Smart Contract
Security Audit
V1

Hoodies Smart Contract

25/3/2023

https://saferico.com/

business@saferico.com
https://t.me/SFI_ANN

–

Table of Contents

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**: Harmony
- **Name**: Hoodies (Hoodies1, Hoodies2)
- **Contract Address**: not deployed yet
- **Code:**

https://docs.google.com/document/d/1k9q9XH-nACSPStDq1qr2nB_7gqSKLwMBhH3uvLUQc1k/edit

Executive Summary

According to our assessment, the customer`s solidity smart contract is **“WELL SECURED”**.

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

The files:

Hoodies.sol

File and Function Level Report

File in Scope:

Contract Name	SHA 256 hash	Contract Address
Hoodies1.sol	**d914bfa0033befd13fba0bd04b46224da4a8d54cf234493299bff94210233e87**	Not deployed yet

- Contract: Hoodies1
- Inherit: ERC721A, ERC721AQueryable, Ownable, 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**
explicitOwnershipsOf	✓	Read / public	**Passed**
supportsInterface	✓	Read / public	**Passed**
explicitOwnershipsOf	✓	Read / public	**Passed**
balanceOf	✓	Read / public	**Passed**
Owner	✓	Read / public	**Passed**
tokensOfOwnerIn	✓	Read / public	**Passed**
tokensOfOwner	✓	Read / public	**Passed**
getApprovedForAll	✓	Read / public	**Passed**
totalSupply	✓	Read / public	**Passed**
getApproved	✓	Read / public	**Passed**
ownerOf	✓	Read / public	**Passed**

tokenURI	✓	Read / public	**Passed**
withdraw	✓	Write / public	**Passed**
mint	✓	Write / payable	**Passed**
approve	✓	Write / payable	**Passed**
safeTransferFrom	✓	Write / payable	**Passed**
safeTransferFrom	✓	Write / payable	**Passed**
transferOwnership	✓	Write / public	**Passed**
setApprovalForAll	✓	Write / public	**Passed**
transferFrom	✓	Write / payable	**Passed**
set_base_uri	✓	Write / public	**Passed**
set_whitelist	✓	Write / public	**Passed**
renounceOwnership	✓	Write / public	**Passed**

File in Scope:

Contract Name	SHA 256 hash	Contract Address
Hoodies2.sol	**d914bfa0033befd13fba0bd04b46224da4a8d54cf234493299bff94210233e87**	Not deployed yet

- Contract: Hoodies2
- Inherit: ERC721A, ERC721AQueryable, Ownable, 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**
explicitOwnershipsOf	✓	Read / public	**Passed**
supportsInterface	✓	Read / public	**Passed**
explicitOwnershipsOf	✓	Read / public	**Passed**
balanceOf	✓	Read / public	**Passed**
Owner	✓	Read / public	**Passed**
tokensOfOwnerIn	✓	Read / public	**Passed**
tokensOfOwner	✓	Read / public	**Passed**
getApprovedForAll	✓	Read / public	**Passed**
totalSupply	✓	Read / public	**Passed**
getApproved	✓	Read / public	**Passed**
ownerOf	✓	Read / public	**Passed**

tokenURI	✓	Read / public	**Passed**
withdraw	✓	Write / public	**Passed**
mint	✓	Write / payable	**Passed**
approve	✓	Write / payable	**Passed**
safeTransferFrom	✓	Write / payable	**Passed**
safeTransferFrom	✓	Write / payable	**Passed**
transferOwnership	✓	Write / public	**Passed**
setApprovalForAll	✓	Write / public	**Passed**
transferFrom	✓	Write / payable	**Passed**
set_base_uri	✓	Write / public	**Passed**
set_whitelist	✓	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**
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:

#Pragam version not fixed

Description

It is a good practice to lock the solidity version for a live deployment (use 0.8.19 instead of ^0.8.9). 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: Acknowledged.

