events can be emitted to keep track of these changes.

Status

Acknowledged

A.8: Modifier only used once

Description

It is common practice to use modifiers to reduce repetitive code that cuts across multiple functions in a contract/codebase. The NFTCreator() modifier is only called once through the scope of the entire codebase.

Remediation

Consider inlining the modifier code in the single function it is used in.

Status

Acknowledged

A.9: Inaccurate Code Behavior

Description

There is no specified pragma solidity version in the contract, and the OpenZeppelin Proxy contract details the need to have this snippet to compile the contract.

/// @custom:oz-upgrades-unsafe-allow constructor

Remediation

Consider updating the code to match up with recommendations made by external contracts and dependencies for optimal functionality.

Status

Acknowledged

B. Contract - Marketplace.sol

High Severity Issues

B.1: Platform owner can steal any owner's NFTs: *POC*

Description

Tokens put up for auction can only be cancelled at any time, even when the auction timer is over. The token owner would not be the recipient of the NFT if the platformOwner calls cancelAuction() before they can. The platformOwner can take ownership of tokens this way.

Recommendation

If this is not expected functionality, consider updating the logic to not allow this happen.

```
function cancelAuction(uint256 _nftId) external nonReentrant {
    ...
    require(((msg.sender == owner) || (msg.sender == nftOwner)), "Only NFT owner or Platform Owner can cancel Auction.
// transfer nft to seller
    IERC721(nftContractAddr).safeTransferFrom(
        address(this),
        msg.sender, // adjust this to nftOwner
        _nftId
    );
```

Status

Resolved

B.2: Bidders Lose Funds On Bidding for Auctioned NFTs

```
require(
    _payAmount >= totalAmountUserPay,
    "Not enough ether to cover item price and market fee, Price + Platfrom Fee"
);

// ether transfer to the seller
IWETH(WETH).transferFrom(msg.sender, address(this), _payAmount);

AuctionDataset[_nftId].totalBidAmount += itemActualPrice;
AuctionDataset[_nftId].highestBid = itemActualPrice;
AuctionDataset[_nftId].highestBidder = msg.sender;
```

```
        } else {
            if (
                (allBidersAddress[_nftId][auctionIndex].biddersAddr[
                    i
                ] != address(0))
            ) {
                uint256 NftPrice = AuctionDataset[_nftId].highestBid;
                // divide ether value in 2 part
                uint256 etherValue_platform = (NftPrice *
                    (platformFee)) / IERC721(nftContractAddr).MAX_BPS();
                uint256 etherValue_royalty = (NftPrice *
                    (royaltyPercent)) /
                    IERC721(nftContractAddr).MAX_BPS();
                // Distribute on 2 different address
                // Transfer Platform Fee
                IWETH(WETH).transfer(platformAddr, etherValue_platform);
                // Transfer Royalty Fee
                IWETH(WETH).transfer(royaltyAddr, etherValue_royalty);
            }
```

Description

The bid function takes two parameters; the nft token id and the amount a bidder is willing to bid. While this function does proper checks that encompasses knowing if a bidder had made a bid before, reverts when seller attempts to bid on their own auctioned nfts, check when an auction has started and the auction time interval. However, there are three parameters to pay attention to, to understand how funds are lost in contract. These parameters are _payAmount, totalAmountUserPay, and itemActualPrice - amount being sent into the contract by the bidder, amount derived from the getNFTFinalRate() function, and amount derived from dividing the result of _payAmount multiply by the max basis point (MAX_BPS) by the result of the addition of the MAX_BPS and platformFee. While the value of itemActualPrice aids in ensuring that the new bidder makes a bid greater than the last bidder, other values ensure that _payAmount exceeds totalAmountUserPay.

Scenario

If a person bids and sends 10 wrapped ether, with the MAX_BPS being 10000 and the platformFee is 10%, represented in basis point as 1000.
itemActualPrice = (10000000000000000000 * 10000) / (1000 + 10000) = 909090909090909090.
Funds lost in contract = 10000000000000000000 - 909090909090909090 = 90909090909090200
The value assigned to the value of the highest bid in the auction dataset struct; the value of itemActualPrice is what is set. This becomes the value being used to compute the etherValue_platform and etherValue_royalty.

Remediation

Consider returning the remnant to the user or update the highest bid value in the trust to the precise amount being sent by user to avoid losing funds to the contract.

Status

Resolved

Medium Severity Issues

B.3: NFT Item Put On Sale Cannot Be Removed Unless Purchased: *POC1*/*POC2*

```
ftrace | funcSig
473     function removeFromSale(uint256 _tokenId⬆) external nonReentrant {
474         Item storage item = items[_tokenId⬆];
475         require(
476             address(item.seller) == msg.sender,
477             "Only Seller can remove the NFT from sale"
478         );
479         require(
480             block.timestamp < AuctionDataset[_tokenId⬆].endAt,
481             "Seller cannot remove nft from auction after end time"
482         );
483         IERC721(nftContractAddr).transferFrom(
484             address(this),
485             msg.sender,
486             item.tokenId
487         );
488         items[_tokenId⬆] = Item(0, address(0), 0, payable(address(0)), false);
489     }
```

Description

The marketplace contract allows nfts holders to put nft items for auctioning and for sales. When any nft item is put on sale by the token holder, it is not possible for the token owner to remove these nfts again even with the removeFromSale function unless purchased. The removeFromSale function has a line of code that disallows removal from contract after putting on sale.

