// HALBORN

Yieldly.Finance - Polygon NFT Marketplace

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **December 7th, 2021 - December 20th, 2021**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	12/07/2021	Gabi Urrutia
0.2	Document Edits	12/10/2021	Gokberk Gulgun
0.3	Final Draft	12/20/2021	Gokberk Gulgun
0.4	Draft Review	12/20/2021	Gabi Urrutia
1.0	Remediation Plan	12/31/2021	Gokberk Gulgun
1.1	Remediation Plan Review	01/03/2022	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Gokberk Gulgun	Halborn	Gokberk.Gulgun@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Yieldly.Finance engaged Halborn to conduct a security assessment on their Compound Smart contract beginning on December 7, 2021 and ending December 20th, 2021. The security assessment was scoped to the NFT Polygon contracts and an audit of the security risk and implications regarding the changes introduced by the development team at Yieldly.Finance before its production release shortly following the assessment's deadline.

1.2 AUDIT SUMMARY

The security engineers involved on the audit are blockchain and smart contract security experts with advanced penetration testing, smart contract hacking, and in-depth knowledge of multiple blockchain protocols.

The purpose of this audit is to achieve the following:

- Ensure that smart contract functions operate as intended
- Identify potential security issues with the smart contracts

In summary, Halborn identified some security risks that were addressed and accepted by Yieldly.Finance team. However, External threats, such as financial related attacks, oracle attacks, and inter-contract functions and calls should be validated for expected logic and state.

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of this audit. While manual testing is recommended to uncover flaws in logic, process,and implementation; automated testing techniques help enhance coverage of the bridge code and can quickly identify items

that do not follow security best practices. The following phases and associated tools were used throughout the term of the audit:

- Research into architecture and purpose
- Smart contract manual code review and walkthrough
- Graphing out functionality and contract logic/connectivity/functions (solgraph)
- Manual assessment of use and safety for the critical Solidity variables and functions in scope to identify any arithmetic related vulnerability classes
- Manual testing by custom scripts
- Scanning of solidity files for vulnerabilities, security hotspots or bugs. (MythX)
- Static Analysis of security for scoped contract, and imported functions. (Slither)
- Testnet deployment (Brownie, Remix IDE)

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

5 - Almost certain an incident will occur.
4 - High probability of an incident occurring.
3 - Potential of a security incident in the long term.
2 - Low probability of an incident occurring.
1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.
3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.
1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL

10 - CRITICAL
9 - 8 - HIGH
7 - 6 - MEDIUM
5 - 4 - LOW
3 - 1 - VERY LOW AND INFORMATIONAL

1.4 SCOPE

IN-SCOPE:
The security assessment was scoped to the following smart contracts:

- Marketplace.sol
- Yieldly.sol

Commit ID: 5a6ba255a1852f8166e6f201e7c9cf8d9f1a3b78

Out-of-scope: External libraries and financial related attacks

Fixes Commit ID: 36883324f918b97f61d26f329b028083fc44e13a

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	1	0	7	17

EXECUTIVE OVERVIEW

LIKELIHOOD

IMPACT

			(HAL-01)	
(HAL-03) (HAL-06) (HAL-07) (HAL-08)				
	(HAL-02) (HAL-04) (HAL-05)			
(HAL-09) (HAL-10) (HAL-11) (HAL-12) (HAL-13) (HAL-14) (HAL-15) (HAL-16) (HAL-17) (HAL-18) (HAL-19) (HAL-20) (HAL-21) (HAL-22) (HAL-23) (HAL-24) (HAL-25)				

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL01 - UNRESTRICTED TOKEN MINTING	High	SOLVED - 12/24/2021
HAL02 - BID TIME CONFUSION ON THE AUCTION	Low	SOLVED - 12/24/2021
HAL03 - CENTRALIZATION RISK	Low	SOLVED - 12/24/2021
HAL04 - LACK OF START DATE CHECK ON THE AUCTION CREATION	Low	SOLVED - 12/24/2021
HAL05 - MISSING ZERO ADDRESS VALIDATION	Low	SOLVED - 12/24/2021
HAL06 - LACK OF PAUSE MODIFIER ON THE MINT FUNCTIONALITY	Low	SOLVED - 12/24/2021
HAL07 - TOKEN IS BUYABLE WHEN THE AUCTION IS ACTIVE	Low	RISK ACCEPTED
HAL08 - OWNER CAN RENOUNCE OWNERSHIP	Low	RISK ACCEPTED
HAL09 - MISFUNCTIONAL TOKEN OWNER SET ON THE AUCTION CREATION	Informational	ACKNOWLEDGED
HAL10 - UPGRADE AT LEAST PRAGMA 0.8.4	Informational	ACKNOWLEDGED
HAL11 - ++i IS MORE GAS EFFICIENT THAN i++ IN THE LOOPS	Informational	ACKNOWLEDGED
HAL12 - FLOATING PRAGMA	Informational	ACKNOWLEDGED
HAL13 - POSSIBLE MISUSE OF PUBLIC FUNCTIONS	Informational	ACKNOWLEDGED
HAL14 - REDUNDANT IMPORT	Informational	ACKNOWLEDGED
HAL15 - LESS THAN 256 UINTS ARE NOT GAS EFFICIENT	Informational	ACKNOWLEDGED
HAL16 - MISSING EVENT EMITTING	Informational	ACKNOWLEDGED
HAL17 - REVERT STRING SIZE OPTIMIZATION	Informational	ACKNOWLEDGED

HAL18 - APPROVE IS NOT REVOKED DURING THE CLOSE TRADE	Informational	ACKNOWLEDGED
HAL19 - MISSING EXTERNAL OWNED ACCOUNT CHECK	Informational	ACKNOWLEDGED
HAL20 - TYPO ON THE REVERT MESSAGE	Informational	ACKNOWLEDGED
HAL21 - SAFEMATH IS NOT NEEDED WHEN USING SOLIDITY VERSION 0.8	Informational	ACKNOWLEDGED
HAL22 - CACHING THE LENGTH IN THE FOR LOOPS	Informational	ACKNOWLEDGED
HAL23 - MISSING ZERO/THRESHOLD CHECK FOR NFT SALE PRICE	Informational	ACKNOWLEDGED
HAL24 - MINT/BUY FUNCTION CAN BE FRONT-RUN	Informational	ACKNOWLEDGED
HAL25 - MINTING PROCESS WILL FAIL IF THE DESTINATION ADDRESS IS CONTRACT	Informational	ACKNOWLEDGED

FINDINGS & TECH DETAILS

3.1 (HAL-01) UNRESTRICTED TOKEN MINTING - HIGH

Description:

In the Yieldly.sol, The owner can call mintAll function for the token minting. However, the mint function does not have any access control modifier. An attacker can mint any tokenId via the public function.

Code Location:

Listing 1: Yieldly.sol (Lines 14)

```
14      function mint(
15          uint256 _id,
16          address _to,
17          string memory  _tokenURI
18      ) public
19      {
20          _safeMint(_to, _id);
21          _setTokenURI(_id, _tokenURI);
22      }
```

Risk Level:

Likelihood - 4
Impact - 4

Recommendation:

Consider to make function private. The function should be called from the users and the contract should eliminate centralization risks.

