Smart Contract
Security Audit
V1

Athlete Hero Smart Contract

15/10/2022

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**: 0x9848d2992cb57c5856a0be62297e66554d242238
- **Code:**

https://github.com/Saferico/Smart-Contracts-for-Projects/blob/main/AthleteHero_nft.sol

NFT Information

- Name: Athlete Hero
- Total Supply: 100
- Holders:
- Total transactions:

Contracts address deployed to test net (Ethereum)

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

https://goerli.etherscan.io/address/0x9848d2992cb57c5856a0be62297e66554d242238

https://goerli.etherscan.io/address/0x0490e583201c4b21c8d2c0bc67041f0c505fe264

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, 1 high, 0 medium, 4 low, 0 very low-level issues and 2 notes in all solidity files of the contract

The files:

AthleteHero.sol

File and Function Level Report

File in Scope:

Contract Name	SHA 256 hash	Contract Address
AthleteHero.sol	**8963d2a357dccbca924509f4fd87dc795c0008cd7a1c7a08fb41d1c9871f10e0**	0x9848d2992cb57c5856a0be62297e66554d242238

- Contract: AthleteHeroTeam
- Inherit: ERC721, ERC721Enumerable, Pausable, Ownable, ERC721Burnable, ReentrancyGuard
- 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**
costWhitelist	✓	Read / public	**Passed**
supportsInterface	✓	Read / public	**Passed**
cost	✓	Read / public	**Passed**
balanceOf	✓	Read / public	**Passed**
Owner	✓	Read / public	**Passed**
tokenOfOwnerByIndex	✓	Read / public	**Passed**
tokenOfByIndex	✓	Read / public	**Passed**
getApprovedForAll	✓	Read / public	**Passed**
maxMintAmount	✓	Read / public	**Passed**
getApproved	✓	Read / public	**Passed**
getOwnerOf	✓	Read / public	**Passed**

ownerOf	✓	Read / public	**Passed**
tokenURI	✓	Read / public	**Passed**
totalSupply	✓	Read / public	**Passed**
maxSupply	✓	Read / public	**Passed**
getTokenIds	✓	Read / public	**Passed**
paused	✓	Read / public	**Passed**
pauseGeneralMint	✓	Read / public	**Passed**
pauseWhitelistMint	✓	Read / public	**Passed**
burn	✓	Write / public	**Passed**
mint	✓	Write / payable	**Passed**
approve	✓	Write / public	**Passed**
safeTransferFrom	✓	Write / public	**Passed**
safeTransferFrom	✓	Write / public	**Passed**
Paused	✓	Write / public	**Passed**
mintWhitelist	✓	Write / payable	**Passed**
unPaused	✓	Write / public	**Passed**
transferOwnership	✓	Write / public	**Passed**
setApprovalForAll	✓	Write / public	**Passed**
transferFrom	✓	Write / public	**Passed**
_pauseGeneralMint	✓	Write / public	**Passed**
_pauseWhitelistMint	✓	Write / public	**Passed**
renounceOwnership	✓	Write / public	**Passed**
_setBaseURI	✓	Write / public	**Passed**
addWhiteList	✓	Write / public	**Passed**
ownerMint	✓	Write / public	**Passed**
setCost	✓	Write / public	**Passed**
setSupply	✓	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**
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:

#The Funds will be locked forever

Description

The smart contract is for mint NFT with price but doesn't have any withdraw function so the funds will be locked forever in the smart contract and the team can't take any funds from it.

Remediation

Add withdraw function which can only control by the owner able.

Status: Closed. Fixed in version 2.

