Smart Contract
Security Audit
V2

Platzee NFT Smart Contract

11/3/2022

https://saferico.com/

business@saferico.com
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**: 0xa4bf1b4f57B14F28b997fF47e0DF73a1a1Ee2745
- **Code:**

https://rinkeby.etherscan.io/address/0xa4bf1b4f57b14f28b997ff47e0df73a1a1ee2745#code

NFT Information

- Name: PLATZEEGT3
- Total Supply: 100
- Holders:
- Total transactions:

Contracts address deployed to test net (ETH)

PLATZEEGT3 Smart contract on ETH test net to test write functions by the auditor.

https://rinkeby.etherscan.io/address/0xa4bf1b4f57b14f28b997ff47e0df73a1a1ee2745

Executive Summary

According to our assessment, the customer`s solidity smart contract is **Insecured**.

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 1 critical, 0 high, 0 medium, 2 low, 0 very low-level issues and 0 note in all solidity files of the contract

The files:

PlatzeeNFT.sol

File and Function Level Report

File in Scope:

Contract Name	SHA 256 hash	Contract Address
PlatzeeNFT.sol	**0853cede6a2cefa6f373911e46fe4ea7ecf0c8aed213352f0581ff37a06866d8**	0xa4bf1b4f57B14F28b997fF47e0DF73a1a1Ee2745

- Contract: PlatzeeNFT
- Inherit: ERC721, ERC721Enumerable, ERC721URIStorage,Ownable
- Observation: All passed including security check
- Test Report: passed
- Score: passed
- Conclusion: passed

Function	Test Result	Type / Return Type	Score
name	✓	Read / public	**Passed**
symbol	✓	Read / public	**Passed**
tokenOfOwnerByIndex	✓	Read / public	**Passed**
supportsInterface	✓	Read / public	**Passed**
ownerOf	✓	Read / public	**Passed**
balanceOf	✓	Read / public	**Passed**
totalSupply	✓	Read / public	**Passed**
getTokenIds	✓	Read / public	**Passed**
baseURI	✓	Read / public	**Passed**
getApprovedForAll	✓	Read / public	**Passed**
tokenByIndex	✓	Read / public	**Passed**
getApproved	✓	Read / public	**Passed**

tokenURI	✓	Read / public	**Passed**
getBalance	✓	Read / public	**Passed**
mint_to_address	✓	Write / payable	**Passed**
approve	✓	Write / public	**Passed**
safeTransferFrom	✓	Write / public	**Passed**
safeTransferFrom	✓	Write / public	**Passed**
transferFrom	✓	Write / public	**Passed**
setMintable	✓	Write / public	**Passed**
mint	✓	Write / payable	**Passed**
setBaseURI	✓	Write / public	**Passed**
mintAll	X	Write / public	**Not Passed**
setApprovalForAll	✓	Write / public	**Passed**
withdraw	✓	Write / payable	**Passed**
setCurrentPrice	✓	Write / public	**Passed**
transferOwnership	✓	Write / public	**Passed**
renounceOwnership	✓	Write / public	**Passed**

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 with Notes**
7	Integer Overflow and Underflow.	**Passed**
8	DoS with Revert.	**Passed**
9	DoS with block gas limit.	**Passed**
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:

#There aren't any limits per address

Description

When the auditor tested the mint function found one address can mint the total supply (100) for 100 times, every time can mint 1 NFT/ Transaction.

```
function mint() public isMintable {

        uint256 newItemId = _tokenIds.current();
        _safeMint(msg.sender, newItemId);
        _tokenIds.increment();
        totalSupplyRemaining--;
        console.log("An NFT w/ ID %s has been minted to %s", newItemId,
msg.sender);
        emit minted(msg.sender, newItemId);
    }
```

Remediation

The developer should add limits per address like 3 per address. Like this statement.

```
require(_mintAmount <= 3, "Mint limit exceeded." );
```

Status: Closed. Fixed in version2.

#New issues on version 2.

