

# Smart Contract Security Audit V1

## Yachties Smart Contract

26/10/2022



<https://saferico.com/>

[business@saferico.com](mailto:business@saferico.com)

[https://t.me/SFI\\_ANN](https://t.me/SFI_ANN)

—

# Table of Contents

## **Table of Contents**

## **Background**

## **Project Information**

NFT Information

Executive Summary

## **File and Function Level Report**

**File in Scope:**

## **Issues Checking Status**

Severity Definitions

Audit Findings

## **Automatic testing**

Testing proves

Inheritance graph

Call graph

## **Unified Modeling Language (UML)**

**Functions signature**

**Automatic general report**

## **Conclusion**

## **Disclaimer**

# Background

The purpose of the audit was to achieve the following:

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

The information in this report should be used to understand the risk exposure of the smart contract, and as a guide to improve the security posture of the smart contract by remediating the issues that were identified.

## Project Information

- **Platform:** Ethereum
- **Contract Address:** 0xc92700f76b45ea02a0611f0ac30e033919ca776f
- **Code:**

<https://github.com/Saferico/Smart-Contracts-for-Projects/blob/main/Yachtie.sol>

## NFT Information

- Name: Yachties
- Total Supply: 3333
- Holders:
- Total transactions:

Contracts address deployed to test net (Ethereum )

Yachties smart contract on ETH test net to test every function by the auditor.

<https://goerli.etherscan.io/address/0xc92700f76b45ea02a0611f0ac30e033919ca776f>

## Executive Summary

According to our assessment, the customer`s solidity smart contract is **“WELL SECURED”**. The team has fixed the high issue and the low-level issues.

Well Secured	✓
Secured	
Poor Secured	
Insecure	

Automated checks are with remix IDE. All issues were performed by the team, which included the analysis of code functionality, manual audit found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the audit overview section. The general overview is presented in the Project Information section and all issues found are located in the audit overview section.

Team found 0 critical, 0 high, 0 medium, 3 low, 0 very low-level issues and 1 note in all solidity files of the contract

The files:

Yachties.sol

# File and Function Level Report

## File in Scope:

Contract Name	SHA 256 hash	Contract Address
Yachties.sol	ed3180920a790242c8af0c1fc9d575d62b636dd3823217d80110f8de2c0e83bd	0xc92700f76b45ea02a0611f0ac30e033919ca776f

- Contract: Yachties
- Inherit: ERC721, ERC721URISStorage, Ownable
- Observation: All passed including security check
- Test Report: passed
- Score: passed
- Conclusion: passed

Function	Test Result	Type / Return Type	Score
name	✓	Read / public	<b>Passed</b>
symbol	✓	Read / public	<b>Passed</b>
getmintstatus	✓	Read / public	<b>Passed</b>
supportsInterface	✓	Read / public	<b>Passed</b>
getCost	✓	Read / public	<b>Passed</b>
balanceOf	✓	Read / public	<b>Passed</b>
Owner	✓	Read / public	<b>Passed</b>
getPhase	✓	Read / public	<b>Passed</b>
getBaseURI	✓	Read / public	<b>Passed</b>
getApprovedForAll	✓	Read / public	<b>Passed</b>
totalSupply	✓	Read / public	<b>Passed</b>
getApproved	✓	Read / public	<b>Passed</b>
ownerOf	✓	Read / public	<b>Passed</b>

gettokenURI	✓	Read / public	<b>Passed</b>
tokenURI	✓	Read / public	<b>Passed</b>
withdraw	✓	Write / public	<b>Passed</b>
mint	✓	Write / payable	<b>Passed</b>
approve	✓	Write / public	<b>Passed</b>
safeTransferFrom	✓	Write / public	<b>Passed</b>
safeTransferFrom	✓	Write / public	<b>Passed</b>
setPhase	✓	Write / public	<b>Passed</b>
openMint	✓	Write / public	<b>Passed</b>
closeMint	✓	Write / public	<b>Passed</b>
transferOwnership	✓	Write / public	<b>Passed</b>
setApprovalForAll	✓	Write / public	<b>Passed</b>
transferFrom	✓	Write / public	<b>Passed</b>
mint_batch	✓	Write / public	<b>Passed</b>
setCost	✓	Write / public	<b>Passed</b>
renounceOwnership	✓	Write / public	<b>Passed</b>
setBaseURI	✓	Write / public	<b>Passed</b>