Medium:

No Medium severity vulnerabilities were found

Low:

#Missing zero address validation

Description

When the owner wants to add the addresses to whitelist, and owner Mint function too, he has to check for the zero address to make, he didn't add the zero address. Otherwise, he will lose some eth gas.

```
function addWhiteList(address[] calldata addresses,uint[] calldata mintAmounts)
external onlyOwner {
        for (uint i = 0; i < addresses.length; i++) {
                whitelist[addresses[i]] =  uint8(mintAmounts[i]);}}

    function ownerMint(address to) public onlyOwner nonReentrant {
        uint256 tokenId = _tokenIdCounter.current();
        _tokenIdCounter.increment();
        _safeMint(to, tokenId);}
```

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 add any address to the whitelist.
The owner can pause and un pause the contract.
The owner can change the price in Whitelist and public mint.
The owner can change the total supply of NFT.
The owner can mint to any address.

```
function pausedGeneralMint(bool _pauseValue) public onlyOwner {
        pauseGeneralMint = _pauseValue;
    }

    function pausedWhitelistMint(bool _pauseValue) public onlyOwner {
        pauseWhitelistMint = _pauseValue;
    }

    function setSupply(uint256 _newMaxSupply,uint256 _newMaxAmount) public
onlyOwner {
        maxSupply = _newMaxSupply;
        maxMintAmount = _newMaxAmount;
    }

    function setCost(uint256 _cost,uint256 _costWhitelist) public onlyOwner {
        cost = _cost;
        costWhitelist = _costWhitelist;
    }

    function addWhiteList(address[] calldata addresses,uint[] calldata mintAmounts)
external onlyOwner {
        for (uint i = 0; i < addresses.length; i++) {
            require(addresses[i] != address(0),"Can't add for zero address!");
                whitelist[addresses[i]] =  uint8(mintAmounts[i]);
        }
    }
```

```
    function ownerMint(address to) public onlyOwner nonReentrant {
        require(to != address(0),"Can't mint for zero address!");
        uint256 tokenId = _tokenIdCounter.current();
        _tokenIdCounter.increment();
        _safeMint(to, tokenId);
    }
```

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.

#Useless functions used in the contract

Description

The smart contract has useless functions like pause and un-pause functions, there are pause/unpause functions for whitelist mint and the same for the public mint too. So no need for the pause and un-pause functions.

```
function pause() public onlyOwner {
        _pause();
    }

    function unpause() public onlyOwner {
        _unpause();
    }

    function pausedGeneralMint(bool _pauseValue) public onlyOwner {
        pauseGeneralMint = _pauseValue;
    }

    function pausedWhitelistMint(bool _pauseValue) public onlyOwner {
        pauseWhitelistMint = _pauseValue;
    }
```

Remediation

Remove all useless or unnecessary functions from the smart contract to save ETH gas.

Status: Closed. Fixed in version 2.

Very Low:

No Very Low severity vulnerabilities were found.

Notes:

#Naming Conventions

Description
The contract follows a consistent naming convention where we are private variables with leading"_" and public variables without it. But we have missed to comply to the condition for certain variable names "__ pauseGeneralMint " which is public.

Remediation
Remove "_" from external variable names and add it to private variable names.

Status: Closed. Fixed in version2.

#Unnecessary import of ERC721 library

Description
The main contract inherits: ERC721, ERC721Enumerable, ERC721URIStorage, Ownable, AccessControl , ERC721Enumerable 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