Scenario 1: As a token holder, say you go on the platform to auction your nft and of course, it is impossible to remove nft until the auction period elapses. When the auction time elapses, the seller, being you the token owner, calls the transferNFTToHigherBidder. This gives the highest bidder the nft. If the highest bidder, the new token owner for that tokenId, decided to putOnSale the nft.

If there is no buyer for this nft put on sale, it is impossible to remove that nft for life as it gets stuck in contract.

```
function removeFromSale(uint256 _tokenId) external nonReentrant {
    ...
    require(block.timestamp < AuctionDataset[_tokenId].endAt, ...);
    ...
}
```

The end time for this nft has formerly elapsed, there will never be a time when block.timestamp is greater than AuctionDataset[_tokenId].endAt.

Scenario 2: On acquiring an nft from the token contract, the token holder decided to directly putOnSale the nft item on the marketplace without an auction. The end time for this item is zero because there's never a time when AuctionDataset[_tokenId].endAt for this particular item, it's impossible to retrieve the item with the removeFromSale function.

Remediation

It is recommended that using a different timeline, aside from the auctioning endTime, to target when items put on sale can be removed. Adopting the separation of concern mechanism will help avert this issue.

Status

Resolved

Low Severity Issues

B.4: Input validation

```
constructor(address _nftContractAddr, address _weth) {
    nftContractAddr = _nftContractAddr;
    WETH = _weth;
    owner = msg.sender;
}
```

Description

Zero addresses can be passed in without reverting, and any arbitrary amount can go in for platformFee as well. It is necessary to sanitize input to avoid breaks in code because of unexpected values.

Recommendation

Before critical changes to the contract state, checks can be made; for example: to confirm the previous value is not the same as the new one to be added, zero address checks, integer over or underflows. These checks should happen at the beginning of the functions to ensure reverts happen before excessive gas is consumed.

Status

Resolved

B.5: Out of Gas Issue

```
// Refund Ethers to rest bidders
for (
    uint256 i = 0;
    i < allBidersAddress[_nftId⬆][auctionIndex].biddersAddr.length;
    i++
) {
    if (
        (allBidersAddress[_nftId⬆][auctionIndex].biddersAddr[i] !=
            AuctionDataset[_nftId⬆].highestBidder) &&
        (allBidersAddress[_nftId⬆][auctionIndex].biddersAddr[i] !=
            address(0))
    ) {
        // Transfer
        address userAddress = allBidersAddress[_nftId⬆][auctionIndex]
            .biddersAddr[i];
        IWETH(WETH).transfer(
            allBidersAddress[_nftId⬆][auctionIndex].biddersAddr[i],
            bids[_nftId⬆][userAddress]
        );
        bids[_nftId⬆][
            allBidersAddress[_nftId⬆][auctionIndex].biddersAddr[i]
        ] = 0;
    } else {
        if (
            (allBidersAddress[_nftId⬆][auctionIndex].biddersAddr[
                i
            ] != address(0))
        ) {
            uint256 NftPrice = AuctionDataset[_nftId⬆].highestBid;
```

The marketplace auctioning process is completed when the sellers of an nft who auctions the nft call the transferToHigherBidder function in order to send the nft to the highest bid and refund older bidders. Within this function, there is a for-loop that will run depending on the number of bidders for a particular nfts. Since the number of bidders could be any number, if the number is high, for instance, more than 250 bidders, there is a high tendency of the function failing if the seller does not have enough ethers to pay for cost.

Recommendation

Consider limiting the number of bidders per nft in order to control the number of times a for-loop will run. Another alternative recommendation is to design the architecture to immediately refund bidders' funds when a new bidder provides a bid greater than the former; this code implementation should be reflected in the bid function. By this way, it saves the seller an amount of gas to pay when they have to transfer the item to the highest bidder and get their auction cost.

Status

Acknowledged

NFJ Comment: Maximum number of bidders will be set to 25; this is set from the frontend to limit number of those to bid on an NFT.

Informational Issues

B.6: Lengthy Error Message

```
require(
    _payAmount >= totalAmountUserPay,
    "Not enough ether to cover item price and market fee, Price + Platfrom Fee"
);
```

Lin# - 352, 473

Description

Lengthy string error messages consume more gas when these functions are called. While it is imperative to have error messages that hint at a problem, it is recommended for them to be brief and clear.

Remediation

Error messages should be brief and concise. It should make the user understand reasons for the failure of a function call.