Remediation plan:

SOLVED: The issue was fixed in the following commit : Fix Commit.

3.2 (HAL-02) BID TIME CONFUSION ON THE AUCTION - LOW

Description:

In the Marketplace.sol, When creating bid for the auction, the contract does not set the bid date. The code section is commented out the relative set statement. On the other hand, firstBidTime is set when the auction created. Therefore, It doesn't show the correct timestamp.

Code Location:

Listing 2: Marketplace.sol (Lines 404)

```
402         if (auctions[tokenId].amount == 0) {
403             // If so, it is the first bid.
404             // auctions[tokenId].firstBidTime = block.timestamp;
405             // We only need to check if the bid matches reserve
                    bid for the first bid,
406             // since future checks will need to be higher than any
                     previous bid.
407             require(
408                 amount >= auctions[tokenId].reservePrice,
409                 "Must bid reservePrice or more"
410             );
411         }
```

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

Ensure that firstBidTime is set on the bid creation if it is not an intended behaviour.

Remediation Plan:

SOLVED: The issue was fixed in the following commit : Fix Commit.

3.3 (HAL-03) CENTRALIZATION RISK - LOW

Description:

In the Marketplace smart contract, the admin has the authority over the NFT token transfer. Any compromise to the privileged account which has access to onlyAdminRecovery may allow the attacker to take advantage.

Code Location:

Listing 3: Marketplace.sol (Lines 556)

```
556     function recoverNFT(uint256 tokenId) external
            onlyAdminRecovery {
557         Yieldly(nftContract).transferFrom(
558             // From the auction contract.
559             address(this),
560             // To the recovery account.
561             adminRecoveryAddress,
562             // For the specified token.
563             tokenId
564         );
565     }
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

Consider to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In the general, we are recommending centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract based accounts with the enhanced

security practices, e.g, Multisignature wallets.

Remediation Plan:

SOLVED: The issue was fixed in the following commit : Fix Commit.

3.4 (HAL-04) LACK OF START DATE CHECK ON THE AUCTION CREATION - LOW

Description:

During the dynamic testing, It has been seen that start_Date variable is not checked. When the new auction is created, Start-Date should be equal or more than the Latest Timestamp.

Code Location:

Listing 4: Start Date Has Not Been Checked

```
1     function createAuction(
2         uint256 tokenId,
3         uint256 startDate,
4         uint256 duration,
5         uint256 reservePrice
6     ) public nonReentrant whenNotPaused auctionNonExistant(tokenId
          ) {}
```

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

Consider to check start date is more than the latest timestamp.

Listing 5

```
1 require(startDate >= block.timestamp, "Can't create past auction
      !");
```

Remediation Plan:

SOLVED: The issue was fixed in the following commit : Fix Commit.

3.5 (HAL-05) MISSING ZERO ADDRESS CHECKS - LOW

Description:

In the contracts, nftContract, WMATICAddress and adminRecoveryAddress are missing address validation in their constructors. Every address should be validated and checked that is different from zero.

Code Location:

Listing 6

```
    constructor(
        address nftContract_,
        address WMATICAddress_,
        address adminRecoveryAddress_
    ) {
        require(
            IERC165(nftContract_).supportsInterface(
                ERC721_INTERFACE_ID),
            "Contract at nftContract_ address does not support NFT
                interface"
        );
        // Initialize immutable memory.
        nftContract = nftContract_;
        WMATICAddress = WMATICAddress_;
        adminRecoveryAddress = adminRecoveryAddress_;
        // Initialize mutable memory.
        _paused = false;
        _adminRecoveryEnabled = true;
    }
```

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

It is recommended to validate that every address input is different from zero.

Remediation Plan:

SOLVED: The issue was fixed in the following commit : Fix Commit.

3.6 (HAL-06) LACK OF PAUSE MODIFIER ON THE MINT FUNCTIONALITY - LOW

Description:

In the Marketplace.sol contract, during the contract is paused, the mint functionality is not paused.

Code Location:

Listing 7

```
1     function mint(string memory _tokenURI, uint256 _price, bool
          _isListOnMarketplace, uint256 _royalty, uint256 startDate,
          uint256 auctionLength) public
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

Consider review pause/unpause functionalities and activate them on the related functions.

Remediation Plan:

SOLVED: The issue was fixed in the following commit : Fix Commit.

3.7 (HAL-07) TOKEN IS BUYABLE WHEN THE AUCTION IS ACTIVE - LOW

Description:

In the Marketplace.sol contract, when the auction is active for the token, the token is still buyable. It depends on the behavior, the auction should be cancelled after the token bought by another owner.

Code Location:

Listing 8

```
1     function buy(uint _id, uint256 _price) external payable {
2         _validate(_id);
3         require(price[_id]==_price, "Error, price is not match");
4         address _previousOwner = ownerMap[_id];
5         address _newOwner = msg.sender;
6
7         // 2.5% commission cut
8         uint256 _commissionValue = price[_id].mul(25).div(1000);
9         uint256 _royaltyValue = price[_id].mul(royaltyMap[_id]).
              div(100);
10        uint256 _sellerValue = price[_id].sub(_commissionValue +
              _royaltyValue);
11        // _owner.transfer(_owner, _sellerValue);
12        transferMATICOrWMATIC(payable(_previousOwner),
              _sellerValue);
13        transferMATICOrWMATIC(payable(creatorMap[_id]),
              _royaltyValue);
14        transferMATICOrWMATIC(payable(adminRecoveryAddress),
              _commissionValue);
15        Yieldly(nftContract).transferFrom(address(this), _newOwner
              , _id);
16        ownerMap[_id] = msg.sender;
17        listedMap[_id] = false;
18        emit Purchase(_previousOwner, _newOwner, price[_id], _id);
19    }
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

Ensure that the behavior is intended, If It is not, cancel the current auction for the token id.

Remediation Plan:

RISK ACCEPTED: The Yieldly.Finance team accepted the risk for this finding.

3.8 (HAL-08) OWNER CAN RENOUNCE OWNERSHIP - LOW

Description:

The Owner of the contract is usually the account which deploys the contract. As a result, the Owner is able to perform privileged actions such as withdraw, mintBatch, addCreatorMap. Two of the contracts are inherited from Ownable contract. In the Marketplace.sol and Yieldly.sol, smart contract, the renounceOwnership function is used to renounce being an owner. If an owner is mistakenly renounced, administrative access would result in the contract having no Owner, eliminating the ability to call privileged functions. In such a case, contracts would have to be redeployed.

Risk Level:

Likelihood - 1
Impact - 3

Recommendations:

It is recommended that the Owner is not able to call renounceOwnership without transferring the Ownership to other address before. In addition, if a multi-signature wallet is used, calling renounceOwnership function should be confirmed for two or more users. As another solution, Renounce Ownership functionality can be disabled with the following line of codes.

Listing 9: Disable Renounce Ownership (Lines 2)

```
2 function renounceOwnership () public override onlyOwner {
3     revert ("can 't renounceOwnership here "); // not possible
          with this smart contract
4 }
```

Remediation Plan:

RISK ACCEPTED: The Yieldly.Finance team accepted the risk for this finding.

3.9 (HAL-09) MISFUNCTIONAL TOKEN OWNER SET ON THE AUCTION CREATION - INFORMATIONAL

Description:

In the Marketplace.sol, The owner of tokenId can create an auction. However, on the Line #359, the owner of the token is set to msg.sender. The check can cause serious problems in the contract, but transferFrom function is protecting the contract from a risk.

Code Location:

Listing 10: Marketplace.sol (Lines 359)

