SMART CONTRACT AUDIT REPORT

For

Peace Rabbit NFT

Prepared By: SFI Team

Prepared on: 30/5/2022

Prepared for: Peace Rabbit Team

Contract address: Not deployed yet

Table of Content

- ## 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) 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.

• Scope of the audit

The scope of this audit was to analyze and document the Peace Rabbit NFT smart contract codebase for quality, security, and correctness.

• Introduction

During the period of **May 29, 2022, to May 30, 2022** - SaferICO

Team performed a security audit for **Peace Rabbit NFT** smart contracts.

The project has 1 file. It contains approx 159 lines of Solidity code. Most of the functions and state variables are well commented on using the Nat spec documentation, but that does not create any vulnerability.

Source Code: https://github.com/Saferico/Smart-Contracts-for-Projects/blob/main/NFT_PRESALE.sol

Check Vulnerabilities

In order to check for the security of the contract, we tested several attacks in order to make sure that the contract is secure and follows best practices automatically.

1. Unit tests passing.

2. Compilator warnings;

3. Race Conditions. Reentrancy. Cross-function Race Conditions. Pitfalls in Race Condition solutions;

4. Possible delays in data delivery;

5. Transaction-Ordering Dependence (front running);

6. Timestamp Dependence;

7. Integer Overflow and Underflow;

8. DoS with (unexpected) Revert;

9. DoS with Block Gas Limit

10. Call Depth Attack. Not relevant in modern ethereum network

11. Methods execution permissions;

12. Oracles calls;

13. Economy model. It's important to forecast scenarios when a user is provided with additional economic motivation or faced with limitations. If application logic is based on incorrect economy model, the application will not function correctly and participants will incur financial losses. This type of issue is most often found in bonus rewards systems.

14. The impact of the exchange rate on the logic;

15. Private user data leaks.

• Issue Categories

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below.

Risk-level	Description
High	A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.
Medium	The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.
Low	Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.
Informational	These are severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

- # Issues Found – Code Review

High severity issues

There are no High severity vulnerabilities found.

Medium severity issues

There are no Medium severity vulnerabilities found.

Low severity issues

#Missing zero address validation

Description

When the owner uses the add to whitelist functions, he has to check for the zero address, he didn't add the burn address to the whitelist. Otherwise, it will cost him more ETH gas.

```
function whitelistUsers(address[] calldata _users) public onlyOwner {
   delete whitelistedAddresses;
   whitelistedAddresses = _users;
}
```

Remediation

Use the require statement to check for zero addresses.

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 WL or non WL stage.

The owner can add any address from the whitelist.

The owner can pause / un pause the smart contract.

```
function setCost(uint256 _newCost) public onlyOwner {
    cost = _newCost;
}
function pause(bool _state) public onlyOwner {
    paused = _state;
}
function whitelistUsers(address[] calldata _users) public onlyOwner {
    delete whitelistedAddresses;
    whitelistedAddresses = _users;
}
```

Remediation

Make these functions internal in next version or the team should announce the investors before pause and un pause 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.

#Pragam version not fixed

Description

It is a good practice to lock the solidity version for a live deployment (use 0.8.13 instead of >=0.7.0 <0.9.0). 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: Closed. Fixed in version 2.

Informational issues

There are no Informational issues found.