## Issues Checking Status

No.	Issue Description	Checking Status
1	Compiler warnings.	Passed
2	Race conditions and Reentrancy. Cross-function race conditions.	Passed
3	Possible delays in data delivery.	Passed
4	Oracle calls.	Passed
5	Design Logic.	Passed
6	Timestamp dependence.	Passed
7	Integer Overflow and Underflow.	Passed
8	DoS with Revert.	Passed
9	DoS with block gas limit.	Passed with Notes
10	Methods execution permissions.	Passed
11	Economy model. If application logic is based on an incorrect economic model, the application would not function correctly and participants would incur financial losses. This type of issue is most often found in bonus rewards systems, Staking and Farming contracts, Vault and Vesting contracts, etc.	Passed
12	The impact of the exchange rate on the logic.	Passed
13	Private user data leaks.	Passed
14	Malicious Event log.	Passed
15	Scoping and Declarations.	Passed
16	Uninitialized storage pointers.	Passed
17	Arithmetic accuracy.	Passed

## Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to tokens 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 functions
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
Note	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored.



# Audit Findings

## Critical:

No Critical severity vulnerabilities were found.

## High:

No High severity vulnerabilities were found.

## Medium:

No Medium severity vulnerabilities were found

## Low:

### #Missing zero address validation

#### Description

When the minter wants to mint to address by using the main mint function, he has to check for the zero address to make he didn't add the zero address. Otherwise, he will lose his NFT and his ETH funds so it is security recommended to check for zero address.

```
function mint(address to, uint256 num) public payable {
    require(mintopen == 1, "Mint is not live");
    if(phase == 0)
    {
        cost=0.015 ether;
        require(msg.value >= cost*num , "Not enough ETH sent; Please check the amount!");
        for (uint i=0; i < num; i++){
            uint256 tokenId = _tokenIdCounter.current();
            require(tokenId < 3332, "The collection is sold out");
            _tokenIdCounter.increment();
            _safeMint(to, tokenId);
            _setTokenURI(tokenId, string(abi.encodePacked(tokenBaseURI, string(abi.encodePacked(Strings.toString(tokenId), ".json"))))); } }
        else
        require(msg.value >= cost*num , "Not enough ETH sent; Please check the amount!");
        for (uint i=0; i < num; i++)
        {
            uint256 tokenId = _tokenIdCounter.current();
            require(tokenId < 3332, "The collection is sold out");
            _tokenIdCounter.increment();
            _safeMint(to, tokenId);
            _setTokenURI(tokenId, string(abi.encodePacked(tokenBaseURI, string(abi.encodePacked(Strings.toString(tokenId), ".json"))))); } }
```

## Remediation

Use the require statement to check for zero addresses.

Status: **Closed**. Fixed in version 2.

## #Pragam version not fixed

### Description

It is a good practice to lock the solidity version for a live deployment (use 0.8.17 instead of ^0.8.4). contracts should be deployed with the same compiler version and flags that they have been tested the most with. Locking the pragma helps ensure that contracts do not accidentally get deployed using, for example, the latest compiler which may have higher risks of undiscovered bugs. Contracts may also be deployed by others and the pragma indicates the compiler version intended by the original authors.

### Remediation

Remove the ^ sign to lock the pragma version.

Status: **Closed**. Fixed in version 2.

## #Owner privileges (In the period when the owner isn't renounced)

### Description

The owner can change the price in public mint.