#Any address can mint without sending the payment

Description

When the auditor tested the mint function found any address can mint up to 3 NFT without sending the payments to the contract address. That because the developer add require statement to check if the price is equal or less than the current price .

```
function mint() public payable isMintable limitByAddress{
        require(msg.sender != address(0), "PLATZEE: mint to the zero address");
        require(msg.value <= currentPrice  ,"PLATZEE: Insufficient funds.");
        uint256 newItemId = _tokenIds.current();
        _safeMint(msg.sender, newItemId);
        _tokenIds.increment();
        totalSupplyRemaining--;
```

```
console.log("An NFT w/ ID %s has been minted to %s", newItemId, msg.sender);
        emit minted(msg.sender, newItemId);
    }
```

Remediation

The developer should fix the require statement and make it equal or higher than the current price.

```
require(msg.value >= currentPrice  ,"PLATZEE: Insufficient
funds.");
```

Status: Opened.

High:

#There isn't any transferOwnership function or renounceOwnership function

Description

The developer didn't add transfer the ownership to a new address and make it as the controller, which mean the developer is the only controller in this project and the same for renounceOwnership function.

Remediation

The developer has to add ownable library to the contract and make platzeeNFT contract inherit ERC721, ERC721Enumerable, ERC721URIStorage, Ownable.

```
import "@openzeppelin/contracts/access/Ownable.sol";
```

Status: Closed. Fixed in version2.

Medium:

No Medium severity vulnerabilities were found

Low:

#Pragam version not fixed

Description

It is a good practice to lock the solidity version for a live deployment (use 0.8.2 instead of ^0.8.0). 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: Open.

mint functions aren't payable

Description

The developer adds mint, mintAll, mint_to_address as public write functions not public payable function, this maybe lead to a vulnerability or make unauthorized or unintended state changes.

P.S: There isn't any price for the NFT and there isn't any withdraw function.

```
function mintAll() public isMintable onlyOwner {
        for (uint256 index = 0; index < MAX_SUPPLY; index++) {
            uint256 newItemId = _tokenIds.current();
            _safeMint(msg.sender, newItemId);
            _tokenIds.increment();
            totalSupplyRemaining--;
            console.log("An NFT w/ ID %s has been minted to %s", newItemId,
msg.sender);
        }
        emit minted(msg.sender, 1);
    }
function mint_to_address(address addressToMint, uint256 quantity) public isMintable
isNotExceedAvailableSupplyByAmount(quantity) onlyOwner{
        // Get the current tokenId, this starts at 0.

        for (uint256 index = 0; index < quantity; index++) {
            uint256 newItemId = _tokenIds.current();
            _safeMint(addressToMint, newItemId);
            _tokenIds.increment();
            totalSupplyRemaining--;
            console.log("An NFT w/ ID %s has been minted to %s", newItemId,
addressToMint);
        }
    }
```

Remediation

The team should make it public and payable, and there any price for NFT should add and add withdraw function to transfer the funds to the owner or any address.

Status: Closed. Fixed in version2.

#Missing zero address validation

Description

The developer missing to check the zero address in the most the code , it should be checked for zero address. Otherwise, they may lose the ability to use the privileged functions.

Remediation
Use the require statement to check for zero addresses.

```
require(account != address(0), "ERC20: mint to the zero address");
```

Status: Closed. Fixed in version2.

#Unnecessary import Strings library

Description
The developer import Strings library in the main contract and no need for that because it already imported in ERC721 contract so it useless import just costing more ETH gas.

```
import "@openzeppelin/contracts/utils/Strings.sol";
```

Remediation
Remove Strings Library to save ETH gas fees.

Status: Closed. Fixed in version2.

#MintAll function doesn't work

Description
When the auditor tested the mintAll function always failed.

```
function mintAll() public isMintable onlyOwner {
        require(msg.sender != address(0), "PLATZEE: mint to the zero address");
        for (uint256 index = 0; index < MAX_SUPPLY; index++) {
            uint256 newItemId = _tokenIds.current();
            _safeMint(msg.sender, newItemId);
            _tokenIds.increment();
            totalSupplyRemaining--;
            console.log("An NFT w/ ID %s has been minted to %s", newItemId,
msg.sender);
        emit minted(msg.sender, 1);}
```

Remediation
Check and test the function again to see what is missing, and try to add some comments for the auditor to test it.