Very Low:

No Very Low severity vulnerabilities were found.

Notes:

#Unnecessary import of ERC721A library and ERC721Enumerable library

Description

The main contract inherits: ERC721A, ReentrancyGuard, Ownable, ERC721AQueryable which is already import ERC721A library and ERC721Enumerable 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: Acknowledged.

Automatic Testing

1- Check for security

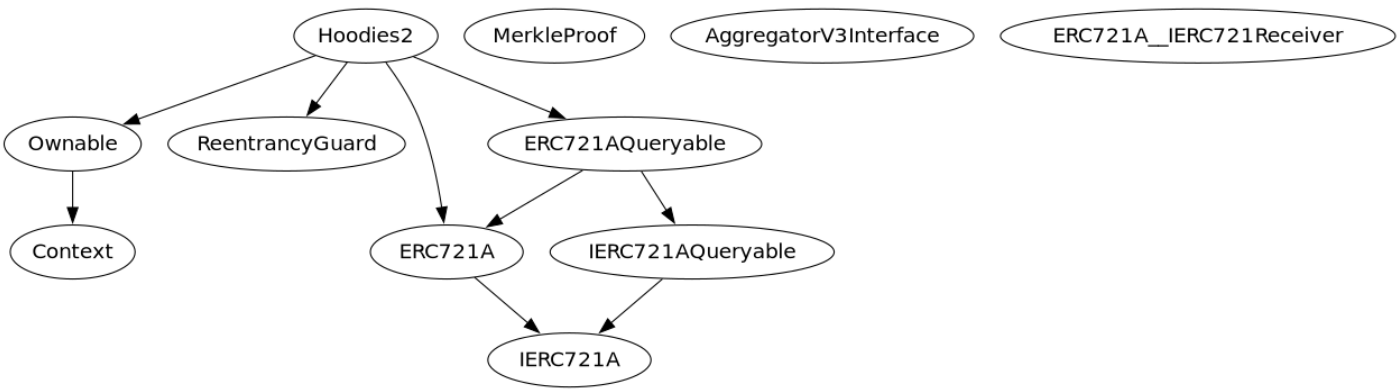
d914bfa0033befd13fba0bd04b46224da4a8d54cf234493299bff94210233e87

File: Hoodies.... | Language: solidity | Size: 3514 bytes | Date: 2023-03-25T10:28:55.716Z