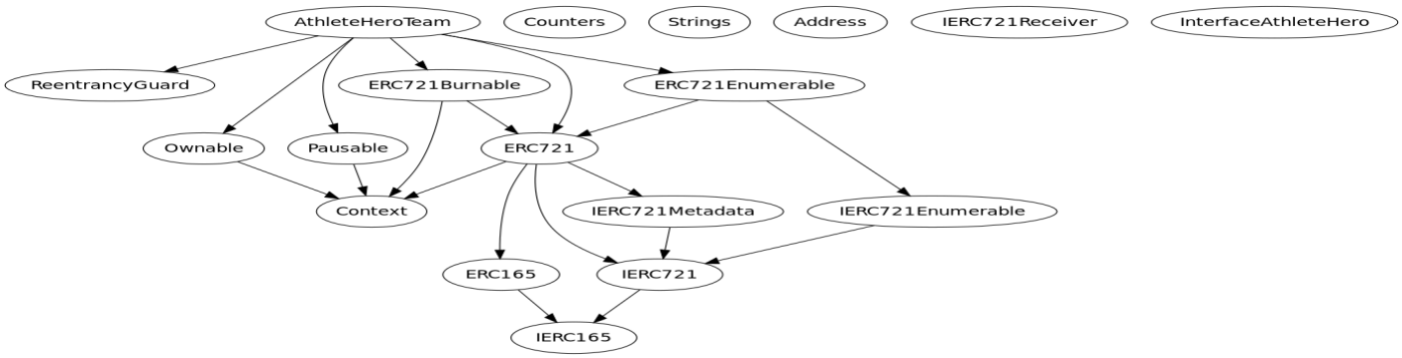
8963d2a357dccbca924509f4fd87dc795c0008cd7a1c7a08fb41d1c9871f10e0

File: AthleteH... | Language: solidity | Size: 6105 bytes | Date: 2022-10-15T09:33:41.522Z