- ## Source Lines
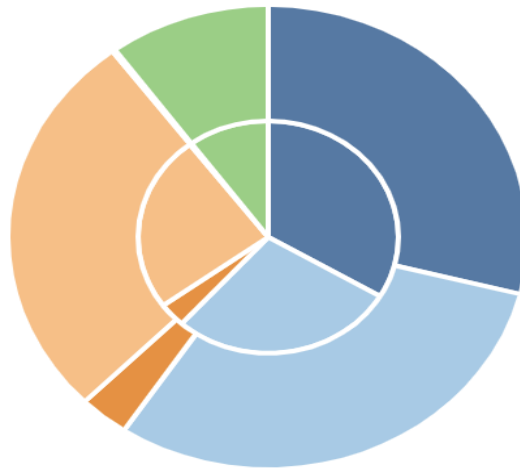
- ## Risk Level
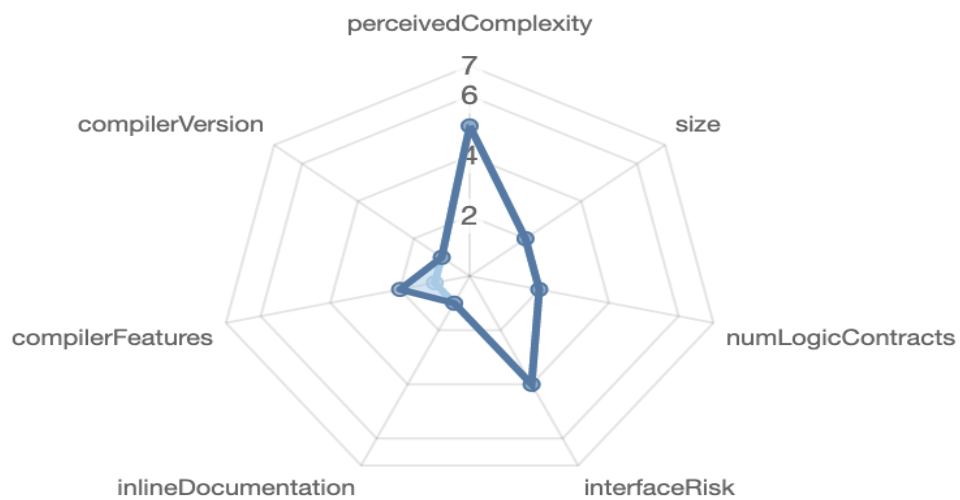
- **Testing proves**

- **Check for security**

- **Solidity Static Analysis**

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

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

- **Inheritance graph**
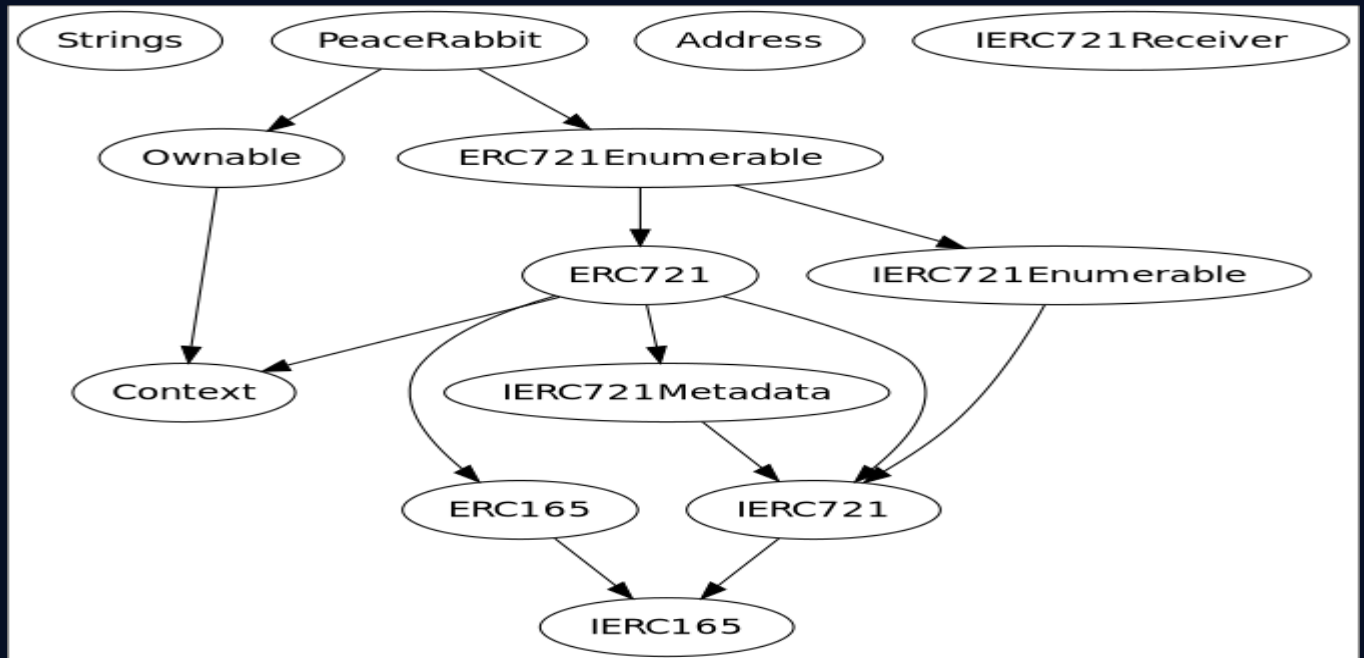
- **Solidity Unit Testing Code & Results**

```solidity
// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.4.22 <0.9.0;

// This import is automatically injected by Remix
import "remix_tests.sol";

// This import is required to use custom transaction context
// Although it may fail compilation in 'Solidity Compiler' plugin
// But it will work fine in 'Solidity Unit Testing' plugin
import "remix_accounts.sol";
import "../NFT_PRESALE.sol";

// File name has to end with '_test.sol', this file can contain more than one testSuite contracts
contract testSuite {

    /// 'beforeAll' runs before all other tests
```

