# SMART CONTRACT

## Security Audit Report

| | |
|---|---|
| Project: | 99 Originals |
| Website: | https://originals.com |
| Platform: | Ethereum |
| Language: | Solidity |
| Date: | April 16th, 2022 |

# Table of contents

`

THIS IS SECURITY AUDIT REPORT DOCUMENT AND WHICH MAY CONTAIN INFORMATION WHICH IS CONFIDENTIAL. WHICH INCLUDES ANY POTENTIAL VULNERABILITIES AND MALICIOUS CODES WHICH CAN BE USED TO EXPLOIT THE SOFTWARE. THIS MUST BE REFERRED INTERNALLY AND ONLY SHOULD BE MADE AVAILABLE TO THE PUBLIC AFTER ISSUES ARE RESOLVED.

# Introduction

Dru Schroer was contracted by the Original99 team to perform the Security audit of the 99 Originals NFT Token smart contract code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on April 16th, 2022.

**The purpose of this audit was to address the following:**

- Ensure that all claimed functions exist and function correctly.
- Identify any security vulnerabilities that may be present in the smart contract.

# Project Background

Original99 Contract is an NFT smart contract, having functions like withdraw, finalizeAuction, placeBid, tokenURI, getWinningBidder, etc. The Original99 contract inherits the Ownable, ReentrancyGuard, ERC721Royalty standard smart contracts from the OpenZeppelin library. These OpenZeppelin contracts are considered community-audited and time-tested, and hence are not part of the audit scope.

# Audit scope

| **Name** | **Code Review and Security Analysis Report for Original99 Token Smart Contract** |
|---|---|
| **Platform** | **Ethereum / Solidity** |
| **File** | Original99.sol |
| **File MD5 Hash** | 4C2F8E3FB648C99EE3669E3DBE58164F |
| **Online Code Link** | 0xf367653cfece1b13bdddc0a506599b9a101d2a78 |
| **Audit Date** | April 16th, 2022 |
| **Revised Code** | 0x1c49ed56da6be87b804bc1b8b817a259aa3132ed |
| **Revision Date** | August 2nd, 2022 |

# ndClaimed Smart Contract Features

| Claimed Feature Detail | Our Observation |
| --- | --- |
| **Tokenomics:**<br>● Name: 99 Originals<br>● Symbol: 99ORIG<br>● Auction Duration: 15 Minutes<br>● Auction Duration Extension: 5 Minutes<br>● Price Initial: 0.1 Ether<br>● Min Bid Increase BPS: 10%<br>● Total Supply: 99 | **YES, This is valid.**<br><br>**Owner authorized wallet can set some percentage value and we suggest handling the private key of that wallet securely.** |

# Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **“Secured”**. This token contract does contain owner control, which does not make it fully decentralized.


We used various tools like Slither, Solhint and Remix IDE. At the same time this finding is based on critical analysis of the manual audit.
All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the Audit overview section. General overview is presented in AS-IS section and all identified issues can be found in the Audit overview section.

**We found 1 critical, 0 high, 0 medium and 1 low and some very low level issues. These issues are fixed / acknowledged in the revised smart contract code.**

**Investors Advice:** Technical audit of the smart contract does not guarantee the ethical nature of the project. Any owner controlled functions should be executed by the owner with responsibility. All investors/users are advised to do their due diligence before investing in the project.

# Technical Quick Stats

| Main Category | Subcategory | Result |
|---|---|---|
| Contract Programming | Solidity version not specified | Passed |
| | Solidity version too old | Passed |
| | Integer overflow/underflow | Passed |
| | DoS possibility | Passed |
| | Function input parameters check bypass | Passed |
| | Function access control lacks management | Passed |
| | Critical operation lacks event log | Moderated |
| | Human/contract checks bypass | Passed |
| | Random number generation/use vulnerability | N/A |
| | Fallback function misuse | Passed |
| | Race condition | Passed |
| | Logical vulnerability | Passed |
| | Features claimed | Passed |
| | Other programming issues | Passed |
| Code Specification | Function visibility not explicitly declared | Passed |
| | Var. storage location not explicitly declared | Passed |
| | Use keywords/functions to be deprecated | Passed |
| | Unused code | Passed |
| Gas Optimization | “Out of Gas” Issue | Passed |
| | High consumption ‘for/while’ loop | Passed |
| | High consumption ‘storage’ storage | Passed |
| | Assert() misuse | Passed |
| Business Risk | The maximum limit for mintage not set | Passed |
| | “Short Address” Attack | Passed |
| | “Double Spend” Attack | Passed |

**Overall Audit Result:** **PASSED**

## Code Quality

This audit scope has 1 smart contract. Smart contract contains Libraries, Smart contracts, inherits and Interfaces.  This is a compact and well written smart contract.