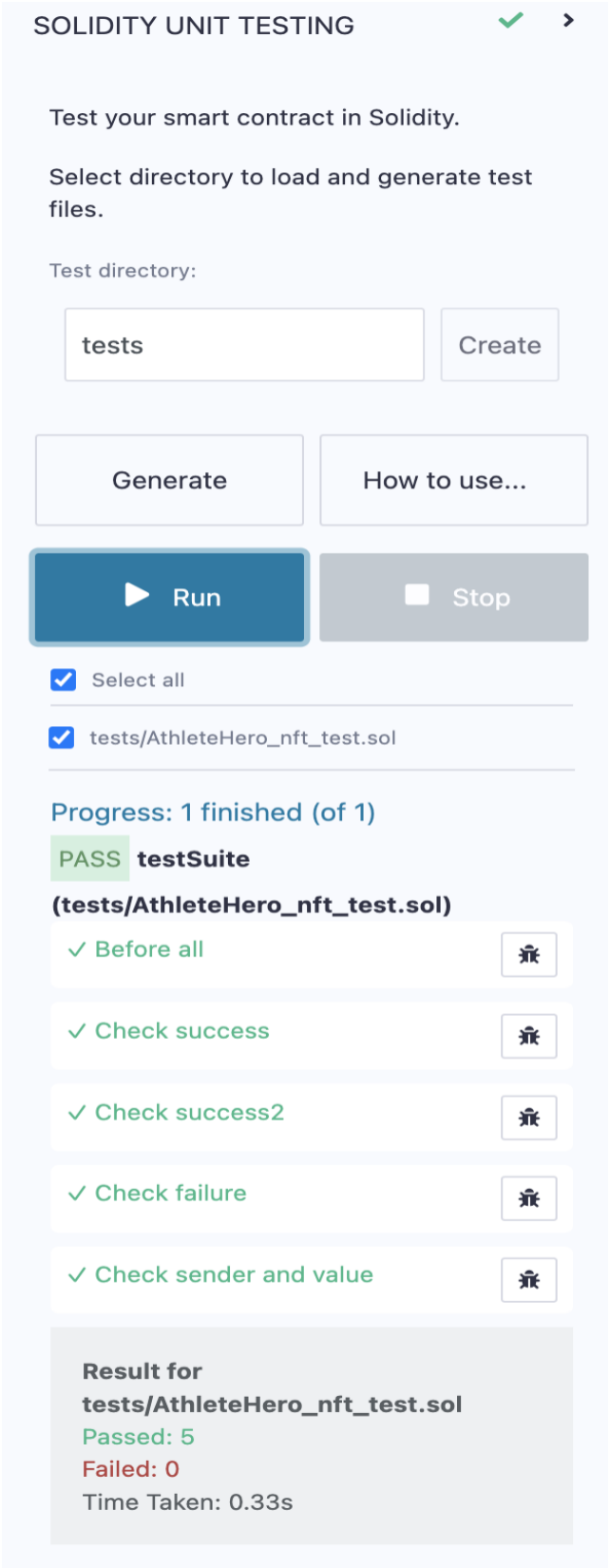
Critical	High	Medium	Low	Note
0	0	0	0	0

2- SOLIDITY STATIC ANALYSIS

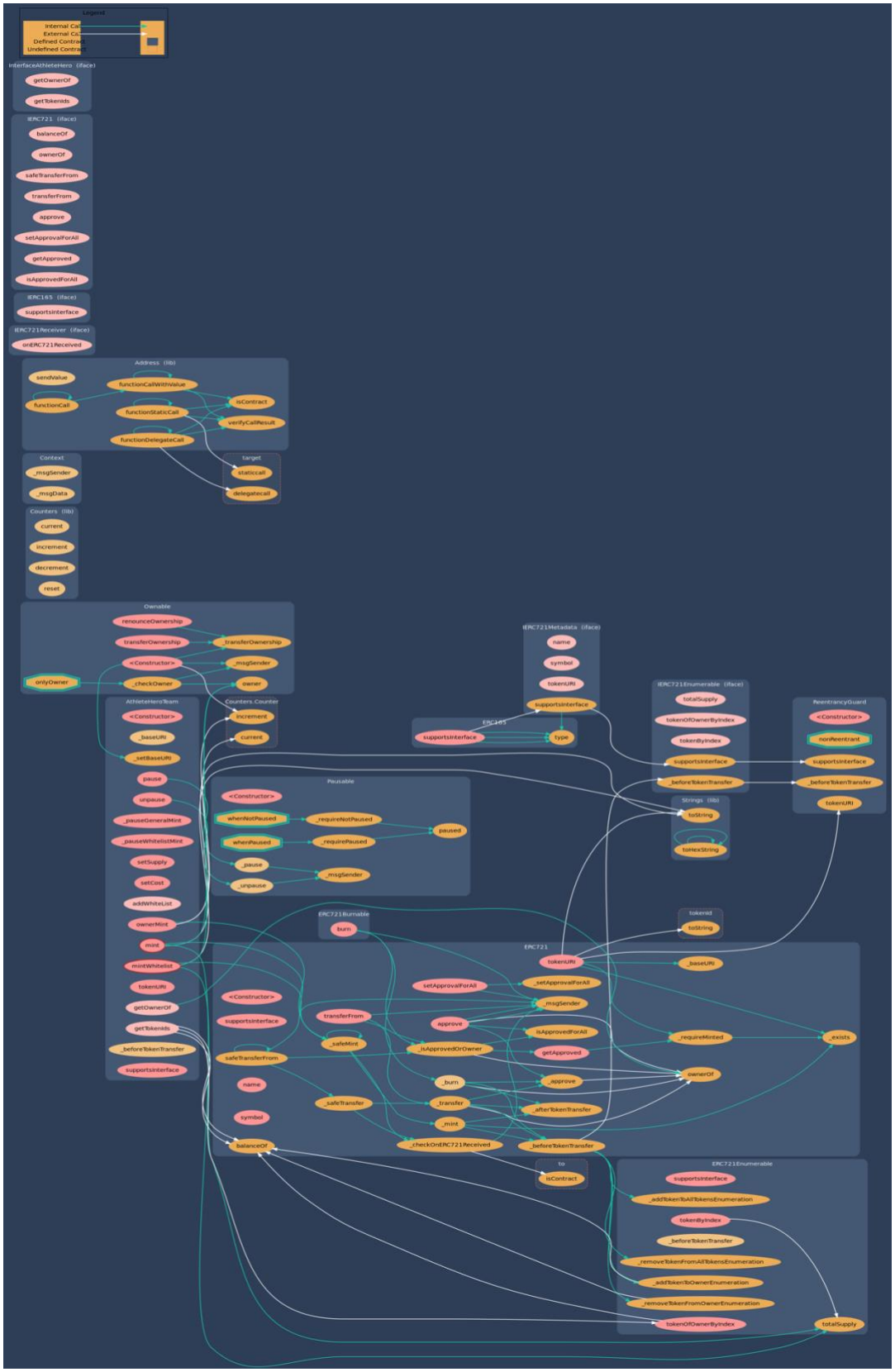
3- Inheritance graph



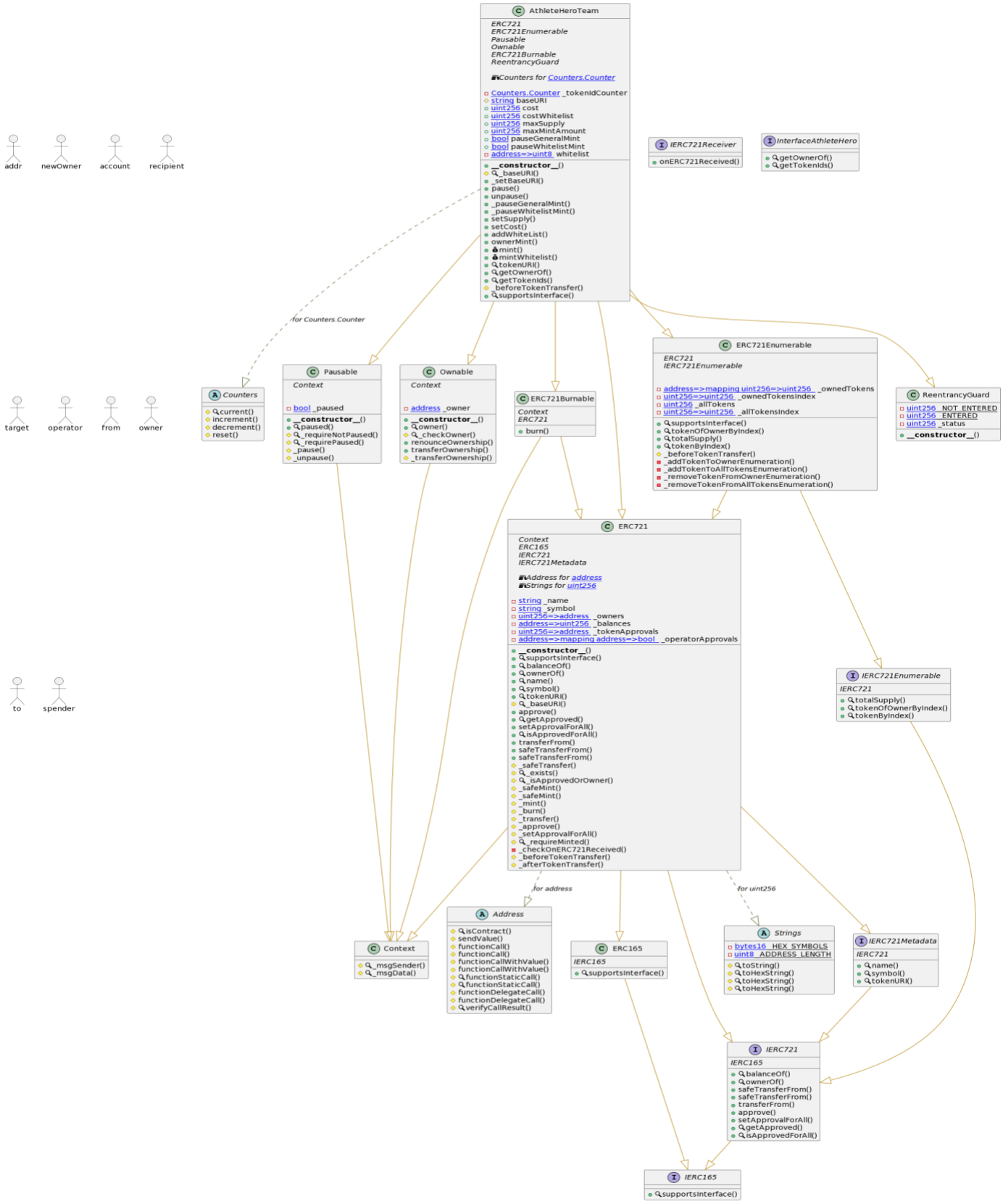
4- SOLIDITY UNIT TESTING



5- Call graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
InterfaceAthleteHero (iface)
getOwnerOf
getTokenIds
IERC721 (iface)
balanceOf
ownerOf
safeTransferFrom
transferFrom
approve
setApprovalForAll
getApproved
isApprovedForAll
IERC165 (iface)
supportsInterface
IERC721Receiver (iface)
onERC721Received
Address (lib)
sendValue
functionCallWithValue
functionCall
functionStaticCall
functionDelegateCall
isContract
verifyCallResult
target
staticcall
delegatecall
Context
_msgSender
_msgData
Counters (lib)
current
increment
decrement
reset
Ownable
renounceOwnership
transferOwnership
_transferOwnership
<Constructor>
onlyOwner
_checkOwner
owner
_msgSender