SOLIDITY UNIT TESTING

Test your smart contract in Solidity.

Select directory to load and generate test files.

Test directory:

| tests | Create |

| Generate | How to use... |

| ▶ Run | ⬛ Stop |

☑ Select all

☑ tests/NFT_PRESALE_test.sol

Progress: 1 finished (of 1)

PASS testSuite

(tests/NFT_PRESALE_test.sol)

✓ Before all	🐞
✓ Check success	🐞
✓ Check success2	🐞
✓ Check failure	🐞
✓ Check sender and value	🐞

Result for tests/NFT_PRESALE_test.sol
Passed: 5
Failed: 0
Time Taken: 0.34s

```solidity
/// More special functions are: 'beforeEach', 'beforeAll', 'afterEach' &
'afterAll'
    function beforeAll() public {
        // <instantiate contract>
        Assert.equal(uint(1), uint(1), "1 should be equal to 1");

    }


    function checkSuccess() public {
        // Use 'Assert' methods: https://remix-
ide.readthedocs.io/en/latest/assert_library.html

        Assert.ok(2 == 2, 'should be true');
        Assert.greaterThan(uint(2), uint(1), "2 should be greater than to
1");
        Assert.lesserThan(uint(2), uint(3), "2 should be lesser than to 3");
    }


    function checkSuccess2() public pure returns (bool) {
        // Use the return value (true or false) to test the contract
        return true;
    }


    function checkFailure() public {
        Assert.notEqual(uint(1), uint(2), "1 should not be equal to 1");
    }


    /// Custom Transaction Context: https://remix-
ide.readthedocs.io/en/latest/unittesting.html#customization
    /// #sender: account-1
    /// #value: 100
    function checkSenderAndValue() public payable {
        // account index varies 0-9, value is in wei
        Assert.equal(msg.sender, TestsAccounts.getAccount(1), "Invalid
sender");

        Assert.equal(msg.value, 100, "Invalid value");
    }
}
```