Status

Resolved

B.7: Inaccurate Code Behavior

```
    function getRoyaltyPlatformFee(uint256 _nftId)
        public
        view
        returns (
            uint256,
            address,
            uint256,
            address
        )
    {
        return IERC721(nftContractAddr).getRoyaltyAndPlatformFeeDetails(_nftId);
    }
}
```

Description

There is no specified pragma solidity version in this contract as well. It is also not properly interfacing with the ArtisticJeweller NFT contract. The ERC721 interface provides a limit to the number of functions it has access to. A function like the one below is not available in the standard ERC721 interface and calling it this way would fail.

Remediation

Consider updating the code to match up with recommendations made by external contracts and dependencies for optimal functionality. Also include a specific pragma solidity version in the contract

Status

Resolved

Functional Testing

Some of the tests performed are mentioned below

- When Contract is Paused prevents token transfer (89ms)
- Should revert when non-owner and NFTcreator attempts to burn token by calling burnNFT
- Should updateRoyalty minted by contract owner
- Successfully mint and burn token by calling mintNFT
- Should know the different roles of NFTowner and NFTcreator
- should put NFTs on sale after putting them on auction [FAILED]
- should remove NFTs from sale
- should transfer NFTs to highest bidder
- Should revert when no approval is given to the contract before auctioning an nft
- Should successfully auction and transfer nft into marketplace contract as nft owner
- Should bid successfully and become the first bidder
- Should bid successfully and become the new highest bidder
- Should revert bid when auction end time has reached.
- Should cancel bid successfully and remove bidder from bidderlist
- Should transfer nft from the contract to the highest bidder when the owner calls
- Should know the effect of seller calling the transferToHigherBidder without bidders
- Should revert when seller calls transferToHigherBidder on same tokenId already sent to a bidder
- Should allow the successful bidder who acquired nft to putOnSale the nft.
- Should allow removal of NFT when put on Sale [FAILED]
- Should revert when unknown attempts to cancel auction
- Should successfully cancel auction before any bid is made