The libraries in Original99 Token are part of its logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned a specific address and its properties / methods can be reused many times by other contracts in the Original99 Token.

The Original99 Token team has **not** provided scenario and unit test scripts, which would have helped to determine the integrity of the code in an automated way.

Code parts are **not** commented on in the smart contracts. Ethereum's NatSpec commenting style is recommended.

## Documentation

We were given a Original99 Token smart contract code in the form of a Rinkeby Etherscan weblink. The hash of that code is mentioned above in the table.

As mentioned above, code parts are not well commented. But the contract is straightforward so it's easy to understand its programming logic.

Another source of the information was its website: https://originals.com which provided rich information about the project and its logical architecture.

## Use of Dependencies

As per our observation, the libraries are used in this smart contract infrastructure that are based on well known industry standard open source projects.

Apart from libraries,  its functions are used in external smart contract calls.

# AS-IS overview

### Functions

| Sl. | Functions | Type | Observation | Conclusion |
|---|---|---|---|---|
| **1** | constructor | write | Passed | No Issue |
| **2** | owner | read | Passed | No Issue |
| **3** | onlyOwner | modifier | Passed | No Issue |
| **4** | renounceOwnership | write | access only Owner | No Issue |
| **5** | transferOwnership | write | access only Owner | No Issue |
| **6** | _transferOwnership | internal | Passed | No Issue |
| **7** | supportsInterface | read | Passed | No Issue |
| **8** | _burn | internal | Passed | No Issue |
| **9** | onlyActiveAuction | modifier | Passed | No Issue |
| **10** | getAuctionEndTime | read | access only Active Auction | No Issue |
| **11** | getSmallestAllowedBid | read | access only Active Auction | No Issue |
| **12** | getInitialPrice | read | access only Active Auction | No Issue |
| **13** | getCurrentPrice | read | access only Active Auction | No Issue |
| **14** | getWinningBidder | read | access only Active Auction | No Issue |
| **15** | getAuctionedTokenId | read | Passed | No Issue |
| **16** | isSold | read | Passed | No Issue |
| **17** | isAuctionEnd | read | access only Active Auction | No Issue |
| **18** | getBaseURI | read | Passed | No Issue |
| **19** | setBaseURI | write | access only Owner | No Issue |
| **20** | _baseURI | internal | Passed | No Issue |
| **21** | tokenURI | read | Passed | No Issue |
| **22** | startAuction | external | access only Owner | No Issue |
| **23** | placeBid | external | Passed | No Issue |
| **24** | finalizeAuction | external | Passed | No Issue |
| **25** | withdraw | external | access only Owner | No Issue |

# Severity Definitions

| Risk Level | Description |
|---|---|
| Critical | Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc. |
| High | High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial |
| Medium | Medium-level vulnerabilities are important to fix; however, they can't lead to tokens lose |
| Low | Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution |
| Lowest / Code Style / Best Practice | Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored. |

# Audit Findings

## Critical Severity

(1) Denial of service (DoS) attack possibility

```
function placeBid() external payable onlyActiveAuction nonReentrant {
    require(msg.value >= getSmallestAllowedBid(), "Too Small Bid");
    require(!isAuctionEnd(), "Auction Ended");

    if (block.timestamp + DURATION_EXTENSION_ALLOWED_BEFORE_END >= auctionEndTime) {
        auctionEndTime = block.timestamp + DURATION_EXTENSION;
    }

    if (winningBidder != address(0)) {
        (bool success, ) = winningBidder.call{value: currentPrice}("");
        require(success, "Returning Escrowed Funds Failed");
    }

    currentPrice = msg.value;
    winningBidder = _msgSender();

    emit AuctionBidPlaced(auctionedTokenId, _msgSender(), msg.value, auctionEndTime);
}
```

An attacker can stop all other bidders and he can always be in the winning position.

More technical details: when an attacker places a bid using an attack smart contract which has a fallback function which will always revert. This will cause all subsequent bidder's calls to fail, as the placebid function gives ether to previous bidder.