Status: Open.

Very Low:

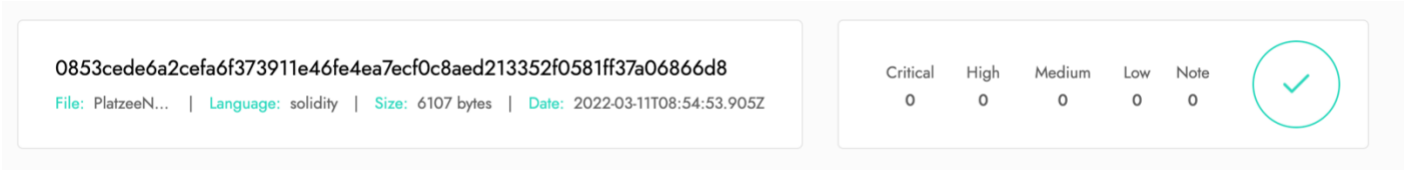
No Very Low severity vulnerabilities were found.

Notes:

No Notes were found.

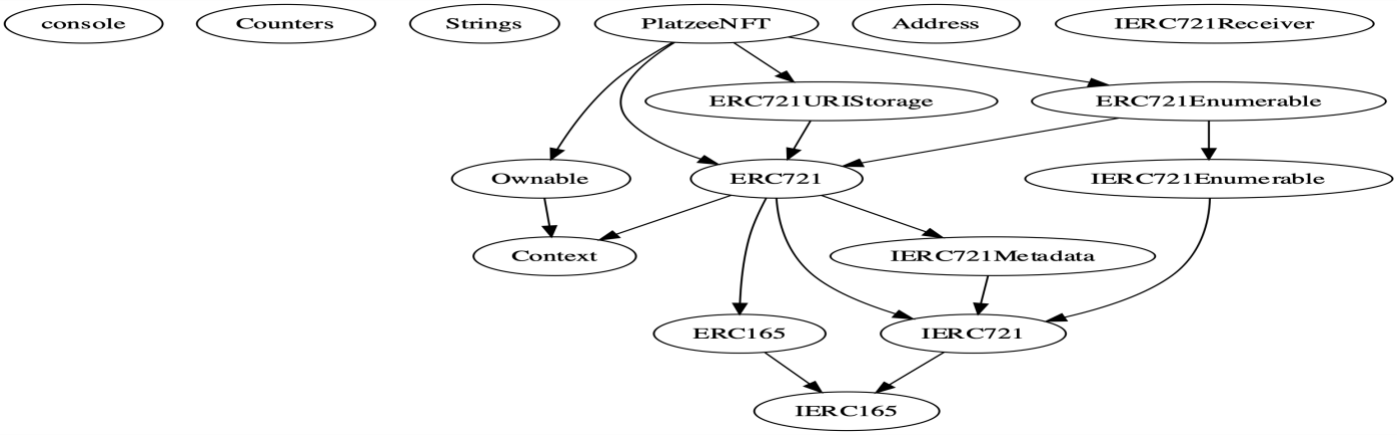
Automatic Testing

1- Check for security

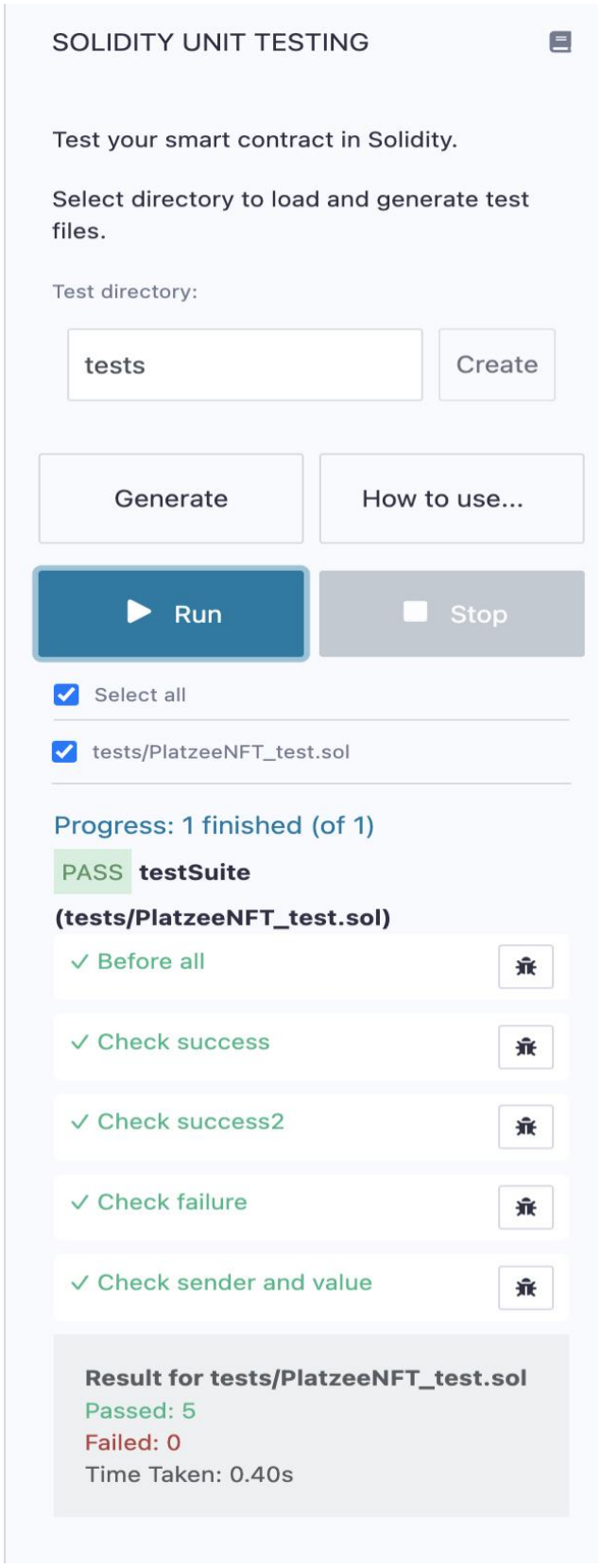


2- SOLIDITY STATIC ANALYSIS

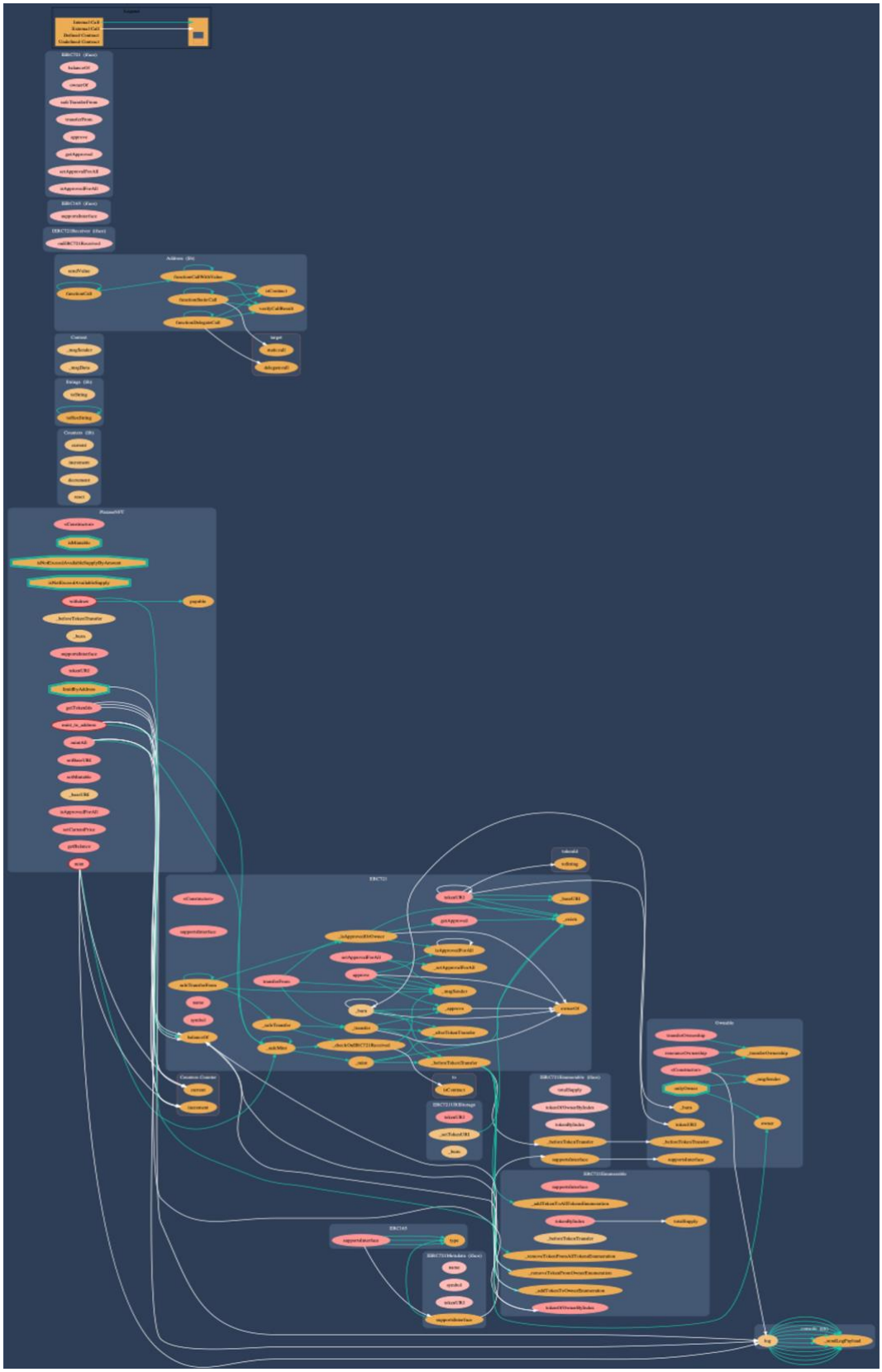
3- Inheritance graph



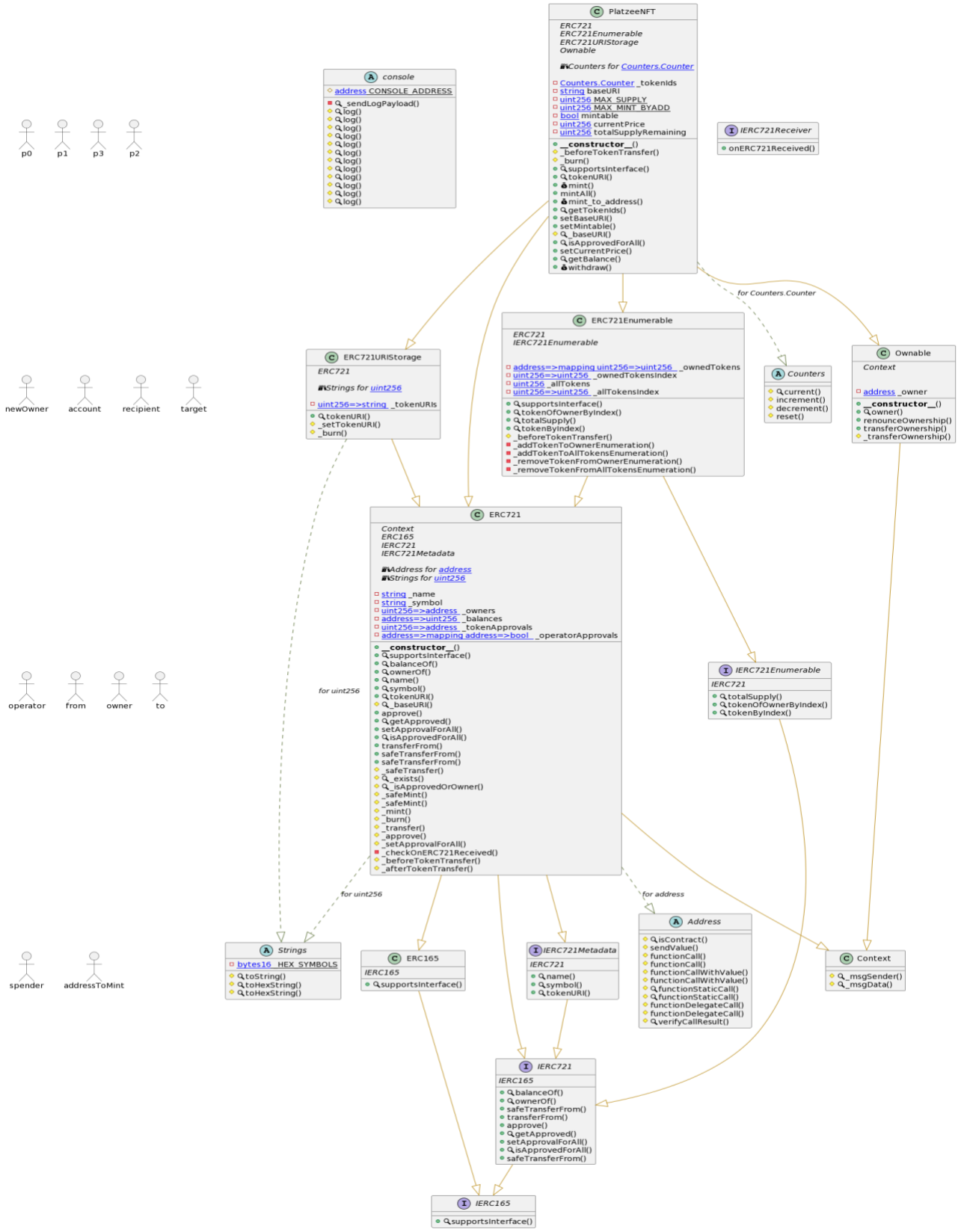
4- SOLIDITY UNIT TESTING



5- Call graph



Unified Modeling Language (UML)



Functions signature

```
 Sighash   |   Function Signature
========================
16279055  =>  isContract(address)
47ee4fe3  =>  _sendLogPayload(bytes)
ef1cefe7  =>  log(address,address,string,uint256)