The owner can open and close the mint.

```
function openMint() public onlyOwner {
    mintopen=1;
}
function closeMint() public onlyOwner {
    mintopen=0;
}

function setCost(uint256 value) public onlyOwner {
    cost=value;
}
```

### Remediation

Make these functions internal in next version or the team should announce the investors before doing anything in the contract to give them time if they want to do anything.

P.S: This issue is common to the majority of NFT smart contracts.

Status: **Acknowledged**.

## Very Low:

No Very Low severity vulnerabilities were found.

### Notes:

#### #Unnecessary import of ERC721 library

##### Description

The main contract inherits: ERC721, ERC721URIStorage, Ownable. ERC721URIStorage which is already import ERC721 library, so no need to import it again in the main contract.

##### Remediation

Remove unnecessary library from the main contract save some gas fees.

Status: **Closed**. Fixed in version2.

# Automatic Testing

## 1- Check for security

ed3180920a790242c8af0c1fc9d575d62b636dd3823217d80110f8de2c0e83...

File: Yachtie.sol | Language: solidity | Size: 4901 bytes | Date: 2022-10-26T04:46:11.694Z

Critical	High	Medium	Low	Note
0	0	0	0	0



## 2- SOLIDITY STATIC ANALYSIS

### SOLIDITY STATIC ANALYSIS

☒ Select all ☒ Autorun Run

**Security**

☒ Select Security

- ☒ **Transaction origin:**  
'tx.origin' used
- ☒ **Check-effects-interaction:**  
Potential reentrancy bugs
- ☒ **Inline assembly:**  
Inline assembly used
- ☒ **Block timestamp:**  
Can be influenced by miners
- ☒ **Low level calls:**  
Should only be used by experienced devs
- ☒ **Block hash:**  
Can be influenced by miners
- ☒ **Selfdestruct:**  
Contracts using destructed contract can be broken

**Gas & Economy**

☒ Select Gas & Economy

- ☒ **Gas costs:**  
Too high gas requirement of functions
- ☒ **This on local calls:**  
Invocation of local functions via 'this'
- ☒ **Delete dynamic array:**  
Use require/assert to ensure complete deletion
- ☒ **For loop over dynamic array:**  
Iterations depend on dynamic array's size
- ☒ **Ether transfer in loop:**  
Transferring Ether in a for/while/do-while loop

### SOLIDITY STATIC ANALYSIS

**ERC**

☒ Select ERC

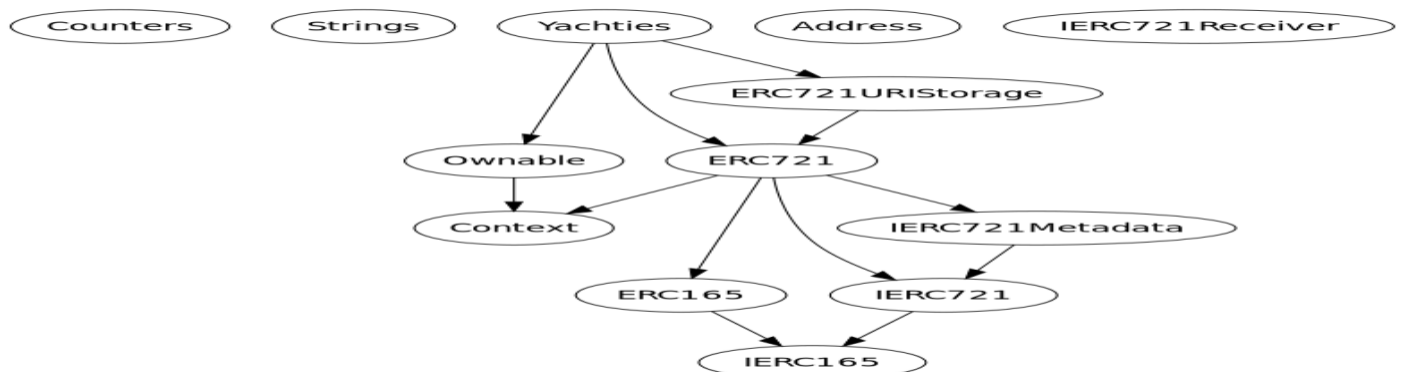
- ☒ **ERC20:**  
'decimals' should be 'uint8'