**Resolution**: prevent all the smart contract calls from calling placeBid function, by making following condition:

require(msg.sender == tx.origin, "caller can not be contract address");

**Status: This issue is fixed in revised contract code**

## High Severity

No High severity vulnerabilities were found.

## Medium

No Medium severity vulnerabilities were found.

## Low

(1) Missing events in some admin functions:

It is recommended to fire appropriate events, where significant state change is being done. This helps UI clients to coordinate properly with the blockchain. such as:

- setBaseURI
- withdraw

**Resolution**: please add an event in above functions.

**Status: This issue is acknowledged**

## Very Low / Informational / Best practices:

(1) Only one auction can be active at a time:

This smart contract will only allow one auction listing at a time. If client want to run multiple NFT auction/minting, then it will not be possible

**Resolution**: If this is part of the plan, then please ignore this point. Otherwise, the logic should be changed which allows multiple auctions at a time

**Status: This issue is acknowledged**

(2) Make sure the hardcoded wallets are correct:

There are hardcoded wallets used in the smart contract code. Please make sure they are the correct ones before deploying to the mainnet.

**Status: This issue is acknowledged**

# Centralization

This smart contract has some functions which can be executed by the Admin (Owner) only. If the admin wallet private key would be compromised, then it would create trouble. Following are Admin functions:

- setBaseURI: Owner can update baseURI.
- startAuction: Owner can create new auction.
- withdraw: Owner can withdraw amount.

It is good practice to renounce the ownership once there is no ownership control needed. This will make the smart contract fully decentralized.

# Conclusion

We were given a contract code. And we have used all possible tests based on given objects as files. We have observed one major issue, and they are fixed / acknowledged. So, **it's good to go for the mainnet deployment**.

Since possible test cases can be unlimited for such smart contracts protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan everything.

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. Smart Contract's high-level description of functionality was presented in the As-is overview section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security state of the reviewed contract, based on standard audit procedure scope, is **"Secured"**.

# Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

**Manual Code Review:**
In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

**Vulnerability Analysis:**
Our audit techniques included manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's web site to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

**Documenting Results:**

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyze the feasibility of an attack in a live system.

**Suggested Solutions:**

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

# Disclaimers

The audit team has analyzed this smart contract in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

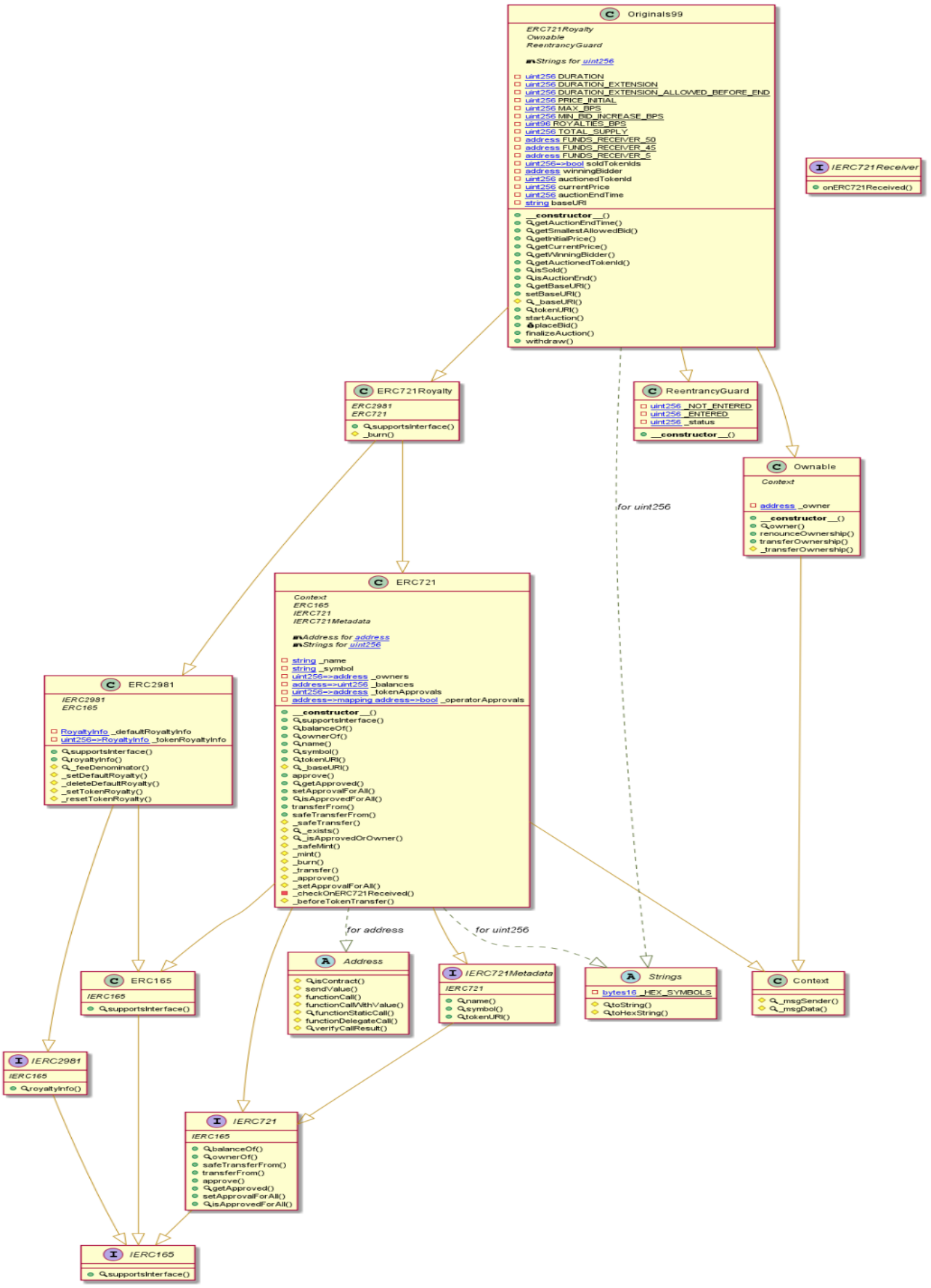
Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only. We also suggest conducting a bug bounty program to confirm the high level of security of this smart contract.