AthleteHeroTeam
<Constructor>
_baseURI
_setBaseURI
pause
unpause
_pauseGeneralMint
_pauseWhitelistMint
setSupply
setCost
addWhiteList
ownerMint
mint
mintWhitelist
tokenURI
getOwnerOf
getTokenIds
_beforeTokenTransfer
supportsInterface
Counters.Counter
increment
current
Pausable
<Constructor>
whenNotPaused
_requireNotPaused
paused
whenPaused
_requirePaused
_pause
_unpause
_msgSender
ERC721Burnable
burn
IERC721Metadata (iface)
name
symbol
tokenURI
supportsInterface
ERC165
supportsInterface
type
IERC721Enumerable (iface)
totalSupply
tokenOfOwnerByIndex
tokenByIndex
supportsInterface
_beforeTokenTransfer
ReentrancyGuard
<Constructor>
nonReentrant
supportsInterface
_beforeTokenTransfer
tokenURI
Strings (lib)
toString
toHexString
tokenId
toString
ERC721
tokenURI
_baseURI
setApprovalForAll
_setApprovalForAll
_msgSender
<Constructor>
supportsInterface
transferFrom
approve
isApprovedForAll
getApproved
_requireMinted
_exists
safeTransferFrom
_safeMint
_isApprovedOrOwner
ownerOf
name
symbol
_burn
_approve
_safeTransfer
_transfer
_afterTokenTransfer
_mint
balanceOf
_checkOnERC721Received
_beforeTokenTransfer
to
isContract
ERC721Enumerable
supportsInterface
_addTokenToAllTokensEnumeration
tokenByIndex
_beforeTokenTransfer
_removeTokenFromAllTokensEnumeration
_addTokenToOwnerEnumeration
_removeTokenFromOwnerEnumeration
tokenOfOwnerByIndex
totalSupply

Unified Modeling Language (UML)



Functions signature

```
Sighash   |   Function Signature
========================
16279055  =>  isContract(address)
83638710  =>  getOwnerOf(uint256)
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)
5c975abb  =>  paused()
abb87a6f  =>  _requireNotPaused()
4a994e05  =>  _requirePaused()
320b2ad9  =>  _pause()
fc8234cb  =>  _unpause()
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)