```
341     function createAuction(
342         uint256 tokenId,
343         uint256 startDate,
344         uint256 duration,
345         uint256 reservePrice
346     ) public nonReentrant whenNotPaused auctionNonExistant(tokenId
            ) {
347         // Check basic input requirements are reasonable.
348         require(msg.sender != address(0));
349         // Initialize the auction details, including null values.
350 
351         // if (_isNew) {
352         //     _tokenIds.increment();
353         //     uint256 newTokenId = _tokenIds.current();
354         //     tokenId = newTokenId;
355         //     price[tokenId] = reservePrice;
356         //     creatorMap[tokenId] = Creator;
357         //     Yieldly(nftContract).mint(tokenId, msg.sender,
                _tokenUri);
358         // }
359         ownerMap[tokenId] = msg.sender;
360         openTrade(tokenId, reservePrice);
361     }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider deleting the following code statement.

Listing 11

```
1         ownerMap[tokenId] = msg.sender;
```

Remediation Plan:

ACKNOWLEDGED: The Yieldly.Finance team acknowledged this finding.

3.10 (HAL-10) USING ++I CONSUMES LESS GAS THAN I++ IN LOOPS - INFORMATIONAL

Description:

In the loop below, the variable `i` is incremented using `i++`. It is known that, in loops, using `++i` costs less gas per iteration than `i++`.

Code Location:

Listing 12: Marketplace.sol (Lines 258)

```
258         for (uint i = 0; i < _newtokenIds.length; i ++) {
259             _tokenIds.increment();
260             creatorMap[_newtokenIds[i]] = _creators[i];
261             price[_newtokenIds[i]] = _prices[i];
262             ownerMap[_newtokenIds[i]] = _owners[i];
263             royaltyMap[_newtokenIds[i]] = _royalties[i];
264             listedMap[_newtokenIds[i]] = _listedMap[i];
265         }
```

Proof of Concept:

For example, based in the following test contract:

Listing 13: Test.sol

```
1 //SPDX-License-Identifier: MIT
2 pragma solidity 0.8.9;
3
4 contract test {
5     function postiincrement(uint256 iterations) public {
6         for (uint256 i = 0; i < iterations; i++) {
7         }
8     }
9     function preiincrement(uint256 iterations) public {
```

```
10          for (uint256 i = 0; i < iterations; ++i) {
11          }
12      }
13  }
```

```
>>> test_contract.postiincrement(1)
Transaction sent: 0x1ecede6b109b707786d3685bd71dd9f22dc389957653036ca04c4cd2e72c5e0b
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 44
  test.postiincrement confirmed   Block: 13622335   Gas used: 21620 (0.32%)

<Transaction '0x1ecede6b109b707786d3685bd71dd9f22dc389957653036ca04c4cd2e72c5e0b'>
>>> test_contract.preiincrement(1)
Transaction sent: 0x205f09a4d2268de4c1a40f35bb2ec2847bf2ab8d584909b42c71a022b047614a
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 45
  test.preiincrement confirmed   Block: 13622336   Gas used: 21593 (0.32%)

<Transaction '0x205f09a4d2268de4c1a40f35bb2ec2847bf2ab8d584909b42c71a022b047614a'>
>>> test_contract.postiincrement(10)
Transaction sent: 0x98c04430526a59balcf947c114b62666a4417165947d3lbf300cd6ae68328033
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 46
  test.postiincrement confirmed   Block: 13622337   Gas used: 22673 (0.34%)

<Transaction '0x98c04430526a59balcf947c114b62666a4417165947d3lbf300cd6ae68328033'>
>>> test_contract.preiincrement(10)
Transaction sent: 0xf060d04714eff8482a828342414d5a20be9958c822d42860e7992aba20elde05
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 47
  test.preiincrement confirmed   Block: 13622338   Gas used: 22601 (0.34%)

<Transaction '0xf060d04714eff8482a828342414d5a20be9958c822d42860e7992aba20elde05'>
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to use `++i` instead of `i++` to increment the value of an `uint` variable inside a loop. This is not applicable outside of loops.

Remediation Plan:

ACKNOWLEDGED: The Yieldly.Finance team acknowledged this finding.

3.11 (HAL-11) UPGRADE AT LEAST PRAGMA 0.8.4 - INFORMATIONAL

Description:

Gas optimizations and additional safety checks are available for free when using newer compiler versions and the optimizer.

- Safemath by default from 0.8.0 (can be more gas efficient than library based safemath.)
- Low level inliner : from 0.8.2, leads to cheaper runtime gas. Especially relevant when the contract has small functions. For example, OpenZeppelin libraries typically have a lot of small helper functions and if they are not inlined, they cost an additional 20 to 40 gas because of 2 extra jump instructions and additional stack operations needed for function calls.
- Optimizer improvements in packed structs: Before 0.8.3, storing packed structs, in some cases used an additional storage read operation. After EIP-2929, if the slot was already cold, this means unnecessary stack operations and extra deploy time costs. However, if the slot was already warm, this means additional cost of 100 gas alongside the same unnecessary stack operations and extra deploy time costs.
- Custom errors from 0.8.4, leads to cheaper deploy time cost and run time cost. Note: the run time cost is only relevant when the revert condition is met. In short, replace revert strings by custom errors.

Code Location:

Listing 14: Yieldly.sol

```
1 pragma solidity ^0.8.0;
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Upgrade **pragma** to at least 0.8.4.

Remediation Plan:

ACKNOWLEDGED: The Yieldly.Finance team acknowledged this finding.

3.12 (HAL-12) FLOATING PRAGMA - INFORMATIONAL

Description:

Contracts should be deployed with the same compiler version and flags used during development and testing. Locking the pragma helps to ensure that contracts do not accidentally get deployed using another pragma. For example, an outdated pragma version might introduce bugs that affect the contract system negatively or recently released pragma versions may have unknown security vulnerabilities.

Code Location:

Listing 15

```
1 pragma solidity ^0.8.0;
2 pragma solidity ^0.8.4;
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider locking the pragma version. It is not recommended using a floating pragma in production. It is possible to lock the pragma by fixing the version both in truffle-config.js for Truffle framework or in hardhat.config.js for HardHat framework.

Remediation Plan:

ACKNOWLEDGED: The Yieldly.Finance team acknowledged this finding.

3.13 (HAL-13) POSSIBLE MISUSE OF PUBLIC FUNCTIONS - INFORMATIONAL

Description:

In the following contracts there are functions marked as public but they are never directly called within the same contract or in any of their descendants:

Marketplace.sol

- closeTrade() (contracts/Marketplace.sol#279-286)
- withdraw() (contracts/Marketplace.sol#315-318)
- updatePrice() (contracts/Marketplace.sol#320-327)
- updateListingStatus() (contracts/Marketplace.sol#329-337)
- mint() (contracts/Marketplace.sol#518-535)
- Yieldly.sol
- mintAll() (contracts/Yieldly.sol#23-33)

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

If the functions are not intended to be called internally or by their descendants, it is better to mark all of these functions as external to reduce gas costs.

Remediation Plan:

ACKNOWLEDGED: The Yieldly.Finance team acknowledged this finding.

3.14 (HAL-14) REDUNDANT IMPORT - INFORMATIONAL

Description:

Hardhat console is an unnecessary import in all contracts since it is used solely for development. It can therefore be removed.

Code Location:

Listing 16

```
1 pragma solidity ^0.8.4;
2
3 import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
4
5 import "@openzeppelin/contracts/utils/math/SafeMath.sol";
6 import "./contracts/ReentrancyGuard.sol";
7 import "./interfaces/IMarket.sol";
8 import "./Yieldly.sol";
9
10 import "@openzeppelin/contracts/utils/Counters.sol";
11 import "@openzeppelin/contracts/access/Ownable.sol";
12 import "hardhat/console.sol";
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Delete import hardhat/console.sol on the deployment.