**Miscellaneous**

☒ Select Miscellaneous

- ☒ **Constant/View/Pure functions:**  
Potentially constant/view/pure functions
- ☒ **Similar variable names:**  
Variable names are too similar
- ☒ **No return:**  
Function with 'returns' not returning
- ☒ **Guard conditions:**  
Ensure appropriate use of require/assert
- ☒ **Result not used:**  
The result of an operation not used
- ☒ **String length:**  
Bytes length != String length
- ☒ **Delete from dynamic array:**  
'delete' leaves a gap in array
- ☒ **Data truncated:**  
Division on int/uint values truncates the result

## 3- Inheritance graph



## 4- SOLIDITY UNIT TESTING

### SOLIDITY UNIT TESTING ✓ >

Test your smart contract in Solidity.

Select directory to load and generate test files.

Test directory:

☒ Select all

☒ tests/Yachtie\_test.sol

Progress: 1 finished (of 1)

**PASS** testSuite (tests/Yachtie\_test.sol)

✓ Before all

✖

✓ Check success

✖

✓ Check success2

✖

✓ Check failure

✖

✓ Check sender and value

✖

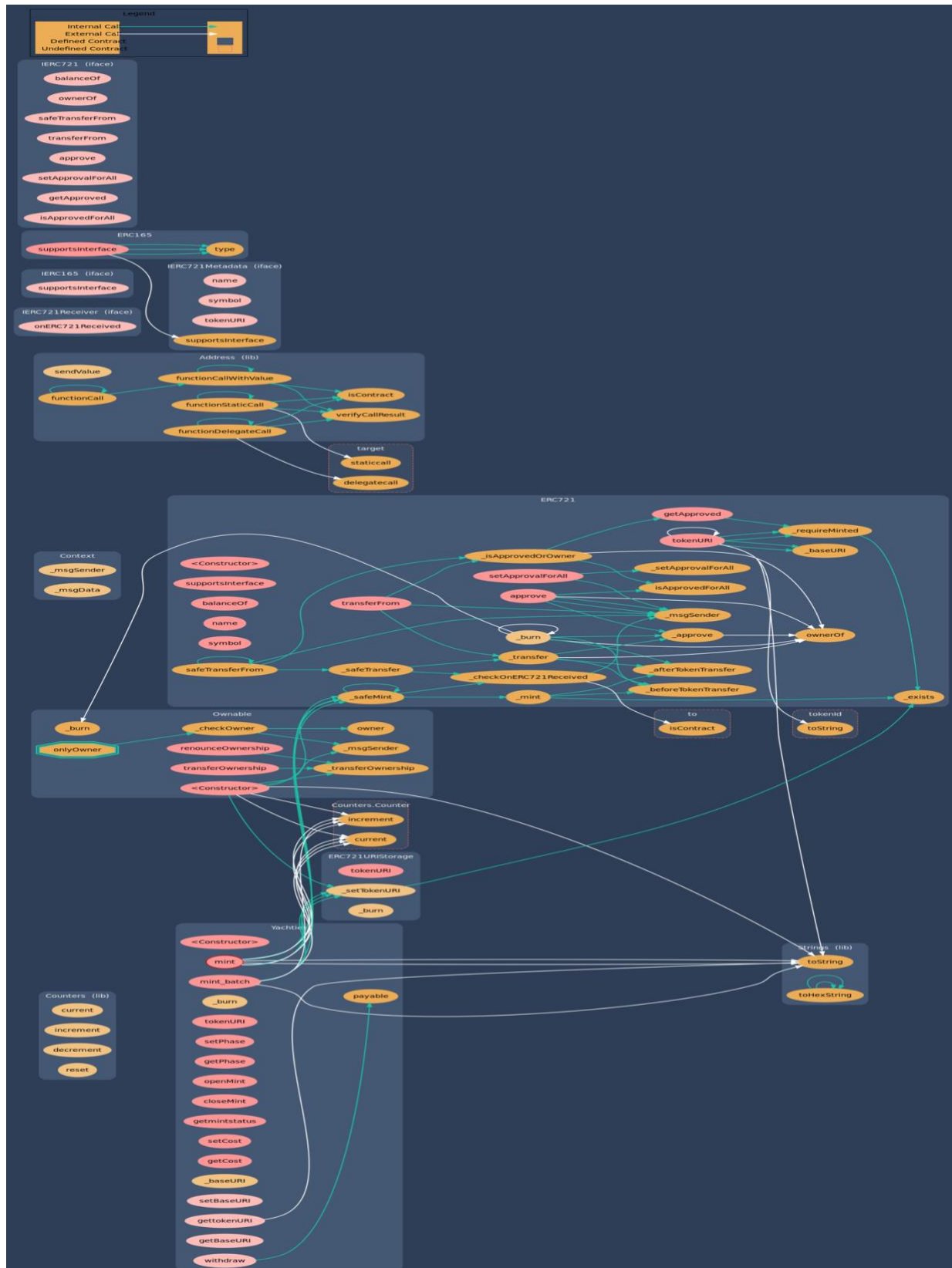
**Result for tests/Yachtie\_test.sol**

Passed: 5

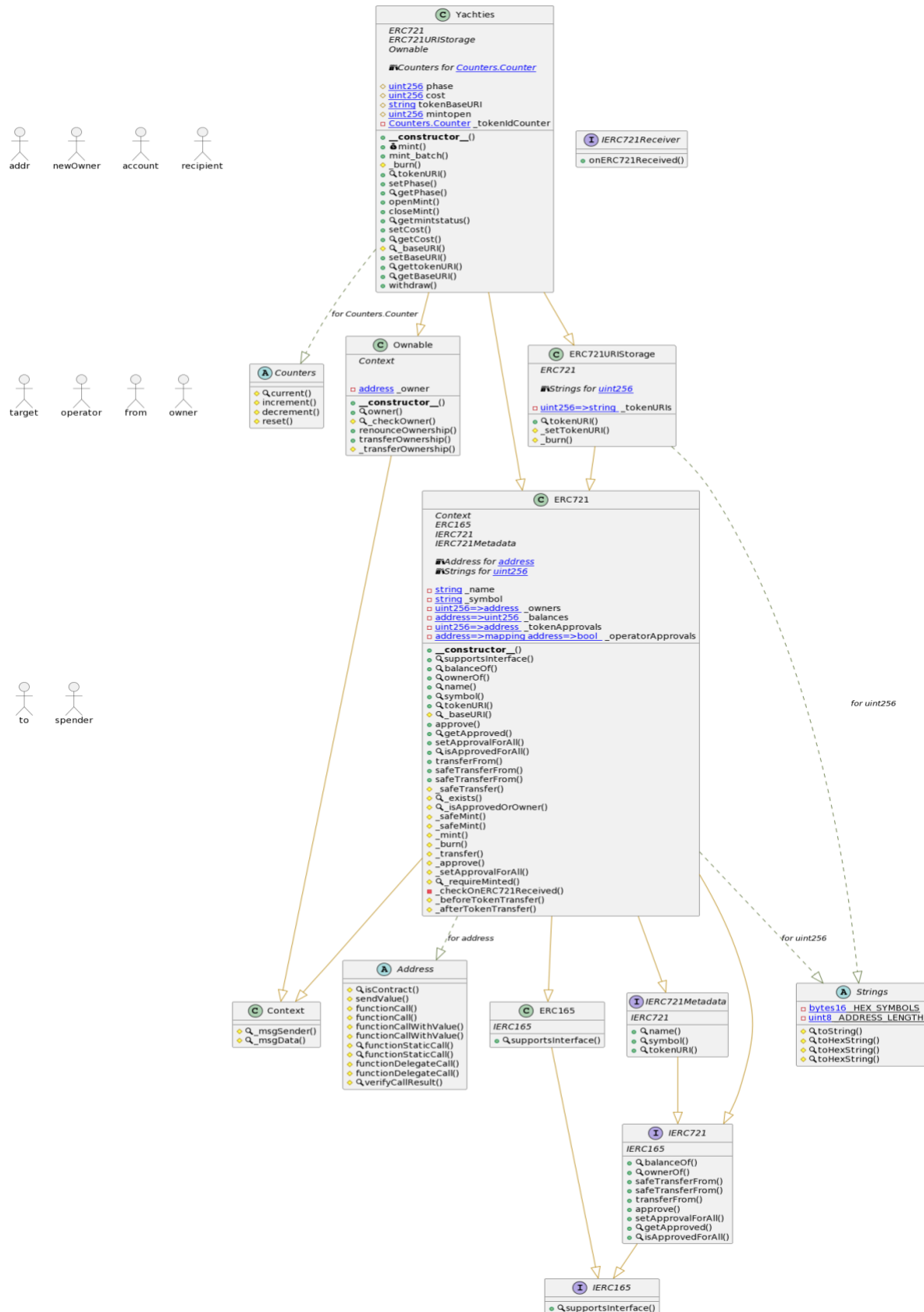
Failed: 0

Time Taken: 0.40s

## 5- Call graph



# Unified Modeling Language (UML)



## Functions signature

Sighash		Function Signature
=====		
16279055	=>	isContract (address)
ad04a8d1	=>	current (Counter)