Call Graph
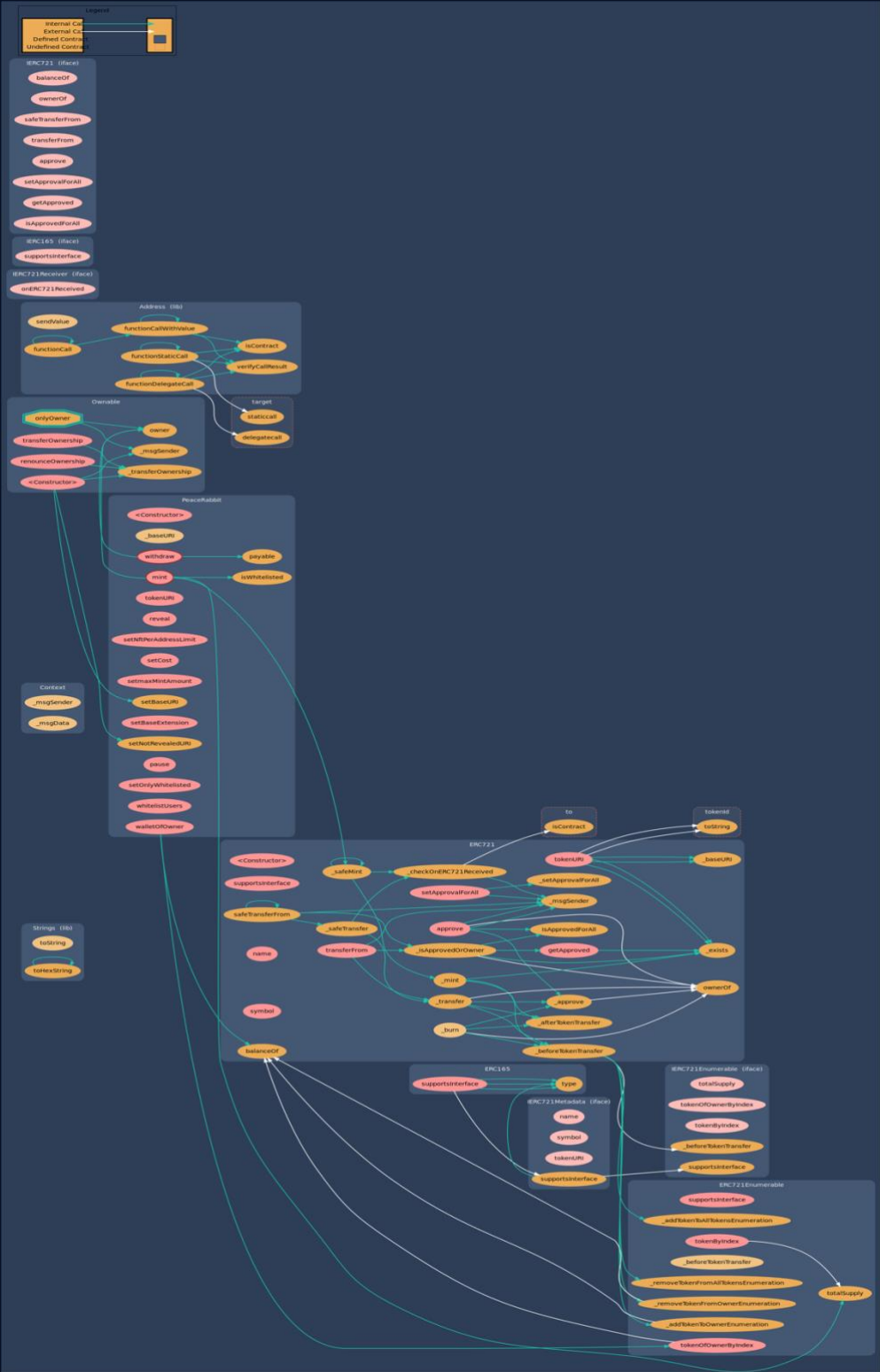
Unified Modeling