Remediation Plan:

ACKNOWLEDGED: The Yieldly.Finance team acknowledged this finding.

3.15 (HAL-15) LESS THAN 256 UINTS ARE NOT GAS EFFICIENT - INFORMATIONAL

Description:

Lower than uint256 size storage instance variables are actually less gas efficient. E.g. using uint16 does not give any efficiency, actually, it is the opposite as EVM operates on default of 256-bit values so uint8 is more expensive in this case as it needs a conversion. It only gives improvements in cases where you can pack variables together, e.g. structs.

Code Location:

Listing 17

```
1     // The minimum amount of time left in an auction after a new
          bid is created; 15 min.
2     uint16 public constant TIME_BUFFER = 900;
3     // The MATIC needed above the current bid for a new bid to be
          valid; 0.001 MATIC.
4     uint8 public constant MIN_BID_INCREMENT_PERCENT = 10;
5     // Interface constant for ERC721, to check values in
          constructor.
6     bytes4 private constant ERC721_INTERFACE_ID = 0x80ac58cd;
7     // Allows external read `getVersion()` to return a version for
           the auction.
8     uint256 private constant RESERVE_AUCTION_VERSION = 1;
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider to review all uint types. Change them with uint256 If the integer is not necessary to present with 16.

Remediation Plan:

ACKNOWLEDGED: The Yieldly.Finance team acknowledged this finding.

3.16 (HAL-16) MISSING EVENT EMITTING - INFORMATIONAL

Description:

It has been observed that important functionality is missing emitting event for a function on the Marketplace.sol contract. The functions should emit events. Events are a method of informing the transaction initiator about the actions taken by the called function. It logs its emitted parameters in a specific log history, which can be accessed outside the contract using some filter parameters. These functions should emit events.

Code Location:

Listing 18

```
1     function turnOffAdminRecovery() external onlyAdminRecovery {
2         _adminRecoveryEnabled = false;
3     }
```

Listing 19

```
1     function openTrade(uint _id, uint256 _price)
2     public
3     {
4         require(ownerMap[_id] == msg.sender, "sender is not owner"
              );
5         require(listedMap[_id] == false, "Already opened");
6         Yieldly(nftContract).approve(address(this), _id);
7         Yieldly(nftContract).transferFrom(msg.sender, address(this
              ), _id);
8         listedMap[_id] = true;
9         price[_id] = _price;
10    }
```

Listing 20

```
1     function closeTrade(uint _id)
2     public
3     {
4         require(ownerMap[_id] == msg.sender, "sender is not owner"
              );
5         require(listedMap[_id] == true, "Already colsed");
6         Yieldly(nftContract).transferFrom(address(this), msg.
              sender, _id);
7         listedMap[_id] = false;
8     }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

For best security practices, consider as much as possible, declaring events at the end of the function. Events can be used to detect the end of the operation.

Remediation Plan:

ACKNOWLEDGED: The Yieldly.Finance team acknowledged this finding.

3.17 (HAL-17) REVERT STRING SIZE OPTIMIZATION - INFORMATIONAL

Description:

Shortening revert strings to fit in 32 bytes will decrease deploy time gas and will decrease runtime gas when the revert condition has been met.

Revert strings that are longer than 32 bytes require at least one additional mstore, along with additional overhead for computing memory offset, etc.

Code Location:

Listing 21

```
1         require(_newtokenIds.length == _creators.length, "tokenIDs
              and creators are not mismatched");
2         require(_newtokenIds.length == _prices.length, "tokenIDs
             and _prices are not mismatched");
3         require(_newtokenIds.length == _owners.length, "tokenIDs
             and _owners are not mismatched");
4         require(_newtokenIds.length == _royalties.length, "
             tokenIDs and _royalties are not mismatched");
5         require(_newtokenIds.length == _listedMap.length, "
             tokenIDs and _listedMap are not mismatched");
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Shorten the revert strings to fit in 32 bytes. That will affect gas optimization.

Remediation Plan:

ACKNOWLEDGED: The Yieldly.Finance team acknowledged this finding.

3.18 (HAL-18) APPROVE IS NOT REVOKED DURING THE CLOSE TRADE - INFORMATIONAL

Description:

On the opentrade function, Approval is given to the contract for the transfer. However, when the trade is closed, the contract does not reset the approval for the NFT.

Code Location:

Listing 22

```
1     function openTrade(uint _id, uint256 _price)
2     public
3     {
4         require(ownerMap[_id] == msg.sender, "sender is not owner"
              );
5         require(listedMap[_id] == false, "Already opened");
6         Yieldly(nftContract).approve(address(this), _id);
7         Yieldly(nftContract).transferFrom(msg.sender, address(this
              ), _id);
8         listedMap[_id] = true;
9         price[_id] = _price;
10    }
11
12    function closeTrade(uint _id)
13    public
14    {
15        require(ownerMap[_id] == msg.sender, "sender is not owner"
              );
16        require(listedMap[_id] == true, "Already colsed");
17        Yieldly(nftContract).transferFrom(address(this), msg.
              sender, _id);
18        listedMap[_id] = false;
19    }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider resetting the approval when the trade is closed.

Remediation Plan:

ACKNOWLEDGED: The Yieldly.Finance team acknowledged this finding.

3.19 (HAL-19) MISSING EXTERNAL OWNED ACCOUNT CHECK - INFORMATIONAL

Description:

The contracts can interact with the mint function and there is no check for the EOA account.

Code Location:

Listing 23

```
1 function mint(string memory _tokenURI, uint256 _price, bool
      _isListOnMarketplace, uint256 _royalty, uint256 startDate,
      uint256 auctionLength)
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider to allow only EOA account for the minting process.

Remediation Plan:

ACKNOWLEDGED: The Yieldly.Finance team acknowledged this finding.

3.20 (HAL-20) TYPO ON THE REVERT MESSAGE - INFORMATIONAL

Description:

In the closeTrade function, require(listedMap[_id] == true, "Already colsed"); has a typo on the comment.

Code Location:

Listing 24

```
1     function closeTrade(uint _id)
2     public
3     {
4         require(ownerMap[_id] == msg.sender, "sender is not owner"
              );
5         require(listedMap[_id] == true, "Already colsed");
6         Yieldly(nftContract).transferFrom(address(this), msg.
              sender, _id);
7         listedMap[_id] = false;
8     }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider replacing colsed with closed.