Automated Tests

No major issues were found. Some false positive errors were reported by the tools. All the other issues have been categorized above according to their level of severity.

```
ERC1967UpgradeUpgradeable._functionDelegateCall(address,bytes) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#198-204) uses delegatecall to a input-controlled function id
        - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#202)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#controlled-delegatecall

AJMarketplace.transferToHigherBidder(uint256) (contracts/Marketplace.sol#182-280) ignores return value by IWETH(WETH).transfer(msg.sender,nftValue_85) (contracts/Marketplace.sol#214)
AJMarketplace.transferToHigherBidder(uint256) (contracts/Marketplace.sol#182-280) ignores return value by IWETH(WETH).transfer(allBidersAddress[_nftId][auctionIndex].biddersAddr[i],bids[_nftId][userAddress]) (contracts
/Marketplace.sol#231-234)
AJMarketplace.transferToHigherBidder(uint256) (contracts/Marketplace.sol#182-280) ignores return value by IWETH(WETH).transfer(platformAddr,etherValue_platform) (contracts/Marketplace.sol#253)
AJMarketplace.transferToHigherBidder(uint256) (contracts/Marketplace.sol#182-280) ignores return value by IWETH(WETH).transfer(royaltyAddr,etherValue_royalty) (contracts/Marketplace.sol#255)
AJMarketplace.bid(uint256,uint256) (contracts/Marketplace.sol#336-389) ignores return value by IWETH(WETH).transferFrom(msg.sender,address(this),_payAmount) (contracts/Marketplace.sol#374)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

ArtisticJeweller (contracts/ArtisticJeweller.sol#11-178) is an upgradeable contract that does not protect its initiliaze functions: ArtisticJeweller.initialize(address,uint128) (contracts/ArtisticJeweller.sol#39-51). A
nyone can delete the contract with: UUPSUpgradeable.upgradeTo(address) (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol#72-75)UUPSUpgradeable.upgradeToAndCall(address,bytes) (node_modul
es/@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol#85-88)Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unprotected-upgradeable-contract

MathUpgradeable.mulDiv(uint256,uint256,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse = (3 * denominator) ^ 2 (node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol#117)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol#121)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol#122)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol#123)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol#124)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol#125)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -denominator = denominator / twos (node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol#102)
        -inverse *= 2 - denominator * inverse (node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol#126)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) performs a multiplication on the result of a division:
        -prod0 = prod0 / twos (node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol#105)
        -result = prod0 * inverse (node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol#132)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

Reentrancy in AJMarketplace.bid(uint256,uint256) (contracts/Marketplace.sol#336-389):
        External calls:
        - IWETH(WETH).transferFrom(msg.sender,address(this),_payAmount) (contracts/Marketplace.sol#374)
        State variables written after the call(s):
        - AuctionDataset[_nftId].totalBidAmount += itemActualPrice (contracts/Marketplace.sol#376)
        - AuctionDataset[_nftId].highestBid = itemActualPrice (contracts/Marketplace.sol#377)
        - AuctionDataset[_nftId].highestBidder = msg.sender (contracts/Marketplace.sol#378)
        - AuctionDataset[_nftId].bidCounter += 1 (contracts/Marketplace.sol#380)
        - allBidersAddress[_nftId][auctionIndex].bidStatus.push(false) (contracts/Marketplace.sol#383)
        - allBidersAddress[_nftId][auctionIndex].bidderLenth += 1 (contracts/Marketplace.sol#384)
        - allBidersAddress[_nftId][auctionIndex].biddersAddr.push(msg.sender) (contracts/Marketplace.sol#387)
Reentrancy in AJMarketplace.bidCancelByUser(uint256) (contracts/Marketplace.sol#94-137):
        External calls:
        - require(bool,string)(IWETH(WETH).transfer(msg.sender,bids[_nftId][msg.sender]),Unable to transfer Refund) (contracts/Marketplace.sol#104-107)
        State variables written after the call(s):
        - bids[_nftId][msg.sender] = 0 (contracts/Marketplace.sol#108)
Reentrancy in AJMarketplace.cancelAuction(uint256) (contracts/Marketplace.sol#139-180):
        External calls:
        - IERC721(nftContractAddr).safeTransferFrom(address(this),msg.sender,_nftId) (contracts/Marketplace.sol#148-152)
        State variables written after the call(s):
        - AuctionDataset[_nftId] = AuctionData(0,address(0),false,false,0,0,address(0),0,0) (contracts/Marketplace.sol#169-179)
Reentrancy in AJMarketplace.putOnSale(uint256,uint256) (contracts/Marketplace.sol#435-455):
        External calls:
        - IERC721(nftContractAddr).transferFrom(msg.sender,address(this),_tokenId) (contracts/Marketplace.sol#441-445)
        State variables written after the call(s):
        - items[_tokenId] = Item(_tokenId,nftContractAddr,_price,address(msg.sender),true) (contracts/Marketplace.sol#446-452)
Reentrancy in AJMarketplace.removeFromSale(uint256) (contracts/Marketplace.sol#457-473):
        External calls:
        - IERC721(nftContractAddr).transferFrom(address(this),msg.sender,item.tokenId) (contracts/Marketplace.sol#467-471)
        State variables written after the call(s):
        - items[_tokenId] = Item(0,address(0),0,address(address(0)),false) (contracts/Marketplace.sol#472)
Reentrancy in AJMarketplace.transferToHigherBidder(uint256) (contracts/Marketplace.sol#182-280):
        External calls:
        - IERC721(nftContractAddr).safeTransferFrom(address(this),AuctionDataset[_nftId].highestBidder,_nftId) (contracts/Marketplace.sol#202-206)
        - IWETH(WETH).transfer(msg.sender,nftValue_85) (contracts/Marketplace.sol#214)
        - IWETH(WETH).transfer(allBidersAddress[_nftId][auctionIndex].biddersAddr[i],bids[_nftId][userAddress]) (contracts/Marketplace.sol#231-234)
        State variables written after the call(s):
        - bids[_nftId][allBidersAddress[_nftId][auctionIndex].biddersAddr[i]] = 0 (contracts/Marketplace.sol#235-237)
Reentrancy in AJMarketplace.transferToHigherBidder(uint256) (contracts/Marketplace.sol#182-280):
        External calls:
        - IERC721(nftContractAddr).safeTransferFrom(address(this),AuctionDataset[_nftId].highestBidder,_nftId) (contracts/Marketplace.sol#202-206)
        - IWETH(WETH).transfer(msg.sender,nftValue_85) (contracts/Marketplace.sol#214)