21bdaf25  =>  log(address,address,string,string)
6f1a594e  =>  log(address,address,string,bool)
8f736d16  =>  log(address,address,string,address)
3971e78c  =>  log(address,address,bool,uint256)
aa6540c8  =>  log(address,address,bool,string)
2cd4134a  =>  log(address,address,bool,bool)
9f1bc36e  =>  log(address,address,bool,address)
94250d77  =>  log(address,address,address,uint256)
f808da20  =>  log(address,address,address,string)
0e378994  =>  log(address,address,address,bool)
665bf134  =>  log(address,address,address,address)
ad04a8d1  =>  current(Counter)
e2bee435  =>  increment(Counter)
854ec98e  =>  decrement(Counter)
440d212a  =>  reset(Counter)
6900a3ae  =>  toString(uint256)
8fba8d5c  =>  toHexString(uint256)
63e1cbea  =>  toHexString(uint256,uint256)
119df25f  =>  _msgSender()
8b49d47e  =>  _msgData()
8da5cb5b  =>  owner()
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)
42842e0e  =>  safeTransferFrom(address,address,uint256)
23b872dd  =>  transferFrom(address,address,uint256)
095ea7b3  =>  approve(address,uint256)
081812fc  =>  getApproved(uint256)
a22cb465  =>  setApprovalForAll(address,bool)
e985e9c5  =>  isApprovedForAll(address,address)
b88d4fde  =>  safeTransferFrom(address,address,uint256,bytes)
18160ddd  =>  totalSupply()
2f745c59  =>  tokenOfOwnerByIndex(address,uint256)
4f6ccce7  =>  tokenByIndex(uint256)
```

```
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)
1fd01de1  =>  _checkOnERC721Received(address,address,uint256,bytes)
cad3be83  =>  _beforeTokenTransfer(address,address,uint256)
8f811a1c  =>  _afterTokenTransfer(address,address,uint256)
69025b5f  =>  _addTokenToOwnerEnumeration(address,uint256)
e03d890b  =>  _addTokenToAllTokensEnumeration(uint256)
68df0d53  =>  _removeTokenFromOwnerEnumeration(address,uint256)
4cbb4a0a  =>  _removeTokenFromAllTokensEnumeration(uint256)
01538868  =>  _setTokenURI(uint256,string)
1249c58b  =>  mint()
595882b3  =>  mintAll()
0eb2df91  =>  mint_to_address(address,uint256)
67f718a9  =>  getTokenIds()
55f804b3  =>  setBaseURI(string)
285d70d4  =>  setMintable(bool)
18b20071  =>  setCurrentPrice(uint256)
12065fe0  =>  getBalance()
3ccfd60b  =>  withdraw()
```