Remediation Plan:

ACKNOWLEDGED: The Yieldly.Finance team acknowledged this finding.

3.21 (HAL-21) SAFEMATH IS NOT NEEDED WHEN USING SOLIDITY VERSION 0.8 - INFORMATIONAL

Description:

The Marketplace contract uses Solidity version 0.8 which already implements overflow checks by default.

At the same time, it uses the SafeMath library which is more gas expensive than the 0.8 overflow checks.

Code Location:

Listing 25

```
1     using SafeMath for uint256;
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

The contract should use the built-in checks and remove SafeMath from the dependencies.

Remediation Plan:

ACKNOWLEDGED: The Yieldly.Finance team acknowledged this finding.

3.22 (HAL-22) CACHING THE LENGTH IN THE FOR LOOPS - INFORMATIONAL

Description:

The solidity compiler will always read the length of the array during each iteration.

- If it is a storage array, this is an extra sload operation (100 additional extra gas (EIP-2929) for each iteration except for the first)
- If it is a memory array, this is an extra mload operation (3 additional gas for each iteration except for the first),
- If it is a calldata array, this is an extra calldataload operation (3 additional gas for each iteration except for the first).

Code Location:

Listing 26

```
        for (uint i = 0; i < _newtokenIds.length; i ++) {
            _tokenIds.increment();
            creatorMap[_newtokenIds[i]] = _creators[i];
            price[_newtokenIds[i]] = _prices[i];
            ownerMap[_newtokenIds[i]] = _owners[i];
            royaltyMap[_newtokenIds[i]] = _royalties[i];
            listedMap[_newtokenIds[i]] = _listedMap[i];
        }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Caching expensive state variables would avoid re-reading from storage.

Listing 27

```
uint length = arr.length;
for (uint i = 0; i < length; i++) {
    // do something that doesn't change arr.length
}
```

Remediation Plan:

ACKNOWLEDGED: The Yieldly.Finance team acknowledged this finding.

3.23 (HAL-23) MISSING ZERO/THRESHOLD CHECK FOR NFT SALE PRICE - INFORMATIONAL

Description:

A zero or some minimum threshold check is missing for _price parameter of **openTrade()** and **updatePrice** functions which sets the mint price for NFTs. If accidentally set to 0 then all sales happen at this incorrect price leading to missed revenue.

Code Location:

Listing 28

```
1     function openTrade(uint _id, uint256 _price)
2     public
3     {
4         require(ownerMap[_id] == msg.sender, "sender is not owner"
              );
5         require(listedMap[_id] == false, "Already opened");
6         Yieldly(nftContract).approve(address(this), _id);
7         Yieldly(nftContract).transferFrom(msg.sender, address(this
              ), _id);
8         listedMap[_id] = true;
9         price[_id] = _price;
10    }
```

Listing 29

```
1     function updatePrice(uint _tokenId, uint256 _price) public
          returns (bool) {
2         uint oldPrice = price[_tokenId];
3         require(msg.sender == ownerMap[_tokenId], "Error, you are
              not the owner");
4         price[_tokenId] = _price;
5
6         emit PriceUpdate(msg.sender, oldPrice, _price, _tokenId);
```

```
7           return true;
8       }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Add zero/threshold check for _price parameter in the functions.

Remediation Plan:

ACKNOWLEDGED: The Yieldly.Finance team acknowledged this finding.

3.24 (HAL-24) MINT/BUY FUNCTION CAN BE FRONT-RUN - INFORMATIONAL

Description:

On the blockchains any such mint/buy call can be observed and attackers can simply wait until another person decides to buy at the current price and then front run that person.

Code Location:

Listing 30

```
1     function mint(string memory _tokenURI, uint256 _price, bool
          _isListOnMarketplace, uint256 _royalty, uint256 startDate,
          uint256 auctionLength) public
```

Listing 31

```
1     function buy(uint _id, uint256 _price) external payable
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Front-running is hard to prevent, maybe an auction-style minting process could work where the top SALE_LIMIT bids are accepted after the sale duration.

Remediation Plan:

ACKNOWLEDGED: The Yieldly.Finance team acknowledged this finding.

3.25 (HAL-25) MINTING PROCESS WILL FAIL IF THE DESTINATION ADDRESS IS CONTRACT - INFORMATIONAL

Description:

In the **safeMint** function, If the target address is a contract, it must implement onERC721Received, which is called upon a safe transfer, and return the magic value `bytes4(keccak256("onERC721Received(address,address,uint256,bytes)"));` otherwise, the transfer is reverted.

Code Location:

Listing 32

```
1     function mint(
2         uint256 _id,
3         address _to,
4         string memory  _tokenURI
5     ) public
6     {
7         _safeMint(_to, _id);
8         _setTokenURI(_id, _tokenURI);
9     }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Ensure that If the platform supports contracts on the minting process.

- Reference : https://docs.openzeppelin.com/contracts/2.x/api/token/erc721

Remediation Plan:

ACKNOWLEDGED: The Yieldly.Finance team acknowledged this finding.

AUTOMATED TESTING

4.1 STATIC ANALYSIS REPORT

Description:

Halborn used automated testing techniques to enhance the coverage of certain areas of the scoped contracts. Among the tools used was Slither, a Solidity static analysis framework. After Halborn verified all the contracts in the repository and was able to compile them correctly into their abi and binary formats, Slither was run on the all-scoped contracts. This tool can statically verify mathematical relationships between Solidity variables to detect invalid or inconsistent usage of the contracts' APIs across the entire code-base.

Slither results:

Marketplace.sol

```
YieldlyMarketplace.attemptMATICTransfer(address,uint256) (contracts/Marketplace.sol#613-622) sends eth to arbitrary user
        Dangerous calls:
        - (success) = to.call{gas: 30000,value: value}() (contracts/Marketplace.sol#620)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations

Reentrancy in YieldlyMarketplace.buy(uint256,uint256) (contracts/Marketplace.sol#288-306):
        External calls:
        - transferMATICOrWMATIC(address(_previousOwner),_sellerValue) (contracts/Marketplace.sol#299)
                - IWMATIC(WMATICAddress).deposit{value: value}() (contracts/Marketplace.sol#603)
                - (success) = to.call{gas: 30000,value: value}() (contracts/Marketplace.sol#620)
                - IWMATIC(WMATICAddress).transfer(to,value) (contracts/Marketplace.sol#604)
        - transferMATICOrWMATIC(address(creatorMap[_id]),_royaltyValue) (contracts/Marketplace.sol#300)
                - IWMATIC(WMATICAddress).deposit{value: value}() (contracts/Marketplace.sol#603)
                - (success) = to.call{gas: 30000,value: value}() (contracts/Marketplace.sol#620)
                - IWMATIC(WMATICAddress).transfer(to,value) (contracts/Marketplace.sol#604)
        - transferMATICOrWMATIC(address(adminRecoveryAddress),_commissionValue) (contracts/Marketplace.sol#301)
                - IWMATIC(WMATICAddress).deposit{value: value}() (contracts/Marketplace.sol#603)
                - (success) = to.call{gas: 30000,value: value}() (contracts/Marketplace.sol#620)
                - IWMATIC(WMATICAddress).transfer(to,value) (contracts/Marketplace.sol#604)
        - Yieldly(nftContract).transferFrom(address(this),_newOwner,_id) (contracts/Marketplace.sol#302)
        External calls sending eth:
        - transferMATICOrWMATIC(address(_previousOwner),_sellerValue) (contracts/Marketplace.sol#299)
                - IWMATIC(WMATICAddress).deposit{value: value}() (contracts/Marketplace.sol#603)
                - (success) = to.call{gas: 30000,value: value}() (contracts/Marketplace.sol#620)
        - transferMATICOrWMATIC(address(creatorMap[_id]),_royaltyValue) (contracts/Marketplace.sol#300)
                - IWMATIC(WMATICAddress).deposit{value: value}() (contracts/Marketplace.sol#603)
                - (success) = to.call{gas: 30000,value: value}() (contracts/Marketplace.sol#620)
        - transferMATICOrWMATIC(address(adminRecoveryAddress),_commissionValue) (contracts/Marketplace.sol#301)
                - IWMATIC(WMATICAddress).deposit{value: value}() (contracts/Marketplace.sol#603)
                - (success) = to.call{gas: 30000,value: value}() (contracts/Marketplace.sol#620)
        State variables written after the call(s):
        - listedMap[_id] = false (contracts/Marketplace.sol#304)
        - ownerMap[_id] = msg.sender (contracts/Marketplace.sol#303)
Reentrancy in YieldlyMarketplace.createBid(uint256,uint256) (contracts/Marketplace.sol#389-446):
        External calls:
        - transferMATICOrWMATIC(auctions[tokenId].bidder,auctions[tokenId].amount) (contracts/Marketplace.sol#427-430)
                - IWMATIC(WMATICAddress).deposit{value: value}() (contracts/Marketplace.sol#603)
                - (success) = to.call{gas: 30000,value: value}() (contracts/Marketplace.sol#620)
                - IWMATIC(WMATICAddress).transfer(to,value) (contracts/Marketplace.sol#604)
        External calls sending eth:
        - transferMATICOrWMATIC(auctions[tokenId].bidder,auctions[tokenId].amount) (contracts/Marketplace.sol#427-430)
                - IWMATIC(WMATICAddress).deposit{value: value}() (contracts/Marketplace.sol#603)
                - (success) = to.call{gas: 30000,value: value}() (contracts/Marketplace.sol#620)
        State variables written after the call(s):
        - auctions[tokenId].amount = amount (contracts/Marketplace.sol#433)
        - auctions[tokenId].bidder = address(msg.sender) (contracts/Marketplace.sol#434)
        - auctions[tokenId].duration += block.timestamp.add(TIME_BUFFER).sub(auctionEnds(tokenId)) (contracts/Marketplace.sol#440-442)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities

YieldlyMarketplace.transferMATICOrWMATIC(address,uint256) (contracts/Marketplace.sol#597-607) ignores return value by IWMATIC(WMATICAddress).transfer(to,value) (contracts/Marketplace.sol#604)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
```

```
Reentrancy in YieldlyMarketplace.closeTrade(uint256) (contracts/Marketplace.sol#279-286):
        External calls:
        - Yieldly(nftContract).transferFrom(address(this),msg.sender,_id) (contracts/Marketplace.sol#284)
        State variables written after the call(s):
        - listedMap[_id] = false (contracts/Marketplace.sol#285)
Reentrancy in YieldlyMarketplace.createAuction(uint256,uint256,uint256,uint256) (contracts/Marketplace.sol#341-385):
        External calls:
        - openTrade(tokenId,reservePrice) (contracts/Marketplace.sol#360)
                - Yieldly(nftContract).approve(address(this),_id) (contracts/Marketplace.sol#273)
                - Yieldly(nftContract).transferFrom(msg.sender,address(this),_id) (contracts/Marketplace.sol#274)
        State variables written after the call(s):
        - auctions[tokenId] = Auction(duration,reservePrice,50,msg.sender,address(adminRecoveryAddress),0,startDate,address(address(0))) (contracts/Marketplace.sol#364-373)
Reentrancy in YieldlyMarketplace.openTrade(uint256,uint256) (contracts/Marketplace.sol#268-277):
        External calls:
        - Yieldly(nftContract).approve(address(this),_id) (contracts/Marketplace.sol#273)
        - Yieldly(nftContract).transferFrom(msg.sender,address(this),_id) (contracts/Marketplace.sol#274)
        State variables written after the call(s):
        - listedMap[_id] = true (contracts/Marketplace.sol#275)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#382-403) ignores return value by IERC721Receiver(to).onERC721Received(_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

Yieldly.mintAll(uint256[],address[],string[])._tokenURIs (contracts/Yieldly.sol#26) shadows:
        - ERC721URIStorage._tokenURIs (node_modules/@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol#15) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