e2bee435	=>	increment (Counter)
854ec98e	=>	decrement (Counter)
440d212a	=>	reset (Counter)
6900a3ae	=>	toString (uint256)
8fba8d5c	=>	toHexString (uint256)
63e1cbea	=>	toHexString (uint256, uint256)
1bb0c665	=>	toHexString (address)
119df25f	=>	_msgSender ()
8b49d47e	=>	_msgData ()
8da5cb5b	=>	owner ()
53a72975	=>	_checkOwner ()
715018a6	=>	renounceOwnership ()
f2fde38b	=>	transferOwnership (address)
d29d44ee	=>	_transferOwnership (address)
24a084df	=>	sendValue (address, uint256)
a0b5ffb0	=>	functionCall (address, bytes)
241b5886	=>	functionCall (address, bytes, string)
2a011594	=>	functionCallWithValue (address, bytes, uint256)
d525ab8a	=>	functionCallWithValue (address, bytes, uint256, string)
c21d36f3	=>	functionStaticCall (address, bytes)
dbc40fb9	=>	functionStaticCall (address, bytes, string)
ee33b7e2	=>	functionDelegateCall (address, bytes)
57387df0	=>	functionDelegateCall (address, bytes, string)
946b5793	=>	verifyCallResult (bool, bytes, string)
150b7a02	=>	onERC721Received (address, address, uint256, bytes)
01ffc9a7	=>	supportsInterface (bytes4)
70a08231	=>	balanceOf (address)
6352211e	=>	ownerOf (uint256)
b88d4fde	=>	safeTransferFrom (address, address, uint256, bytes)
42842e0e	=>	safeTransferFrom (address, address, uint256)
23b872dd	=>	transferFrom (address, address, uint256)