        - IERC721(nftContractAddr).safeTransferFrom(address(this),AuctionDataset[_nftId].seller,_nftId) (contracts/Marketplace.sol#262-266)
        State variables written after the call(s):
        - AuctionDataset[_nftId] = AuctionData(0,address(0),false,false,0,0,address(0),0,0) (contracts/Marketplace.sol#269-279)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

ERC1967UpgradeUpgradeable._upgradeToAndCallUUPS(address,bytes,bool).slot (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#98) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

ERC1967UpgradeUpgradeable._upgradeToAndCallUUPS(address,bytes,bool) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#87-105) ignores return value by IERC1822ProxiableUpgrade
able(newImplementation).proxiableUUID() (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#98-102)
ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#434-456) ignores return value by IERC721ReceiverUpgradeable(t
o).onERC721Received(_msgSender(),from,tokenId,data) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#441-452)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

ArtisticJeweller.updatePlatformDetails(address,uint128) (contracts/ArtisticJeweller.sol#136-142) should emit an event for:
        - PlatformFee = _platformCuts (contracts/ArtisticJeweller.sol#140)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

ArtisticJeweller.initialize(address,uint128)._platformReciverAddr (contracts/ArtisticJeweller.sol#39) lacks a zero-check on :
                - PlatformAddress = address(_platformReciverAddr) (contracts/ArtisticJeweller.sol#50)
ArtisticJeweller.updatePlatformDetails(address,uint128)._platformReciverAddr (contracts/ArtisticJeweller.sol#137) lacks a zero-check on :
                - PlatformAddress = address(_platformReciverAddr) (contracts/ArtisticJeweller.sol#141)
AJMarketplace.constructor(address,address)._nftContractAddr (contracts/Marketplace.sol#67) lacks a zero-check on :
                - nftContractAddr = _nftContractAddr (contracts/Marketplace.sol#68)
```

```
AJMarketplace.constructor(address,address)._weth (contracts/Marketplace.sol#67) lacks a zero-check on :
                - WETH = _weth (contracts/Marketplace.sol#69)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

AJMarketplace.getRoyaltyPlatformFee(uint256) (contracts/Marketplace.sol#519-530) has external calls inside a loop: IAJ(nftContractAddr).getRoyaltyAndPlatformFeeDetails(_nftId) (contracts/Marketplace.sol#529)
AJMarketplace.cancelAuction(uint256) (contracts/Marketplace.sol#139-180) has external calls inside a loop: require(bool,string)(IWETH(WETH).transfer(user,bids[_nftId][user]),Unable to transfer Refund) (contracts/Market
place.sol#162-165)
AJMarketplace.transferToHigherBidder(uint256) (contracts/Marketplace.sol#182-280) has external calls inside a loop: IWETH(WETH).transfer(allBidersAddress[_nftId][auctionIndex].biddersAddr[i],bids[_nftId][userAddress])
(contracts/Marketplace.sol#231-234)
AJMarketplace.transferToHigherBidder(uint256) (contracts/Marketplace.sol#182-280) has external calls inside a loop: IWETH(WETH).transfer(platformAddr,etherValue_platform) (contracts/Marketplace.sol#253)
AJMarketplace.transferToHigherBidder(uint256) (contracts/Marketplace.sol#182-280) has external calls inside a loop: IWETH(WETH).transfer(royaltyAddr,etherValue_royalty) (contracts/Marketplace.sol#255)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop

Variable 'ERC1967UpgradeUpgradeable._upgradeToAndCallUUPS(address,bytes,bool).slot (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#98)' in ERC1967UpgradeUpgradeable._upgrad
eToAndCallUUPS(address,bytes,bool) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#87-105) potentially used before declaration: require(bool,string)(slot == _IMPLEMENTATION
_SLOT,ERC1967Upgrade: unsupported proxiableUUID) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#99)
Variable 'ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes).retval (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#441)' in ERC721Upgradeable._checkOnERC721Rec
eived(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#434-456) potentially used before declaration: retval == IERC721ReceiverUpgradeable.onERC721Recei
ved.selector (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#442)
Variable 'ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes).reason (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#443)' in ERC721Upgradeable._checkOnERC721Rec
eived(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#434-456) potentially used before declaration: reason.length == 0 (node_modules/@openzeppelin/con
tracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#444)
Variable 'ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes).reason (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#443)' in ERC721Upgradeable._checkOnERC721Rec
eived(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#434-456) potentially used before declaration: revert(uint256,uint256)(32 + reason,mload(uint256)
(reason)) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#449)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

Reentrancy in AJMarketplace.bid(uint256,uint256) (contracts/Marketplace.sol#336-389):
        External calls:
        - IWETH(WETH).transferFrom(msg.sender,address(this),_payAmount) (contracts/Marketplace.sol#374)
        State variables written after the call(s):
        - bids[_nftId][msg.sender] += _payAmount (contracts/Marketplace.sol#379)
Reentrancy in AJMarketplace.bidCancelByUser(uint256) (contracts/Marketplace.sol#94-137):
        External calls:
        - require(bool,string)(IWETH(WETH).transfer(msg.sender,bids[_nftId][msg.sender]),Unable to transfer Refund) (contracts/Marketplace.sol#104-107)
        State variables written after the call(s):
        - AuctionDataset[_nftId].highestBidder = higerAddr (contracts/Marketplace.sol#123)
        - AuctionDataset[_nftId].highestBid = itemActualPrice (contracts/Marketplace.sol#132)
        - AuctionDataset[_nftId].bidCounter -= 1 (contracts/Marketplace.sol#133)
Reentrancy in AJMarketplace.putOnAuction(uint256,uint256,uint256) (contracts/Marketplace.sol#282-311):
        External calls:
        - IERC721(nftContractAddr).transferFrom(msg.sender,address(this),_nftId) (contracts/Marketplace.sol#305-309)
        State variables written after the call(s):
        - nftAuctionCount[_nftId] += 1 (contracts/Marketplace.sol#310)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in AJMarketplace.purchaseNFT(uint256) (contracts/Marketplace.sol#475-517):
        External calls:
        - IERC721(nftContractAddr).transferFrom(address(this),msg.sender,item.tokenId) (contracts/Marketplace.sol#505-509)
        External calls sending eth:
        - address(platformAddr).transfer(etherValue_platform) (contracts/Marketplace.sol#501)