## Technical Disclaimer

Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can’t guarantee explicit security of the audited smart contracts.

# Appendix

## Code Flow Diagram - Original99 Token



# Slither Results Log

```
INFO:Detectors:
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).retval (Originals99.sol#1048)' in ERC721._checkOnERC
1Received(address,address,uint256,bytes) (Originals99.sol#1041-1062) potentially used before declaration: retval == IERC721
ceiver.onERC721Received.selector (Originals99.sol#1049)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (Originals99.sol#1050)' in ERC721._checkOnERC
1Received(address,address,uint256,bytes) (Originals99.sol#1041-1062) potentially used before declaration: reason.length ==
(Originals99.sol#1051)
Variable 'ERC721._checkOnERC721Received(address,address,uint256,bytes).reason (Originals99.sol#1050)' in ERC721._checkOnERC
1Received(address,address,uint256,bytes) (Originals99.sol#1041-1062) potentially used before declaration: revert(uint256,ui
256)(32 + reason,mload(uint256)(reason)) (Originals99.sol#1055)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables
INFO:Detectors:
Reentrancy in Originals99.finalizeAuction() (Originals99.sol#1234-1248):
        External calls:
        - _safeMint(winningBidder,auctionedTokenId) (Originals99.sol#1240)
                - IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,_data) (Originals99.sol#1048-1058)
        State variables written after the call(s):
        - currentPrice = 0 (Originals99.sol#1245)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in Originals99.finalizeAuction() (Originals99.sol#1234-1248):
        External calls:
        - _safeMint(winningBidder,auctionedTokenId) (Originals99.sol#1240)
                - IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,_data) (Originals99.sol#1048-1058)
        Event emitted after the call(s):
        - AuctionFinalized(auctionedTokenId,winningBidder) (Originals99.sol#1243)
Reentrancy in Originals99.placeBid() (Originals99.sol#1215-1232):
        External calls:
        - (success) = winningBidder.call{value: currentPrice}() (Originals99.sol#1224)
        Event emitted after the call(s):
        - AuctionBidPlaced(auctionedTokenId,_msgSender(),msg.value,auctionEndTime) (Originals99.sol#1231)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Originals99.isAuctionEnd() (Originals99.sol#1182-1184) uses timestamp for comparisons
        Dangerous comparisons:
        - auctionEndTime <= block.timestamp (Originals99.sol#1183)
Originals99.placeBid() (Originals99.sol#1215-1232) uses timestamp for comparisons
```

```
        Dangerous comparisons:
        - block.timestamp + DURATION_EXTENSION_ALLOWED_BEFORE_END >= auctionEndTime (Originals99.sol#1219)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (Originals99.sol#82-92) uses assembly
        - INLINE ASM (Originals99.sol#88-90)
Address.verifyCallResult(bool,bytes,string) (Originals99.sol#251-271) uses assembly
        - INLINE ASM (Originals99.sol#263-266)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (Originals99.sol#1041-1062) uses assembly
        - INLINE ASM (Originals99.sol#1054-1056)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address.functionCall(address,bytes) (Originals99.sol#135-137) is never used and should be removed
Address.functionCall(address,bytes,string) (Originals99.sol#145-151) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (Originals99.sol#164-170) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (Originals99.sol#178-189) is never used and should be removed
Address.functionDelegateCall(address,bytes) (Originals99.sol#224-226) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (Originals99.sol#234-243) is never used and should be removed
Address.functionStaticCall(address,bytes) (Originals99.sol#197-199) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (Originals99.sol#207-216) is never used and should be removed
Address.sendValue(address,uint256) (Originals99.sol#110-115) is never used and should be removed
Address.verifyCallResult(bool,bytes,string) (Originals99.sol#251-271) is never used and should be removed
Context._msgData() (Originals99.sol#574-576) is never used and should be removed
ERC2981._deleteDefaultRoyalty() (Originals99.sol#536-538) is never used and should be removed
ERC2981._resetTokenRoyalty(uint256) (Originals99.sol#563-565) is never used and should be removed
ERC2981._setTokenRoyalty(uint256,address,uint96) (Originals99.sol#549-558) is never used and should be removed
ERC721._baseURI() (Originals99.sol#764-766) is never used and should be removed
ERC721._burn(uint256) (Originals99.sol#961-973) is never used and should be removed
ERC721Royalty._burn(uint256) (Originals99.sol#1095-1098) is never used and should be removed
Strings.toHexString(uint256) (Originals99.sol#36-47) is never used and should be removed
Strings.toHexString(uint256,uint256) (Originals99.sol#52-62) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```