095ea7b3	=>	approve (address, uint256)
a22cb465	=>	setApprovalForAll (address, bool)
081812fc	=>	getApproved (uint256)
e985e9c5	=>	isApprovedForAll (address, address)
06fdde03	=>	name ()
95d89b41	=>	symbol ()
c87b56dd	=>	tokenURI (uint256)
743976a0	=>	_baseURI ()
24b6b8c0	=>	_safeTransfer (address, address, uint256, bytes)
f8e76cc0	=>	_exists (uint256)
4cdc9549	=>	_isApprovedOrOwner (address, uint256)
b3e1c718	=>	_safeMint (address, uint256)
6a4f832b	=>	_safeMint (address, uint256, bytes)
4e6ec247	=>	_mint (address, uint256)
9b1f9e74	=>	_burn (uint256)
30e0789e	=>	_transfer (address, address, uint256)
7b7d7225	=>	_approve (address, uint256)
8c4e3f32	=>	_setApprovalForAll (address, address, bool)



```
a0aea85d => _requireMinted(uint256)
1fd01de1 => _checkOnERC721Received(address,address,uint256,bytes)
cad3be83 => _beforeTokenTransfer(address,address,uint256)
8f811a1c => _afterTokenTransfer(address,address,uint256)
01538868 => _setTokenURI(uint256,string)
40c10f19 => mint(address,uint256)
c8704c59 => mint_batch()
2cc82655 => setPhase(uint256)
eced0280 => getPhase()
bce6d672 => openMint()
64f101f0 => closeMint()
b29f7e8e => getmintstatus()
44a0d68a => setCost(uint256)
bd3e19d4 => getCost()
55f804b3 => setBaseURI(string)
f3f0cd7f => gettokenURI(uint256)
714c5398 => getBaseURI()
3ccfd60b => withdraw()
```