Language (UML)
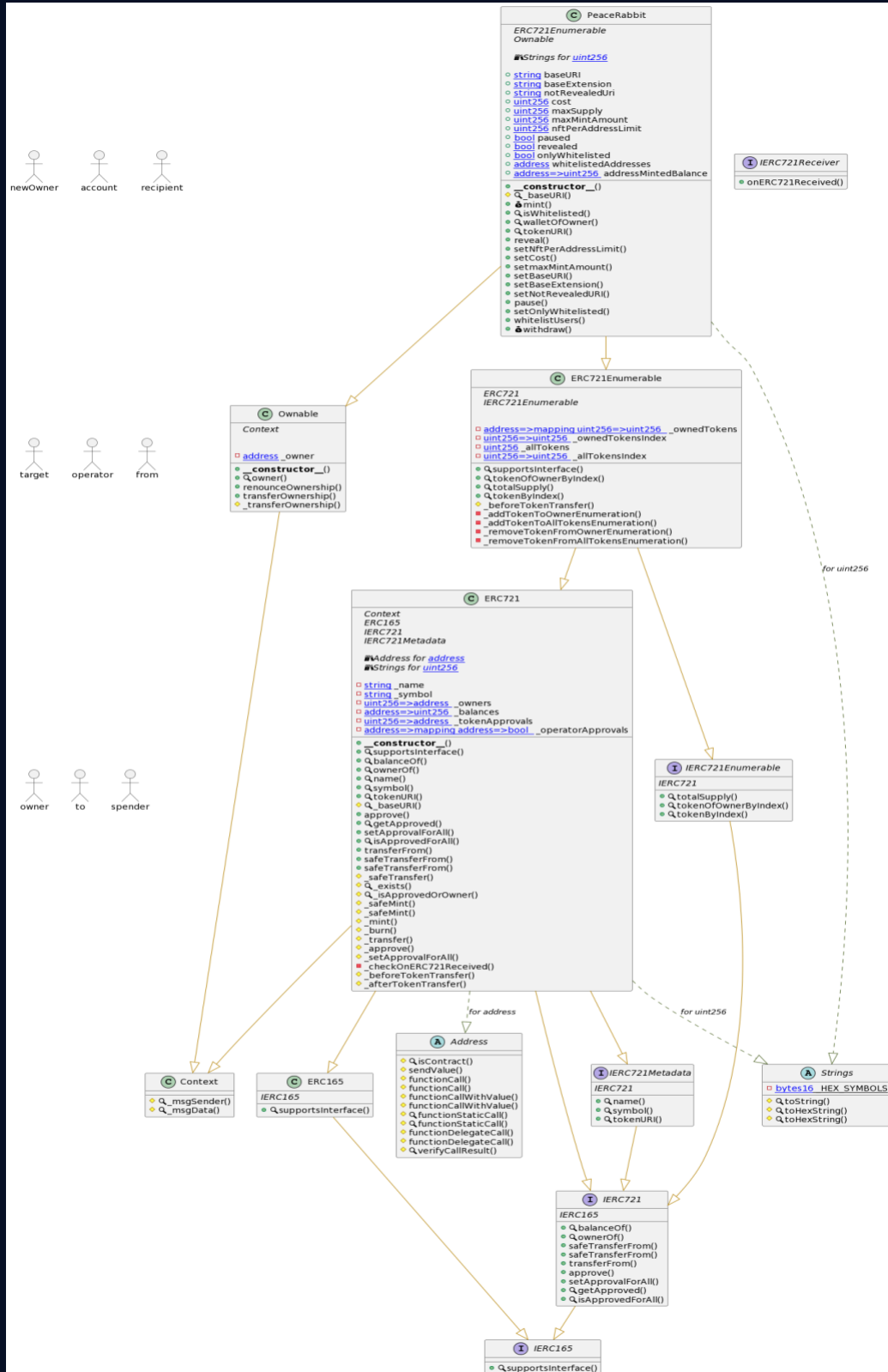
• Capabilities