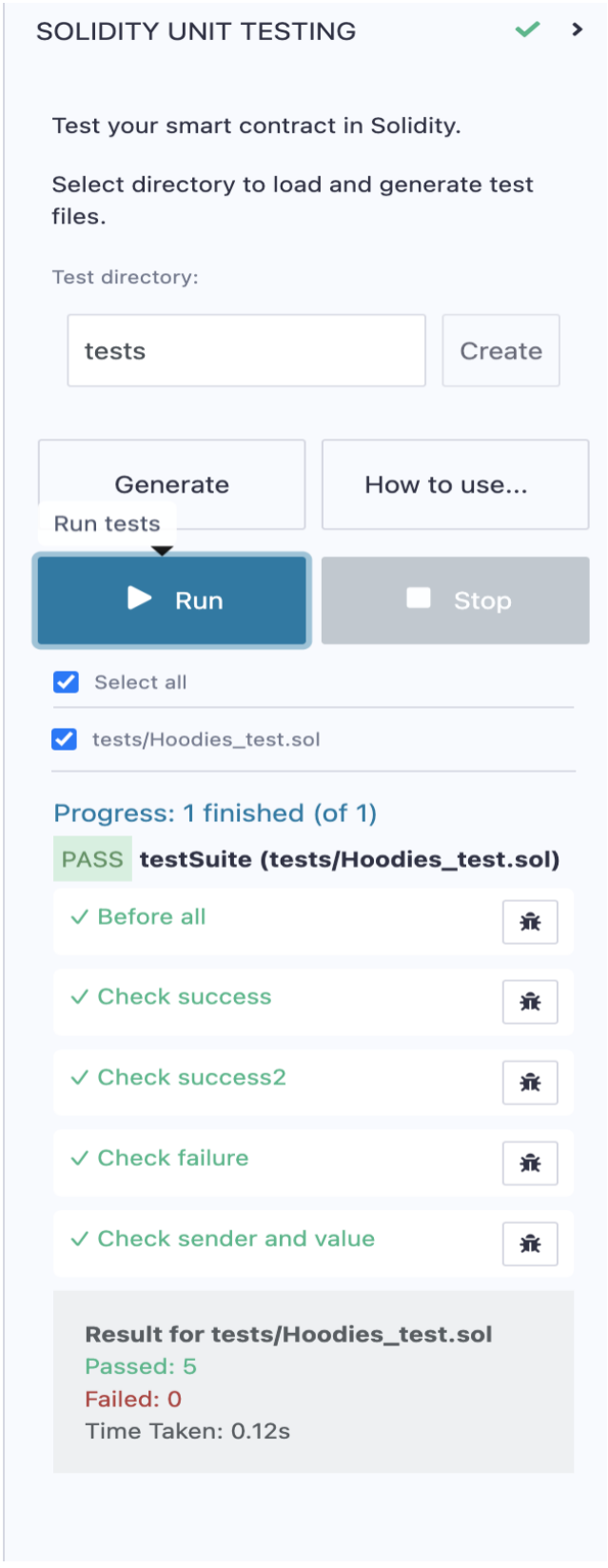
Critical	High	Medium	Low	Note
0	0	0	0	0

2- SOLIDITY STATIC ANALYSIS

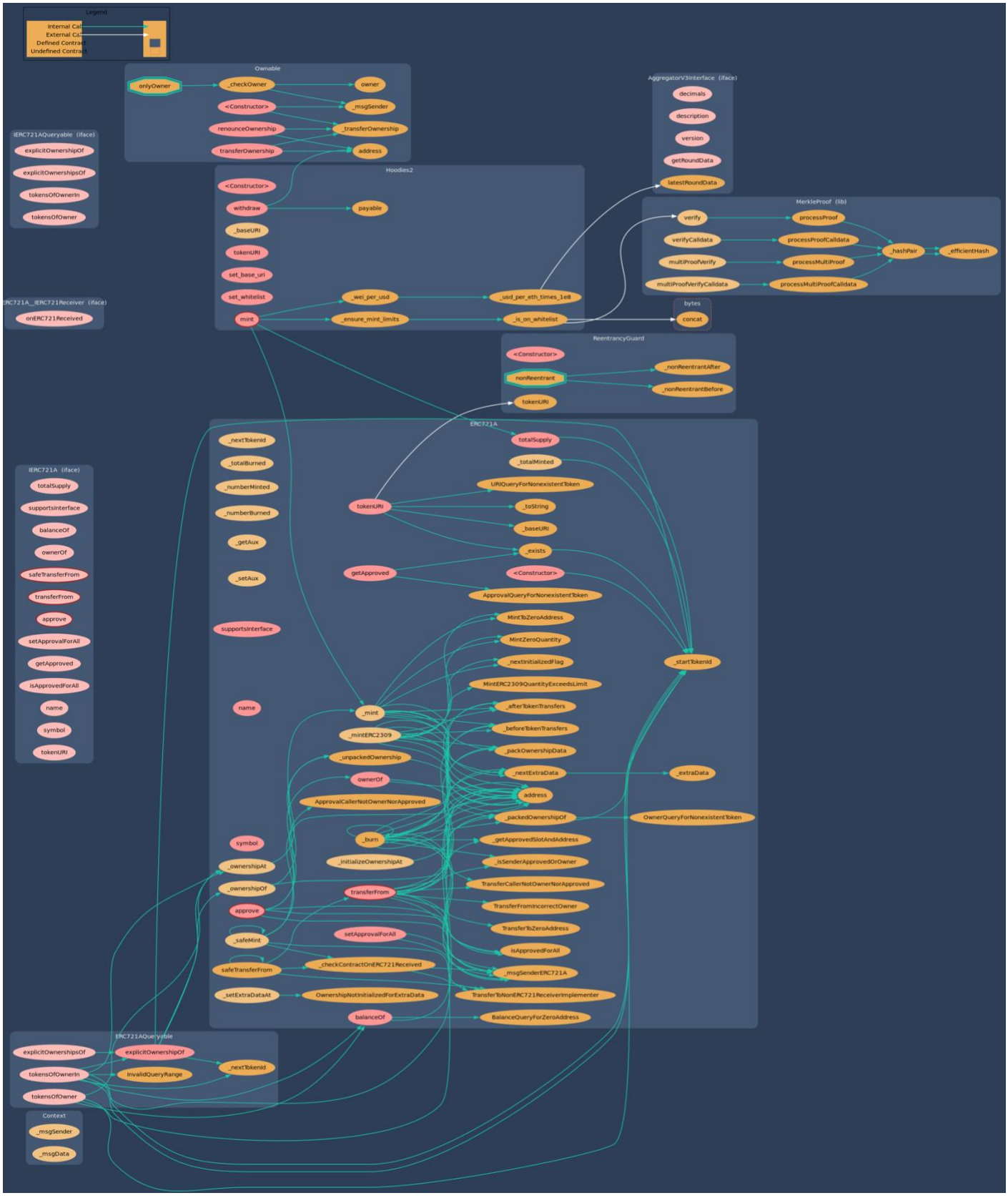
3- Inheritance graph



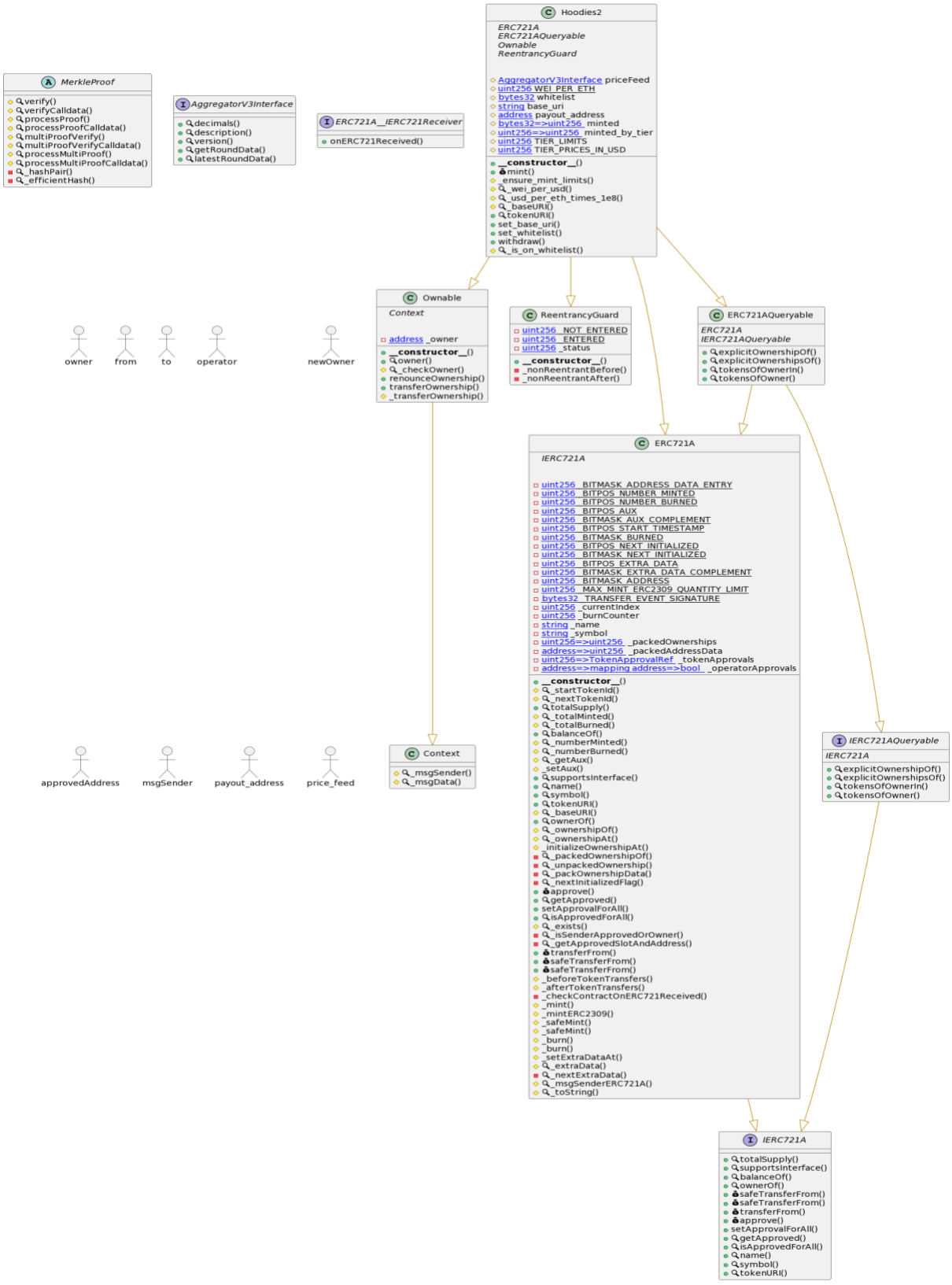
4- SOLIDITY UNIT TESTING



5- Call graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
Ownable
onlyOwner
_checkOwner
owner
<Constructor>
_msgSender
renounceOwnership
_transferOwnership
transferOwnership
address
IERC721AQueryable (iface)
explicitOwnershipOf
explicitOwnershipsOf
tokensOfOwnerIn
tokensOfOwner
Hoodies2
<Constructor>
withdraw
payable
_baseURI
tokenURI
set_base_uri
set_whitelist
mint
_wei_per_usd
_usd_per_eth_times_1e8
_ensure_mint_limits
_is_on_whitelist
AggregatorV3Interface (iface)
decimals
description
version
getRoundData
latestRoundData
MerkleProof (lib)
verify
processProof