# Automatic general report


## Files Description Table

File Name	SHA-1 Hash
/Users/macbook/Desktop/smart contracts/Yachtie.sol	7aa5209a4f93b7550b52aba5480dcdb0e0e52b08

## Contracts Description Table

Contract	Type	Bases		
:-----: :-----: :-----: :-----: :-----:				
L	**Function Name**	**Visibility**	**Mutability**	
**Modifiers**				
**Counters**   Library				
L   current	Internal			
L   increment	Internal			
L   decrement	Internal			
L   reset	Internal			
**Strings**   Library				
L   toString	Internal			
L   toHexString	Internal			
L   toHexString	Internal			
L   toHexString	Internal			
**Context**   Implementation				
L   _msgSender	Internal			
L   _msgData	Internal			
**Ownable**   Implementation   Context				
L   <Constructor>	Public	!		NO!
L   owner	Public	!	NO!	
L   _checkOwner	Internal			
L   renounceOwnership	Public	!		onlyOwner
L   transferOwnership	Public	!		onlyOwner
L   _transferOwnership	Internal			
**Address**   Library				
L   isContract	Internal			
L   sendValue	Internal			
L   functionCall	Internal			
L   functionCall	Internal			
L   functionCallWithValue	Internal			
L   functionCallWithValue	Internal			
L   functionStaticCall	Internal			
L   functionStaticCall	Internal			
L   functionDelegateCall	Internal			
L   functionDelegateCall	Internal			
L   verifyCallResult	Internal			

```

| | | | | | | |
| **IERC721Receiver** | Interface | | |
| L | onERC721Received | External | ! |  | NO |
| | | |
| **IERC165** | Interface | | |
| L | supportsInterface | External | ! | | NO |
| | | |
| **ERC165** | Implementation | IERC165 | | |
| L | supportsInterface | Public | ! | | NO |
| | | |
| **IERC721** | Interface | IERC165 | | |
| L | balanceOf | External | ! | | NO |
| L | ownerOf | External | ! | | NO |
| L | safeTransferFrom | External | ! |  | NO |
| L | safeTransferFrom | External | ! |  | NO |
| L | transferFrom | External | ! |  | NO |
| L | approve | External | ! |  | NO |
| L | setApprovalForAll | External | ! |  | NO |
| L | getApproved | External | ! | | NO |
| L | isApprovedForAll | External | ! | | NO |
| | | |
| **IERC721Metadata** | Interface | IERC721 | | |
| L | name | External | ! | | NO |
| L | symbol | External | ! | | NO |
| L | tokenURI | External | ! | | NO |
| | | |
| **ERC721** | Implementation | Context, ERC165, IERC721, IERC721Metadata | | |
| L | <Constructor> | Public | ! |  | NO |
| L | supportsInterface | Public | ! | | NO |
| L | balanceOf | Public | ! | | NO |
| L | ownerOf | Public | ! | | NO |
| L | name | Public | ! | | NO |
| L | symbol | Public | ! | | NO |
| L | tokenURI | Public | ! | | NO |
| L | _baseURI | Internal |  | | |
| L | approve | Public | ! |  | NO |
| L | getApproved | Public | ! | | NO |
| L | setApprovalForAll | Public | ! |  | NO |
| L | isApprovedForAll | Public | ! | | NO |
| L | transferFrom | Public | ! |  | NO |
| L | safeTransferFrom | Public | ! |  | NO |
| L | safeTransferFrom | Public | ! |  | NO |
| L | _safeTransfer | Internal |  | | |
| L | _exists | Internal |  | | |
| L | _isApprovedOrOwner | Internal |  | | |
| L | _safeMint | Internal |  | | |
| L | _safeMint | Internal |  | | |
| L | _mint | Internal |  | | |
| L | _burn | Internal |  | | |
| L | _transfer | Internal |  | | |
| L | _approve | Internal |  | | |
| L | _setApprovalForAll | Internal |  | | |
| L | _requireMinted | Internal |  | | |
| L | _checkOnERC721Received | Private |  |  | | |
| L | _beforeTokenTransfer | Internal |  |  | | |