Automatic general report

```
Files Description Table
|  File Name  |  SHA-1 Hash  |
|-------------|--------------|
| /Users/macbook/Desktop/smart contracts/PlatzeeNFT.sol |
df22b048ba844b0aac57f0be08b9c070aeccf67a |
 Contracts Description Table
|  Contract  |         Type        |       Bases      |                  |
|
|:----------:|:-------------------:|:----------------:|:----------------:|:--------
-------:|
|     └      |  **Function Name**  |  **Visibility**  |  **Mutability**  |
**Modifiers**  |
||||||
| **console** | Library |  |||
| └ | _sendLogPayload | Private 🔐 |   | |
| └ | log | Internal 🔒 |   | |
| └ | log | Internal 🔒 |   | |
| └ | log | Internal 🔒 |   | |
| └ | log | Internal 🔒 |   | |
| └ | log | Internal 🔒 |   | |
| └ | log | Internal 🔒 |   | |
| └ | log | Internal 🔒 |   | |
| └ | log | Internal 🔒 |   | |
| └ | log | Internal 🔒 |   | |
| └ | log | Internal 🔒 |   | |
| └ | log | Internal 🔒 |   | |
||||||
| **Counters** | Library |  |||
| └ | current | Internal 🔒 |   | |
| └ | increment | Internal 🔒 | 🛑 | |
| └ | decrement | Internal 🔒 | 🛑 | |
| └ | reset | Internal 🔒 | 🛑 | |
||||||
| **Strings** | Library |  |||
| └ | toString | Internal 🔒 |   | |
| └ | toHexString | Internal 🔒 |   | |
| └ | toHexString | Internal 🔒 |   | |
||||||
| **Context** | Implementation |  |||
| └ | _msgSender | Internal 🔒 |   | |
| └ | _msgData | Internal 🔒 |   | |
||||||
| **Ownable** | Implementation | Context |||
| └ | <Constructor> | Public ❗ | 🛑 |NO❗ |
| └ | owner | Public ❗ |   |NO❗ |
| └ | renounceOwnership | Public ❗ | 🛑 | onlyOwner |
| └ | transferOwnership | Public ❗ | 🛑 | onlyOwner |
| └ | _transferOwnership | Internal 🔒 | 🛑 | |
||||||
| **Address** | Library |  |||
| └ | isContract | Internal 🔒 |   | |
```