verifyCalldata
processProofCalldata
multiProofVerify
processMultiProof
multiProofVerifyCalldata
processMultiProofCalldata
_hashPair
_efficientHash
bytes
concat
IERC721A__IERC721Receiver (iface)
onERC721Received
ReentrancyGuard
<Constructor>
nonReentrant
_nonReentrantAfter
_nonReentrantBefore
tokenURI
ERC721A
_nextTokenId
_totalBurned
_numberMinted
_numberBurned
_getAux
_setAux
supportsInterface
name
_mint
_mintERC2309
_unpackedOwnership
ownerOf
ApprovalCallerNotOwnerNorApproved
_burn
_initializeOwnershipAt
transferFrom
approve
_safeMint
safeTransferFrom
_setExtraDataAt
balanceOf
totalSupply
_totalMinted
URIQueryForNonexistentToken
tokenURI
_toString
_baseURI
_exists
getApproved
<Constructor>
ApprovalQueryForNonexistentToken
MintToZeroAddress
MintZeroQuantity
_nextInitializedFlag
MintERC2309QuantityExceedsLimit
_afterTokenTransfers
_beforeTokenTransfers
_packOwnershipData
_nextExtraData
address
_packedOwnershipOf
_getApprovedSlotAndAddress
_isSenderApprovedOrOwner
TransferCallerNotOwnerNorApproved
TransferFromIncorrectOwner
TransferToZeroAddress
setApprovalForAll
isApprovedForAll
_msgSenderERC721A
_checkContractOnERC721Received
OwnershipNotInitializedForExtraData
TransferToNonERC721ReceiverImplementer
BalanceQueryForZeroAddress
_startTokenId
_extraData
OwnerQueryForNonexistentToken
IERC721A (iface)
totalSupply
supportsInterface
balanceOf
ownerOf
safeTransferFrom
transferFrom
approve
setApprovalForAll
getApproved
isApprovedForAll
name
symbol
tokenURI
ERC721AQueryable
explicitOwnershipOf
explicitOwnershipsOf
tokensOfOwnerIn
tokensOfOwner
InvalidQueryRange
_nextTokenId
Context
_msgSender
_msgData