a0aea85d  =>  _requireMinted(uint256)
1fd01de1  =>  _checkOnERC721Received(address,address,uint256,bytes)
cad3be83  =>  _beforeTokenTransfer(address,address,uint256)
8f811a1c  =>  _afterTokenTransfer(address,address,uint256)
42966c68  =>  burn(uint256)
69025b5f  =>  _addTokenToOwnerEnumeration(address,uint256)
e03d890b  =>  _addTokenToAllTokensEnumeration(uint256)
68df0d53  =>  _removeTokenFromOwnerEnumeration(address,uint256)
4cbb4a0a  =>  _removeTokenFromAllTokensEnumeration(uint256)
d004b036  =>  getTokenIds(address)
31b5b907  =>  _setBaseURI(string)
8456cb59  =>  pause()
3f4ba83a  =>  unpause()
391d3c21  =>  _pauseGeneralMint(bool)
a1781243  =>  _pauseWhitelistMint(bool)
fc784d49  =>  setSupply(uint256,uint256)
7696e088  =>  setCost(uint256,uint256)
a0d41d9f  =>  addWhiteList(address[],uint256[])
1e3bcc8e  =>  ownerMint(address)
a0712d68  =>  mint(uint256)
4618163e  =>  mintWhitelist(uint256)
```