```
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (Originals99.sol#110-115):
        - (success) = recipient.call{value: amount}() (Originals99.sol#113)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (Originals99.sol#178-189):
        - (success,returndata) = target.call{value: value}(data) (Originals99.sol#187)
Low level call in Address.functionStaticCall(address,bytes,string) (Originals99.sol#207-216):
        - (success,returndata) = target.staticcall(data) (Originals99.sol#214)
Low level call in Address.functionDelegateCall(address,bytes,string) (Originals99.sol#234-243):
        - (success,returndata) = target.delegatecall(data) (Originals99.sol#241)
Low level call in Originals99.placeBid() (Originals99.sol#1215-1232):
        - (success) = winningBidder.call{value: currentPrice}() (Originals99.sol#1224)
Low level call in Originals99.withdraw() (Originals99.sol#1250-1265):
        - (success1) = FUNDS_RECEIVER_50.call{value: split1}() (Originals99.sol#1257)
        - (success2) = FUNDS_RECEIVER_45.call{value: split2}() (Originals99.sol#1260)
        - (success3) = FUNDS_RECEIVER_5.call{value: address(this).balance}() (Originals99.sol#1263)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter ERC2981.royaltyInfo(uint256,uint256)._tokenId (Originals99.sol#491) is not in mixedCase
Parameter ERC2981.royaltyInfo(uint256,uint256)._salePrice (Originals99.sol#491) is not in mixedCase
Parameter ERC721.safeTransferFrom(address,address,uint256,bytes)._data (Originals99.sol#838) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable Originals99.FUNDS_RECEIVER_45 (Originals99.sol#1116) is too similar to Originals99.FUNDS_RECEIVER_50 (Originals99.
l#1115)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
```