└	sendValue	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	functionStaticCall	Internal 🔒		
└	functionStaticCall	Internal 🔒		
└	functionDelegateCall	Internal 🔒	🛑	
└	functionDelegateCall	Internal 🔒	🛑	
└	verifyCallResult	Internal 🔒		
IERC721Receiver	Interface				
└	onERC721Received	External ❗	🛑	NO❗
IERC165	Interface				
└	supportsInterface	External ❗		NO❗
ERC165	Implementation	IERC165		
└	supportsInterface	Public ❗		NO❗
IERC721	Interface	IERC165		
└	balanceOf	External ❗		NO❗
└	ownerOf	External ❗		NO❗
└	safeTransferFrom	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
└	approve	External ❗	🛑	NO❗
└	getApproved	External ❗		NO❗
└	setApprovalForAll	External ❗	🛑	NO❗
└	isApprovedForAll	External ❗		NO❗
└	safeTransferFrom	External ❗	🛑	NO❗
IERC721Enumerable	Interface	IERC721		
└	totalSupply	External ❗		NO❗
└	tokenOfOwnerByIndex	External ❗		NO❗
└	tokenByIndex	External ❗		NO❗
IERC721Metadata	Interface	IERC721		
└	name	External ❗		NO❗
└	symbol	External ❗		NO❗
└	tokenURI	External ❗		NO❗
ERC721	Implementation	Context, ERC165, IERC721, IERC721Metadata		
└	<Constructor>	Public ❗	🛑	NO❗
└	supportsInterface	Public ❗		NO❗
└	balanceOf	Public ❗		NO❗
└	ownerOf	Public ❗		NO❗
└	name	Public ❗		NO❗
└	symbol	Public ❗		NO❗
└	tokenURI	Public ❗		NO❗
└	_baseURI	Internal 🔒		
└	approve	Public ❗	🛑	NO❗
└	getApproved	Public ❗		NO❗
└	setApprovalForAll	Public ❗	🛑	NO❗
└	isApprovedForAll	Public ❗		NO❗
└	transferFrom	Public ❗	🛑	NO❗
└	safeTransferFrom	Public ❗	🛑	NO❗
└	safeTransferFrom	Public ❗	🛑	NO❗
└	_safeTransfer	Internal 🔒	🛑	
└	_exists	Internal 🔒		

└	_isApprovedOrOwner	Internal 🔒		
└	_safeMint	Internal 🔒	🛑	
└	_safeMint	Internal 🔒	🛑	
└	_mint	Internal 🔒	🛑	
└	_burn	Internal 🔒	🛑	
└	_transfer	Internal 🔒	🛑	
└	_approve	Internal 🔒	🛑	
└	_setApprovalForAll	Internal 🔒	🛑	
└	_checkOnERC721Received	Private 🔐	🛑	
└	_beforeTokenTransfer	Internal 🔒	🛑	
└	_afterTokenTransfer	Internal 🔒	🛑	
ERC721Enumerable	Implementation	ERC721, IERC721Enumerable		
└	supportsInterface	Public ❗		NO❗
└	tokenOfOwnerByIndex	Public ❗		NO❗
└	totalSupply	Public ❗		NO❗
└	tokenByIndex	Public ❗		NO❗
└	_beforeTokenTransfer	Internal 🔒	🛑	
└	_addTokenToOwnerEnumeration	Private 🔐	🛑	
└	_addTokenToAllTokensEnumeration	Private 🔐	🛑	
└	_removeTokenFromOwnerEnumeration	Private 🔐	🛑	
└	_removeTokenFromAllTokensEnumeration	Private 🔐	🛑	
ERC721URIStorage	Implementation	ERC721		
└	tokenURI	Public ❗		NO❗
└	_setTokenURI	Internal 🔒	🛑	
└	_burn	Internal 🔒	🛑	
PlatzeeNFT	Implementation	ERC721, ERC721Enumerable, ERC721URIStorage, Ownable		
└	<Constructor>	Public ❗	🛑	ERC721
└	_beforeTokenTransfer	Internal 🔒	🛑	
└	_burn	Internal 🔒	🛑	
└	supportsInterface	Public ❗		NO❗
└	tokenURI	Public ❗		NO❗
└	mint	Public ❗	💵	isMintable limitByAddress
└	mintAll	Public ❗	🛑	isMintable onlyOwner
└	mint_to_address	Public ❗	💵	isMintable isNotExceedAvailableSupplyByAmount onlyOwner
└	getTokenIds	Public ❗		NO❗
└	setBaseURI	Public ❗	🛑	onlyOwner
└	setMintable	Public ❗	🛑	onlyOwner
└	_baseURI	Internal 🔒		
└	isApprovedForAll	Public ❗		NO❗
└	setCurrentPrice	Public ❗	🛑	onlyOwner
└	getBalance	Public ❗		onlyOwner
└	withdraw	Public ❗	💵	onlyOwner

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

Conclusion

Team found high issues. So, it isnot 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 “ In secured”.

✓ Has high severity issues were found.

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.