        - address(royaltyAddr).transfer(etherValue_royalty) (contracts/Marketplace.sol#502)
        - address(item.seller).transfer(etherValue_Selling) (contracts/Marketplace.sol#503)
        Event emitted after the call(s):
        - Bought(item.tokenId,item.nft,item.price,item.seller,msg.sender) (contracts/Marketplace.sol#510-516)
Reentrancy in AJMarketplace.putOnSale(uint256,uint256) (contracts/Marketplace.sol#435-455):
        External calls:
        - IERC721(nftContractAddr).transferFrom(msg.sender,address(this),_tokenId) (contracts/Marketplace.sol#441-445)
        Event emitted after the call(s):
        - Offered(_tokenId,nftContractAddr,_price,msg.sender) (contracts/Marketplace.sol#454)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

AJMarketplace.transferToHigherBidder(uint256) (contracts/Marketplace.sol#182-280) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= AuctionDataset[_nftId].endAt,Auction is still ongoing!) (contracts/Marketplace.sol#185-188)
AJMarketplace.bid(uint256,uint256) (contracts/Marketplace.sol#336-389) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp < AuctionDataset[_nftId].endAt,Ended!) (contracts/Marketplace.sol#352)
AJMarketplace.AuctionRemainingTime(uint256) (contracts/Marketplace.sol#400-411) uses timestamp for comparisons
        Dangerous comparisons:
        - AuctionDataset[_nftId].endAt > currentTime (contracts/Marketplace.sol#406)
AJMarketplace.removeFromSale(uint256) (contracts/Marketplace.sol#457-473) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp < AuctionDataset[_tokenId].endAt,Seller cannot remove nft from auction after end time) (contracts/Marketplace.sol#463-466)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

ERC721Upgradeable._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#434-456) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#448-450)
AddressUpgradeable._revert(bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#206-218) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#211-214)
StorageSlotUpgradeable.getAddressSlot(bytes32) (node_modules/@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#52-57) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#54-56)
StorageSlotUpgradeable.getBooleanSlot(bytes32) (node_modules/@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#62-67) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#64-66)
StorageSlotUpgradeable.getBytes32Slot(bytes32) (node_modules/@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#72-77) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#74-76)
StorageSlotUpgradeable.getUint256Slot(bytes32) (node_modules/@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#82-87) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#84-86)
StringsUpgradeable.toString(uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/StringsUpgradeable.sol#18-38) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/StringsUpgradeable.sol#24-26)
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/StringsUpgradeable.sol#30-32)
MathUpgradeable.mulDiv(uint256,uint256,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol#55-135) uses assembly
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol#66-70)
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol#86-93)
        - INLINE ASM (node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol#100-109)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

AJMarketplace.bidCancelByUser(uint256) (contracts/Marketplace.sol#94-137) compares to a boolean constant:
        -allBidersAddress[_nftId][auctionIndex].bidStatus[i - 1] != true (contracts/Marketplace.sol#119)
AJMarketplace.cancelAuction(uint256) (contracts/Marketplace.sol#139-180) compares to a boolean constant:
        -allBidersAddress[_nftId][auctionIndex].bidStatus[i] != true (contracts/Marketplace.sol#159)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Different versions of Solidity are used:
        - Version used: ['0.8.17', '^0.8', '^0.8.0', '^0.8.1', '^0.8.2']
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/interfaces/draft-IERC1822Upgradeable.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)
        - ^0.8.2 (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/PausableUpgradeable.sol#4)
```

```
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.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/IERC721Upgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/extensions/ERC721BurnableUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/extensions/IERC721MetadataUpgradeable.sol#4)
        - ^0.8.1 (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/CountersUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/StringsUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/IERC165Upgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/security/ReentrancyGuard.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/IERC721.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#4)
        - 0.8.17 (contracts/ArtisticJeweller.sol#2)
        - 0.8.17 (contracts/Marketplace.sol#2)
        - ^0.8 (contracts/WETH.sol#16)
        - 0.8.17 (contracts/interfaces/IArtisticJeweller.sol#2)
        - 0.8.17 (contracts/interfaces/IWETH.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

ArtisticJeweller._beforeTokenTransfer(address,address,uint256) (contracts/ArtisticJeweller.sol#67-73) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/interfaces/draft-IERC1822Upgradeable.sol#4) allows old versions
Pragma version^0.8.2 (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/beacon/IBeaconUpgradeable.sol#4) allows old versions
Pragma version^0.8.2 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/IERC721ReceiverUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/IERC721Upgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/extensions/ERC721BurnableUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/extensions/IERC721MetadataUpgradeable.sol#4) allows old versions
Pragma version^0.8.1 (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/CountersUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/StorageSlotUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/StringsUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/IERC165Upgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/security/ReentrancyGuard.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/IERC721.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#4) allows old versions
Pragma version0.8.17 (contracts/ArtisticJeweller.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.17 (contracts/Marketplace.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8 (contracts/WETH.sol#16) is too complex
Pragma version0.8.17 (contracts/interfaces/IArtisticJeweller.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version0.8.17 (contracts/interfaces/IWETH.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