```
INFO:Detectors:
Variable Originals99.FUNDS_RECEIVER_45 (Originals99.sol#1116) is too similar to Originals99.FUNDS_RECEIVER_50 (Originals99.
l#1115)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (Originals99.sol#655-657)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (Originals99.sol#663-666)
balanceOf(address) should be declared external:
        - ERC721.balanceOf(address) (Originals99.sol#721-724)
name() should be declared external:
        - ERC721.name() (Originals99.sol#738-740)
symbol() should be declared external:
        - ERC721.symbol() (Originals99.sol#745-747)
tokenURI(uint256) should be declared external:
        - ERC721.tokenURI(uint256) (Originals99.sol#752-757)
        - Originals99.tokenURI(uint256) (Originals99.sol#1198-1201)
approve(address,uint256) should be declared external:
        - ERC721.approve(address,uint256) (Originals99.sol#771-781)
setApprovalForAll(address,bool) should be declared external:
        - ERC721.setApprovalForAll(address,bool) (Originals99.sol#795-797)
transferFrom(address,address,uint256) should be declared external:
        - ERC721.transferFrom(address,address,uint256) (Originals99.sol#809-818)
safeTransferFrom(address,address,uint256) should be declared external:
        - ERC721.safeTransferFrom(address,address,uint256) (Originals99.sol#823-829)
getAuctionEndTime() should be declared external:
        - Originals99.getAuctionEndTime() (Originals99.sol#1150-1152)
getInitialPrice() should be declared external:
        - Originals99.getInitialPrice() (Originals99.sol#1162-1164)
getCurrentPrice() should be declared external:
        - Originals99.getCurrentPrice() (Originals99.sol#1166-1168)
getWinningBidder() should be declared external:
        - Originals99.getWinningBidder() (Originals99.sol#1170-1172)
getAuctionedTokenId() should be declared external:
        - Originals99.getAuctionedTokenId() (Originals99.sol#1174-1176)
isSold(uint256) should be declared external:
        - Originals99.isSold(uint256) (Originals99.sol#1178-1180)
getBaseURI() should be declared external:
        - Originals99.getBaseURI() (Originals99.sol#1186-1188)
setBaseURI(string) should be declared external:
        - Originals99.setBaseURI(string) (Originals99.sol#1190-1192)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Originals99.sol analyzed (15 contracts with 75 detectors), 65 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

# Solidity Static Analysis

**Original99.sol**

## Security

### Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Originals99.placeBid(): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1215:4:

### Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 1220:29:

### Low level calls:

Use of "call": should be avoided whenever possible. It can lead to unexpected behavior if return value is not handled properly. Please use Direct Calls via specifying the called contract's interface.
more
Pos: 1263:28:

## Gas & Economy

### Gas costs:

Gas requirement of function Originals99.finalizeAuction is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 1234:4:

### Gas costs:

Gas requirement of function Originals99.withdraw is infinite: If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
Pos: 1250:4:

## Miscellaneous

### Constant/View/Pure functions:

ERC721._beforeTokenTransfer(address,address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 1078:4:

### Similar variable names:

Originals99.withdraw() : Variables have very similar names "FUNDS_RECEIVER_45" and "FUNDS_RECEIVER_5". Note: Modifiers are currently not considered by this static analysis.
Pos: 1263:28:

### Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 1225:12:

### Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 1255:26:

# Solhint Linter

**Original99.sol**

```
Originals99.sol:3:1: Error: Compiler version ^0.8.0 does not satisfy
the r semver requirement
Originals99.sol:88:9: Error: Avoid using inline assembly. It is
acceptable only in rare cases
Originals99.sol:113:28: Error: Avoid using low level calls.
Originals99.sol:187:51: Error: Avoid using low level calls.
Originals99.sol:241:51: Error: Avoid using low level calls.
Originals99.sol:263:17: Error: Avoid using inline assembly. It is
acceptable only in rare cases
Originals99.sol:595:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
Originals99.sol:629:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
Originals99.sol:703:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
Originals99.sol:1054:21: Error: Avoid using inline assembly. It is
acceptable only in rare cases
Originals99.sol:1082:24: Error: Code contains empty blocks
Originals99.sol:1143:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
Originals99.sol:1183:34: Error: Avoid to make time-based decisions in
your business logic
Originals99.sol:1209:26: Error: Avoid to make time-based decisions in
your business logic
Originals99.sol:1219:13: Error: Avoid to make time-based decisions in
your business logic
Originals99.sol:1220:30: Error: Avoid to make time-based decisions in
your business logic
Originals99.sol:1224:32: Error: Avoid using low level calls.
Originals99.sol:1257:29: Error: Avoid using low level calls.
Originals99.sol:1260:29: Error: Avoid using low level calls.
Originals99.sol:1263:29: Error: Avoid using low level calls.
```

**Software analysis result:**

These software reported many false positive results and some are informational issues. So, those issues can be safely ignored.