YieldlyMarketplace.constructor(address,address,address).WMATICAddress_ (contracts/Marketplace.sol#234) lacks a zero-check on :
                - WMATICAddress = WMATICAddress_ (contracts/Marketplace.sol#243)
YieldlyMarketplace.constructor(address,address,address).adminRecoveryAddress_ (contracts/Marketplace.sol#235) lacks a zero-check on :
                - adminRecoveryAddress = adminRecoveryAddress_ (contracts/Marketplace.sol#244)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).retval (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#389)' in ERC721._checkOnERC721Received(address,address,uint256,
bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#382-403) potentially used before declaration: retval == IERC721Receiver.onERC721Received.selector (node_modules/@openzeppelin/contract
s/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(address,address,uint256,
bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#382-403) potentially used before declaration: reason.length == 0 (node_modules/@openzeppelin/contracts/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(address,address,uint256,
bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#382-403) potentially used before declaration: revert(uint256,uint256)(32 + reason,mload(uint256)(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
Reentrancy in YieldlyMarketplace.endAuction(uint256) (contracts/Marketplace.sol#450-491):
        External calls:
        - Yieldly(nftContract).transferFrom(address(this),Creator,tokenId) (contracts/Marketplace.sol#466)
        - Yieldly(nftContract).transferFrom(address(this),winner,tokenId) (contracts/Marketplace.sol#469)
        - transferMATICOrWMATIC(address(adminRecoveryAddress),_commissionValue) (contracts/Marketplace.sol#471)
                - IWMATIC(WMATICAddress).deposit{value: value}() (contracts/Marketplace.sol#603)
                - (success) = to.call{gas: 30000,value: value}() (contracts/Marketplace.sol#620)
                - IWMATIC(WMATICAddress).transfer(to,value) (contracts/Marketplace.sol#604)
        - transferMATICOrWMATIC(address(Creator),amount.sub(_commissionValue)) (contracts/Marketplace.sol#473)
                - IWMATIC(WMATICAddress).deposit{value: value}() (contracts/Marketplace.sol#603)
                - (success) = to.call{gas: 30000,value: value}() (contracts/Marketplace.sol#620)
                - IWMATIC(WMATICAddress).transfer(to,value) (contracts/Marketplace.sol#604)
        - transferMATICOrWMATIC(address(creatorMap[tokenId]),_royaltyValue) (contracts/Marketplace.sol#476)
                - IWMATIC(WMATICAddress).deposit{value: value}() (contracts/Marketplace.sol#603)
                - (success) = to.call{gas: 30000,value: value}() (contracts/Marketplace.sol#620)
                - IWMATIC(WMATICAddress).transfer(to,value) (contracts/Marketplace.sol#604)
        - transferMATICOrWMATIC(address(Creator),amount.sub(_royaltyValue).sub(_commissionValue)) (contracts/Marketplace.sol#477)
                - IWMATIC(WMATICAddress).deposit{value: value}() (contracts/Marketplace.sol#603)
                - (success) = to.call{gas: 30000,value: value}() (contracts/Marketplace.sol#620)
                - IWMATIC(WMATICAddress).transfer(to,value) (contracts/Marketplace.sol#604)
        External calls sending eth:
        - transferMATICOrWMATIC(address(adminRecoveryAddress),_commissionValue) (contracts/Marketplace.sol#471)
                - IWMATIC(WMATICAddress).deposit{value: value}() (contracts/Marketplace.sol#603)
                - (success) = to.call{gas: 30000,value: value}() (contracts/Marketplace.sol#620)
        - transferMATICOrWMATIC(address(Creator),amount.sub(_commissionValue)) (contracts/Marketplace.sol#473)
                - IWMATIC(WMATICAddress).deposit{value: value}() (contracts/Marketplace.sol#603)
                - (success) = to.call{gas: 30000,value: value}() (contracts/Marketplace.sol#620)
        - transferMATICOrWMATIC(address(creatorMap[tokenId]),_royaltyValue) (contracts/Marketplace.sol#476)
                - IWMATIC(WMATICAddress).deposit{value: value}() (contracts/Marketplace.sol#603)
                - (success) = to.call{gas: 30000,value: value}() (contracts/Marketplace.sol#620)
        - transferMATICOrWMATIC(address(Creator),amount.sub(_royaltyValue).sub(_commissionValue)) (contracts/Marketplace.sol#477)
                - IWMATIC(WMATICAddress).deposit{value: value}() (contracts/Marketplace.sol#603)
                - (success) = to.call{gas: 30000,value: value}() (contracts/Marketplace.sol#620)
        State variables written after the call(s):
        - listedMap[tokenId] = false (contracts/Marketplace.sol#481)
Reentrancy in YieldlyMarketplace.mint(string,uint256,bool,uint256,uint256,uint256) (contracts/Marketplace.sol#518-535):
        External calls:
        - Yieldly(nftContract).mint(newTokenId,msg.sender,_tokenURI) (contracts/Marketplace.sol#530)
        - createAuction(newTokenId,startDate,auctionLength,_price) (contracts/Marketplace.sol#534)
                - Yieldly(nftContract).approve(address(this),_id) (contracts/Marketplace.sol#273)
                - Yieldly(nftContract).transferFrom(msg.sender,address(this),_id) (contracts/Marketplace.sol#274)
        State variables written after the call(s):
        - createAuction(newTokenId,startDate,auctionLength,_price) (contracts/Marketplace.sol#534)
                - price[_id] = _price (contracts/Marketplace.sol#276)
Reentrancy in Yieldly.mint(uint256,address,string) (contracts/Yieldly.sol#14-22):
        External calls:
        - _safeMint(_to,_id) (contracts/Yieldly.sol#20)
                - 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(_id,_tokenURI) (contracts/Yieldly.sol#21)
                - _tokenURIs[tokenId] = _tokenURI (node_modules/@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol#47)
YieldlyMarketplace.createBid(uint256,uint256) (contracts/Marketplace.sol#389-446) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(auctions[tokenId].firstBidTime <= block.timestamp,Auction is not yet started) (contracts/Marketplace.sol#400)
        - auctionEnds(tokenId) < block.timestamp.add(TIME_BUFFER) (contracts/Marketplace.sol#437)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

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

YieldlyMarketplace.openTrade(uint256,uint256) (contracts/Marketplace.sol#268-277) compares to a boolean constant:
        -require(bool,string)(listedMap[_id] == false,Already opened) (contracts/Marketplace.sol#272)
YieldlyMarketplace.closeTrade(uint256) (contracts/Marketplace.sol#279-286) compares to a boolean constant:
        -require(bool,string)(listedMap[_id] == true,Already colsed) (contracts/Marketplace.sol#283)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Different versions of Solidity is used:
        - Version used: ['0.8.2', '^0.8.0', '^0.8.2']
        - ^0.8.2 (contracts/Marketplace.sol#3)
        - ^0.8.0 (contracts/Yieldly.sol#3)
        - 0.8.2 (contracts/contracts/ReentrancyGuard.sol#3)
        - 0.8.2 (contracts/Interfaces/Decimal.sol#3)
        - 0.8.2 (contracts/Interfaces/IMarket.sol#3)
        - 0.8.2 (contracts/Interfaces/Math.sol#3)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/IERC721.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/IERC721Receiver.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/extensions/IERC721Metadata.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Counters.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Strings.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/ERC165.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/introspection/IERC165.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#4)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
```

AUTOMATED TESTING

```
Address.functionCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#80-82) is never used and should be removed
Address.functionCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#90-96) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#109-115) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#123-134) is never used and should be removed
Address.functionDelegateCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#169-171) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#179-188) is never used and should be removed
Address.functionStaticCall(address,bytes) (node_modules/@openzeppelin/contracts/utils/Address.sol#142-144) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#152-161) is never used and should be removed
Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#55-60) is never used and should be removed
Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#196-216) is never used and should be removed
Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.sol#21-23) is never used and should be removed
Counters.decrement(Counters.Counter) (node_modules/@openzeppelin/contracts/utils/Counters.sol#32-38) is never used and should be removed
Decimal.div(uint256,Decimal.D256) (contracts/interfaces/Decimal.sol#43-49) is never used and should be removed
Decimal.mul(uint256,Decimal.D256) (contracts/interfaces/Decimal.sol#35-41) is never used and should be removed
Decimal.one() (contracts/interfaces/Decimal.sol#27-29) is never used and should be removed
Decimal.onePlus(Decimal.D256) (contracts/interfaces/Decimal.sol#31-33) is never used and should be removed
ERC721._burn(uint256) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#302-314) is never used and should be removed
ERC721URIStorage._burn(uint256) (node_modules/@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol#60-66) is never used and should be removed
Math.getPartial(uint256,uint256,uint256) (contracts/interfaces/Math.sol#23-29) is never used and should be removed
Math.getPartialRoundUp(uint256,uint256,uint256) (contracts/interfaces/Math.sol#34-44) is never used and should be removed
Math.max(uint256,uint256) (contracts/interfaces/Math.sol#68-70) is never used and should be removed
Math.min(uint256,uint256) (contracts/interfaces/Math.sol#64-66) is never used and should be removed
Math.to128(uint256) (contracts/interfaces/Math.sol#46-50) is never used and should be removed
Math.to32(uint256) (contracts/interfaces/Math.sol#58-62) is never used and should be removed
Math.to96(uint256) (contracts/interfaces/Math.sol#52-56) is never used and should be removed
SafeMath.div(uint256,uint256,string) (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#191-200) is never used and should be removed
SafeMath.mod(uint256,uint256) (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#151-153) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#217-226) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#168-177) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#22-28) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#64-69) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#76-81) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#47-57) is never used and should be removed
SafeMath.trySub(uint256,uint256) (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#35-40) is never used and should be removed
Strings.toHexString(uint256) (node_modules/@openzeppelin/contracts/utils/Strings.sol#40-51) is never used and should be removed
Strings.toHexString(uint256,uint256) (node_modules/@openzeppelin/contracts/utils/Strings.sol#56-66) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.2 (contracts/Marketplace.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Yieldly.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.2 (contracts/contracts/ReentrancyGuard.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.2 (contracts/interfaces/Decimal.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.2 (contracts/interfaces/IMarket.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.2 (contracts/interfaces/Math.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/IERC721.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/IERC721Receiver.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC721/extensions/IERC721Metadata.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6