```

```

| L | _afterTokenTransfer | Internal 🔒 | 🔒 | |
| | | |
| **ERC721URIStorage** | Implementation | ERC721 | | |
| L | tokenURI | Public ! | NO! |
| L | _setTokenURI | Internal 🔒 | 🔒 | |
| L | _burn | Internal 🔒 | 🔒 | |
| | | |
| **Yachties** | Implementation | ERC721, ERC721URIStorage, Ownable | | |
| L | <Constructor> | Public ! | 🔒 | ERC721 |
| L | mint | Public ! | 💰 NO! |
| L | mint_batch | Public ! | 🔒 | onlyOwner |
| L | _burn | Internal 🔒 | 🔒 | |
| L | tokenURI | Public ! | NO! |
| L | setPhase | Public ! | 🔒 | onlyOwner |
| L | getPhase | Public ! | | onlyOwner |
| L | openMint | Public ! | 🔒 | onlyOwner |
| L | closeMint | Public ! | 🔒 | onlyOwner |
| L | getmintstatus | Public ! | | onlyOwner |
| L | setCost | Public ! | 🔒 | onlyOwner |
| L | getCost | Public ! | | onlyOwner |
| L | _baseURI | Internal 🔒 | | |
| L | setBaseURI | External ! | 🔒 | onlyOwner |
| L | getTokenURI | External ! | | onlyOwner |
| L | getBaseURI | External ! | | onlyOwner |
| L | withdraw | External ! | 🔒 | onlyOwner |

```

#### Legend

Symbol	Meaning
⚙️	Function can modify state
💰	Function is payable

## Conclusion

The contracts are written systematically. Team found no critical issues. So, it is good to go for production.

Since possible test cases can be unlimited and developer level documentation (code flow diagram with function level description) not provided, for such an extensive smart contract 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.

Security state of the reviewed contract is “ Well Secured”.

- ✓ No volatile code.
- ✓ No high severity issues were found.
- ✓ Low (or very low) level issues have been fixed.

# Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as of the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against the team on the basis of what it says or doesn't say, or how team produced it, and it is important for you to conduct your own independent investigations before making any decisions. team go into more detail on this in the below disclaimer below – please make sure to read it in full.

By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and Saferico and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (Saferico s) owe no duty of care towards you or any other person, nor does Saferico make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and Saferico hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, Saferico hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Saferico, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report. The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.