```
Low level call in ERC1967UpgradeUpgradeable._functionDelegateCall(address,bytes) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#198-204):
        - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#202)
Low level call in AddressUpgradeable.sendValue(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#60-65):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#63)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#128-137):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#135)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#155-162):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#160)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

ArtisticJeweller (contracts/ArtisticJeweller.sol#11-178) should inherit from IAJ (contracts/interfaces/IArtisticJeweller.sol#4-12)
WETH9 (contracts/WETH.sol#18-77) should inherit from IWETH (contracts/interfaces/IWETH.sol#4-11)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-inheritance

Function OwnableUpgradeable.__Ownable_init() (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#29-31) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#33-35) is not in mixedCase
Variable OwnableUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#94) is not in mixedCase
Function ERC1967UpgradeUpgradeable.__ERC1967Upgrade_init() (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#21-22) is not in mixedCase
Function ERC1967UpgradeUpgradeable.__ERC1967Upgrade_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#24-25) is not in mixedCase
Variable ERC1967UpgradeUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#211) is not in mixedCase
Function UUPSUpgradeable.__UUPSUpgradeable_init() (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol#23-24) is not in mixedCase
Function UUPSUpgradeable.__UUPSUpgradeable_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol#26-27) is not in mixedCase
Variable UUPSUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol#107) is not in mixedCase
Variable UUPSUpgradeable.__self (node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol#29) is not in mixedCase
Function PausableUpgradeable.__Pausable_init() (node_modules/@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#34-36) is not in mixedCase
Function PausableUpgradeable.__Pausable_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#38-40) is not in mixedCase
Variable PausableUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#116) is not in mixedCase
Function ERC721Upgradeable.__ERC721_init(string,string) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#45-47) is not in mixedCase
Function ERC721Upgradeable.__ERC721_init_unchained(string,string) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#49-52) is not in mixedCase
Function ERC721Upgradeable.__unsafe_increaseBalance(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#508-510) is not in mixedCase
Variable ERC721Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#517) is not in mixedCase
Function ERC721BurnableUpgradeable.__ERC721Burnable_init() (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/extensions/ERC721BurnableUpgradeable.sol#15-16) is not in mixedCase
Function ERC721BurnableUpgradeable.__ERC721Burnable_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/extensions/ERC721BurnableUpgradeable.sol#18-19) is not in mixedCase
Variable ERC721BurnableUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/extensions/ERC721BurnableUpgradeable.sol#38) is not in mixedCase
Function ContextUpgradeable.__Context_init() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#18-19) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#21-22) is not in mixedCase
Variable ContextUpgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#36) is not in mixedCase
Function ERC165Upgradeable.__ERC165_init() (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#24-25) is not in mixedCase
Function ERC165Upgradeable.__ERC165_init_unchained() (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#27-28) is not in mixedCase
Variable ERC165Upgradeable.__gap (node_modules/@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol#41) is not in mixedCase
Parameter ArtisticJeweller.initialize(address,uint128)._platformReciverAddr (contracts/ArtisticJeweller.sol#39) is not in mixedCase
Parameter ArtisticJeweller.initialize(address,uint128)._platformCuts (contracts/ArtisticJeweller.sol#39) is not in mixedCase
Parameter ArtisticJeweller.mintNFT(address,string,uint128,address)._json (contracts/ArtisticJeweller.sol#100) is not in mixedCase
Parameter ArtisticJeweller.mintNFT(address,string,uint128,address)._nftRoyalty (contracts/ArtisticJeweller.sol#101) is not in mixedCase
Parameter ArtisticJeweller.mintNFT(address,string,uint128,address)._royaltyAddress (contracts/ArtisticJeweller.sol#102) is not in mixedCase
Parameter ArtisticJeweller.updateNFTJsonData(uint256,string)._jsonData (contracts/ArtisticJeweller.sol#128) is not in mixedCase
Parameter ArtisticJeweller.updatePlatformDetails(address,uint128)._platformReciverAddr (contracts/ArtisticJeweller.sol#137) is not in mixedCase
Parameter ArtisticJeweller.updatePlatformDetails(address,uint128)._platformCuts (contracts/ArtisticJeweller.sol#138) is not in mixedCase
Parameter ArtisticJeweller.updateNFTRoyalty(uint256,uint128,address)._royaltyFee (contracts/ArtisticJeweller.sol#147) is not in mixedCase
Parameter ArtisticJeweller.updateNFTRoyalty(uint256,uint128,address)._royaltyAddress (contracts/ArtisticJeweller.sol#148) is not in mixedCase
Parameter ArtisticJeweller.getRoyaltyAndPlatformFeeDetails(uint256)._nftId (contracts/ArtisticJeweller.sol#160) is not in mixedCase
Variable ArtisticJeweller.PlatformAddress (contracts/ArtisticJeweller.sol#23) is not in mixedCase
Variable ArtisticJeweller.PlatformFee (contracts/ArtisticJeweller.sol#24) is not in mixedCase
Modifier ArtisticJeweller.NFTOwner(uint256) (contracts/ArtisticJeweller.sol#81-87) is not in mixedCase
Modifier ArtisticJeweller.NFTCreator(uint256) (contracts/ArtisticJeweller.sol#89-95) is not in mixedCase
Struct AJMarketplace.allBiddersRec (contracts/Marketplace.sol#15-19) is not in CapWords
Parameter AJMarketplace.reverseLogo(uint256)._nftId (contracts/Marketplace.sol#75) is not in mixedCase
```