Components

📝Contracts	📚Libraries	🔍Interfaces	🥚Abstract
2	2	5	4

Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

🌐Public	💰Payable
53	2

External	Internal	Private	Pure	View
17	82	5	4	37

StateVariables

Total	🌐Public
24	12

Capabilities

Solidity Versions observed	🖊 Experimental Features	💰 Can Receive Funds	🖥 Uses Assembly	🐚 Has Destroyable Contracts
^0.8.0 ^0.8.1 >=0.7.0 <0.9.0		yes	yes (2 asm blocks)	

⛏ Transfers ETH	⚡ Low-Level Calls	👥 DelegateCall	🔡 Uses Hash Functions	✒ ECRecover	⚙ New/Create/Create2
		yes			

♻ TryCatch	Σ Unchecked
yes	

• Source Units In Scope

Source Units in Scope

Source Units Analyzed: 1
Source Units in Scope: 1 (100%)

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝📚🔍🥚	NFT_PRESALE.sol	8	5	1468	1187	557	645	460	🖥💰👥🌟🐚
📝📚🔍🥚	Totals	8	5	1468	1187	557	645	460	🖥💰👥🌟🐚

Legend: [—]

- **Lines**: total lines of the source unit

- **nLines**: normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)

- **nSLOC**: normalized source lines of code (only source-code lines; no comments, no blank lines)

- **Comment Lines**: lines containing single or block comments

- **Complexity Score**: a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

• Function Signature

```
Sighash   |   Function Signature
=======================
16279055  =>  isContract(address)
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)
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)
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)

a0712d68 => mint(uint256)

3af32abf => isWhitelisted(address)

438b6300 => walletOfOwner(address)

a475b5dd => reveal()

d0eb26b0 => setNftPerAddressLimit(uint256)

44a0d68a => setCost(uint256)

7f00c7a6 => setmaxMintAmount(uint256)

55f804b3 => setBaseURI(string)

da3ef23f => setBaseExtension(string)

f2c4ce1e => setNotRevealedURI(string)

02329a29 => pause(bool)

3c952764 => setOnlyWhitelisted(bool)

edec5f27 => whitelistUsers(address[])

3ccfd60b => withdraw()
```

- ## **Automatic General Report**

Files Description Table

File Name	SHA-1 Hash
/Users/macbook/Desktop/smart contracts/NFT_PRESALE.sol	07321544e4058c221c362d5e309bbf61f131d9ef

Contracts Description Table

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
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❗
└	safeTransferFrom	External ❗	⬤	NO❗
└	transferFrom	External ❗	⬤	NO❗
└	approve	External ❗	⬤	NO❗
└	setApprovalForAll	External ❗	⬤	NO❗
└	getApproved	External ❗		NO❗
└	isApprovedForAll	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 🔐	⬤		
PeaceRabbit	Implementation	ERC721Enumerable, Ownable		
└	<Constructor>	Public ❗	⬤	ERC721
└	_baseURI	Internal 🔒		
└	mint	Public ❗	💶	NO❗
└	isWhitelisted	Public ❗		NO❗
└	walletOfOwner	Public ❗		NO❗
└	tokenURI	Public ❗		NO❗
└	reveal	Public ❗	⬤	onlyOwner
└	setNftPerAddressLimit	Public ❗	⬤	onlyOwner
└	setCost	Public ❗	⬤	onlyOwner
└	setmaxMintAmount	Public ❗	⬤	onlyOwner
└	setBaseURI	Public ❗	⬤	onlyOwner
└	setBaseExtension	Public ❗	⬤	onlyOwner
└	setNotRevealedURI	Public ❗	⬤	onlyOwner
└	pause	Public ❗	⬤	onlyOwner
└	setOnlyWhitelisted	Public ❗	⬤	onlyOwner
└	whitelistUsers	Public ❗	⬤	onlyOwner
└	withdraw	Public ❗	💶	onlyOwner

Legend

Symbol	Meaning
⬤	Function can modify state
💶	Function is payable

- ## Summary of the Audit

According to all test, the customer`s solidity smart contract is **Well Secure**.

The general overview is presented in the Project Information section and all issues found are located in the audit overview section.

The test found 0 critical, 0 high, 0 medium, 3 low issues, and 0 notes.