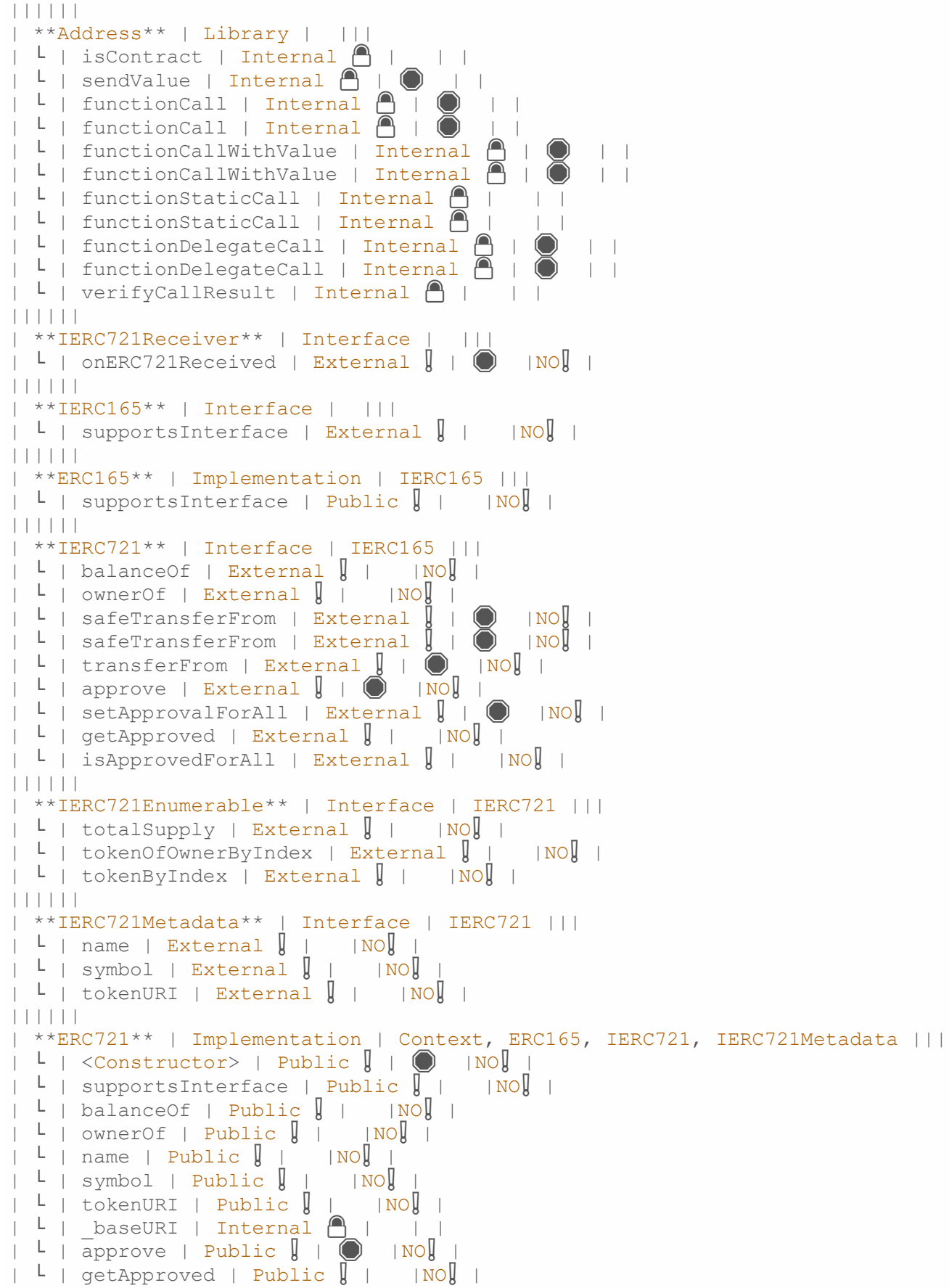
Automatic general report

Files Description Table

File Name	SHA-1 Hash
/Users/macbook/Desktop/smart contracts/AthleteHero_nft.sol	926586d0490d1de40962d9cf40e5777a7c4fba91

Contracts Description Table

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
ReentrancyGuard	Implementation			
└	<Constructor>	Public ❗	🛑	NO❗
Counters	Library			
└	current	Internal 🔒		
└	increment	Internal 🔒	🛑	
└	decrement	Internal 🔒	🛑	
└	reset	Internal 🔒	🛑	
Strings	Library			
└	toString	Internal 🔒		
└	toHexString	Internal 🔒		
└	toHexString	Internal 🔒		
└	toHexString	Internal 🔒		
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🛑	NO❗
└	owner	Public ❗		NO❗
└	_checkOwner	Internal 🔒		
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	_transferOwnership	Internal 🔒	🛑	
Pausable	Implementation	Context		
└	<Constructor>	Public ❗	🛑	NO❗
└	paused	Public ❗		NO❗
└	_requireNotPaused	Internal 🔒		
└	_requirePaused	Internal 🔒		
└	_pause	Internal 🔒	🛑	whenNotPaused
└	_unpause	Internal 🔒	🛑	whenPaused


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 🔒	🛑	
└	_requireMinted	Internal 🔒		
└	_checkOnERC721Received	Private 🔐	🛑	
└	_beforeTokenTransfer	Internal 🔒	🛑	
└	_afterTokenTransfer	Internal 🔒	🛑	
ERC721Burnable	Implementation	Context, ERC721		
└	burn	Public ❗	🛑	NO❗
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 🔐	🛑	
InterfaceAthleteHero	Interface			
└	getOwnerOf	External ❗		NO❗
└	getTokenIds	External ❗		NO❗
AthleteHeroTeam	Implementation	ERC721, ERC721Enumerable, Pausable, Ownable, ERC721Burnable, ReentrancyGuard		
└	<Constructor>	Public ❗	🛑	ERC721
└	_baseURI	Internal 🔒		
└	_setBaseURI	Public ❗	🛑	onlyOwner
└	pause	Public ❗	🛑	onlyOwner
└	unpause	Public ❗	🛑	onlyOwner
└	_pauseGeneralMint	Public ❗	🛑	onlyOwner
└	_pauseWhitelistMint	Public ❗	🛑	onlyOwner
└	setSupply	Public ❗	🛑	onlyOwner
└	setCost	Public ❗	🛑	onlyOwner
└	addWhiteList	External ❗	🛑	onlyOwner
└	ownerMint	Public ❗	🛑	onlyOwner nonReentrant
└	mint	Public ❗	💸	nonReentrant whenNotPaused
└	mintWhitelist	Public ❗	💸	nonReentrant whenNotPaused
└	tokenURI	Public ❗		NO❗
└	getOwnerOf	External ❗		NO❗

└	getTokenIds	External ❗		NO❗
└	_beforeTokenTransfer	Internal 🔒	🛑	whenNotPaused
└	supportsInterface	Public ❗		NO❗

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 after fixing the high issue.

✓ 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.