```
symbol() should be declared external:
        - ERC721Upgradeable.symbol() (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#91-93)
tokenURI(uint256) should be declared external:
        - ERC721Upgradeable.tokenURI(uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#98-103)
approve(address,uint256) should be declared external:
        - ERC721Upgradeable.approve(address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#117-127)
setApprovalForAll(address,bool) should be declared external:
        - ERC721Upgradeable.setApprovalForAll(address,bool) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#141-143)
transferFrom(address,address,uint256) should be declared external:
        - ERC721Upgradeable.transferFrom(address,address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#155-164)
safeTransferFrom(address,address,uint256) should be declared external:
        - ERC721Upgradeable.safeTransferFrom(address,address,uint256) (node_modules/@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol#169-175)
initialize(address,uint128) should be declared external:
        - ArtisticJeweller.initialize(address,uint128) (contracts/ArtisticJeweller.sol#39-51)
pause() should be declared external:
        - ArtisticJeweller.pause() (contracts/ArtisticJeweller.sol#53-55)
unpause() should be declared external:
        - ArtisticJeweller.unpause() (contracts/ArtisticJeweller.sol#57-59)
safeMint(address) should be declared external:
        - ArtisticJeweller.safeMint(address) (contracts/ArtisticJeweller.sol#61-65)
mintNFT(address,string,uint128,address) should be declared external:
        - ArtisticJeweller.mintNFT(address,string,uint128,address) (contracts/ArtisticJeweller.sol#98-116)
burnNFT(uint256) should be declared external:
        - ArtisticJeweller.burnNFT(uint256) (contracts/ArtisticJeweller.sol#119-125)
updateNFTJsonData(uint256,string) should be declared external:
        - ArtisticJeweller.updateNFTJsonData(uint256,string) (contracts/ArtisticJeweller.sol#128-133)
updatePlatformDetails(address,uint128) should be declared external:
        - ArtisticJeweller.updatePlatformDetails(address,uint128) (contracts/ArtisticJeweller.sol#136-142)
updateNFTRoyalty(uint256,uint128,address) should be declared external:
        - ArtisticJeweller.updateNFTRoyalty(uint256,uint128,address) (contracts/ArtisticJeweller.sol#145-152)
getTotalNFTCount() should be declared external:
        - ArtisticJeweller.getTotalNFTCount() (contracts/ArtisticJeweller.sol#155-157)
getRoyaltyAndPlatformFeeDetails(uint256) should be declared external:
        - ArtisticJeweller.getRoyaltyAndPlatformFeeDetails(uint256) (contracts/ArtisticJeweller.sol#160-176)
getBidderAddressAndStatus(uint256,uint8) should be declared external:
        - AJMarketplace.getBidderAddressAndStatus(uint256,uint8) (contracts/Marketplace.sol#325-334)
getAllBidders(uint256) should be declared external:
        - AJMarketplace.getAllBidders(uint256) (contracts/Marketplace.sol#391-398)
AuctionRemainingTime(uint256) should be declared external:
        - AJMarketplace.AuctionRemainingTime(uint256) (contracts/Marketplace.sol#400-411)
deposit() should be declared external:
        - WETH9.deposit() (contracts/WETH.sol#34-37)
withdraw(uint256) should be declared external:
        - WETH9.withdraw(uint256) (contracts/WETH.sol#38-43)
totalSupply() should be declared external:
        - WETH9.totalSupply() (contracts/WETH.sol#45-47)
approve(address,uint256) should be declared external:
        - WETH9.approve(address,uint256) (contracts/WETH.sol#49-53)
transfer(address,uint256) should be declared external:
        - WETH9.transfer(address,uint256) (contracts/WETH.sol#55-57)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

Closing Summary

In this report, we have considered the security of NFJ Labs. We performed our audit according to the procedure described above.

Some Issues of high, medium, low, and informational severity were found during the course of the audit.

Disclaimer

QuillAudits smart contract audit is not a security warranty, investment advice, or an endorsement of the NFJ Labs 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 NFJ Labs Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

About QuillAudits

QuillAudits is a secure smart contracts audit platform designed by QuillHash Technologies. We are a team of dedicated blockchain security experts and smart contract auditors determined to ensure that Smart Contract-based Web3 projects can avail the latest and best security solutions to operate in a trustworthy and risk-free ecosystem.

700+
Audits Completed

$16B
Secured

700K
Lines of Code Audited

Follow Our Journey

Audit Report
April, 2023

For

NFJ LABS.

QuillAudits

Canada, India, Singapore, United Kingdom

www.quillaudits.com

audits@quillhash.com