Low level call in YieldlyMarketplace.attemptMATICTransfer(address,uint256) (contracts/Marketplace.sol#613-622):
        - (success) = to.call{gas: 30000,value: value}() (contracts/Marketplace.sol#620)
Low level call in Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts/utils/Address.sol#55-60):
        - (success) = recipient.call{value: amount}() (node_modules/@openzeppelin/contracts/utils/Address.sol#58)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#123-134):
        - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#132)
Low level call in Address.functionStaticCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#152-161):
        - (success,returndata) = target.staticcall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#159)
Low level call in Address.functionDelegateCall(address,bytes,string) (node_modules/@openzeppelin/contracts/utils/Address.sol#179-188):
        - (success,returndata) = target.delegatecall(data) (node_modules/@openzeppelin/contracts/utils/Address.sol#186)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter YieldlyMarketplace.addCreatorMap(bool,uint256[],address[],uint256[],address[],uint256[],bool[])._isNew (contracts/Marketplace.sol#250) is not in mixedCase
Parameter YieldlyMarketplace.addCreatorMap(bool,uint256[],address[],uint256[],address[],uint256[],bool[])._newtokenIds (contracts/Marketplace.sol#250) is not in mixedCase
Parameter YieldlyMarketplace.addCreatorMap(bool,uint256[],address[],uint256[],address[],uint256[],bool[])._creators (contracts/Marketplace.sol#250) is not in mixedCase
Parameter YieldlyMarketplace.addCreatorMap(bool,uint256[],address[],uint256[],address[],uint256[],bool[])._prices (contracts/Marketplace.sol#250) is not in mixedCase
Parameter YieldlyMarketplace.addCreatorMap(bool,uint256[],address[],uint256[],address[],uint256[],bool[])._owners (contracts/Marketplace.sol#250) is not in mixedCase
Parameter YieldlyMarketplace.addCreatorMap(bool,uint256[],address[],uint256[],address[],uint256[],bool[])._royalties (contracts/Marketplace.sol#250) is not in mixedCase
Parameter YieldlyMarketplace.addCreatorMap(bool,uint256[],address[],uint256[],address[],uint256[],bool[])._listedMap (contracts/Marketplace.sol#250) is not in mixedCase
Parameter YieldlyMarketplace.openTrade(uint256,uint256)._id (contracts/Marketplace.sol#268) is not in mixedCase
Parameter YieldlyMarketplace.openTrade(uint256,uint256)._price (contracts/Marketplace.sol#268) is not in mixedCase
Parameter YieldlyMarketplace.closeTrade(uint256)._id (contracts/Marketplace.sol#279) is not in mixedCase
Parameter YieldlyMarketplace.buy(uint256,uint256)._id (contracts/Marketplace.sol#288) is not in mixedCase
Parameter YieldlyMarketplace.buy(uint256,uint256)._price (contracts/Marketplace.sol#288) is not in mixedCase
Parameter YieldlyMarketplace.updatePrice(uint256,uint256)._tokenId (contracts/Marketplace.sol#320) is not in mixedCase
Parameter YieldlyMarketplace.updatePrice(uint256,uint256)._price (contracts/Marketplace.sol#320) is not in mixedCase
Parameter YieldlyMarketplace.updateListingStatus(uint256,bool)._tokenId (contracts/Marketplace.sol#329) is not in mixedCase
Parameter YieldlyMarketplace.mint(string,uint256,bool,uint256,uint256,uint256)._tokenURI (contracts/Marketplace.sol#518) is not in mixedCase
Parameter YieldlyMarketplace.mint(string,uint256,bool,uint256,uint256,uint256)._price (contracts/Marketplace.sol#518) is not in mixedCase
Parameter YieldlyMarketplace.mint(string,uint256,bool,uint256,uint256,uint256)._isListOnMarketplace (contracts/Marketplace.sol#518) is not in mixedCase
Parameter YieldlyMarketplace.mint(string,uint256,bool,uint256,uint256,uint256)._royalty (contracts/Marketplace.sol#518) is not in mixedCase
Variable YieldlyMarketplace.WMATICAddress (contracts/Marketplace.sol#52) is not in mixedCase
Parameter Yieldly.mint(uint256,address,string)._id (contracts/Yieldly.sol#15) is not in mixedCase
Parameter Yieldly.mint(uint256,address,string)._to (contracts/Yieldly.sol#16) is not in mixedCase
Parameter Yieldly.mint(uint256,address,string)._tokenURI (contracts/Yieldly.sol#17) is not in mixedCase
Parameter Yieldly.mintAll(uint256[],address[],string[])._ids (contracts/Yieldly.sol#24) is not in mixedCase
Parameter Yieldly.mintAll(uint256[],address[],string[])._tos (contracts/Yieldly.sol#25) is not in mixedCase
Parameter Yieldly.mintAll(uint256[],address[],string[])._tokenURIs (contracts/Yieldly.sol#26) is not in mixedCase
Parameter ERC721.safeTransferFrom(address,address,uint256,bytes)._data (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#179) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

IMediaModified.marketContract (contracts/Marketplace.sol#18) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
```

Yieldly.sol

```
ERC721._checkOnERC721Received(address,address,uint256,bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#382-403) ignores return value by IERC721Receiver(to).onERC721Received(_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

Yieldly.mintAll(uint256[],address[],string[])._tokenURIs (contracts/Yieldly.sol#26) shadows:
        - ERC721URIStorage._tokenURIs (node_modules/@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol#15) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).retval (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#389)' in ERC721._checkOnERC721Received(address,address,uint256,
bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#382-403) potentially used before declaration: retval == IERC721Receiver.onERC721Received.selector (node_modules/@openzeppelin/contract
s/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(address,address,uint256,
bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#382-403) potentially used before declaration: reason.length == 0 (node_modules/@openzeppelin/contracts/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(address,address,uint256,
bytes) (node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol#382-403) potentially used before declaration: revert(uint256,uint256)(32 + reason,mload(uint256)(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

Reentrancy in Yieldly.mint(uint256,address,string) (contracts/Yieldly.sol#14-22):
        External calls:
        - _safeMint(_to,_id) (contracts/Yieldly.sol#20)
                - 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(_id,_tokenURI) (contracts/Yieldly.sol#21)
                - _tokenURIs[tokenId] = _tokenURI (node_modules/@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol#47)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

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
```

- No major issues were found by Slither.

THANK YOU FOR CHOOSING

// HALBORN