Unified Modeling Language (UML)



Functions signature

```
Sighash   |   Function Signature
========================
86333816  =>  multiProofVerify(bytes32[],bool[],bytes32,bytes32[])
119df25f  =>  _msgSender()
8b49d47e  =>  _msgData()
8da5cb5b  =>  owner()
53a72975  =>  _checkOwner()
715018a6  =>  renounceOwnership()
f2fde38b  =>  transferOwnership(address)
d29d44ee  =>  _transferOwnership(address)
5a9a49c7  =>  verify(bytes32[],bytes32,bytes32)
7ffe9d8c  =>  verifyCalldata(bytes32[],bytes32,bytes32)
62702a6b  =>  processProof(bytes32[],bytes32)
53b0f85f  =>  processProofCalldata(bytes32[],bytes32)
862aac96  =>  multiProofVerifyCalldata(bytes32[],bool[],bytes32,bytes32[])
ea5d3eb6  =>  processMultiProof(bytes32[],bool[],bytes32[])
32712d22  =>  processMultiProofCalldata(bytes32[],bool[],bytes32[])
00a4cf28  =>  _hashPair(bytes32,bytes32)
41ed615b  =>  _efficientHash(bytes32,bytes32)
62898eb8  =>  _nonReentrantBefore()
c7443eb2  =>  _nonReentrantAfter()
313ce567  =>  decimals()
7284e416  =>  description()
54fd4d50  =>  version()
9a6fc8f5  =>  getRoundData(uint80)
feaf968c  =>  latestRoundData()
18160ddd  =>  totalSupply()
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)
150b7a02  =>  onERC721Received(address,address,uint256,bytes)
98995f77  =>  _startTokenId()
4a60f620  =>  _nextTokenId()
736bf591  =>  _totalMinted()
fd01bd4c  =>  _totalBurned()
4d388a98  =>  _numberMinted(address)
6ba1b8d0  =>  _numberBurned(address)
f4a540c5  =>  _getAux(address)
4ff8c452  =>  _setAux(address,uint64)
743976a0  =>  _baseURI()
fb372cf2  =>  _ownershipOf(uint256)
cbaf28ce  =>  _ownershipAt(uint256)
f2d31624  =>  _initializeOwnershipAt(uint256)
```

```
444996c1  =>  _packedOwnershipOf(uint256)
4fe3c13e  =>  _unpackedOwnership(uint256)
bf460657  =>  _packOwnershipData(address,uint256)
e0e30f80  =>  _nextInitializedFlag(uint256)
f8e76cc0  =>  _exists(uint256)
848b8eac  =>  _isSenderApprovedOrOwner(address,address,address)
f112a2af  =>  _getApprovedSlotAndAddress(uint256)
ef435773  =>  _beforeTokenTransfers(address,address,uint256,uint256)
08c018f7  =>  _afterTokenTransfers(address,address,uint256,uint256)
d88343e2  =>  _checkContractOnERC721Received(address,address,uint256,bytes)
4e6ec247  =>  _mint(address,uint256)
4908d13b  =>  _mintERC2309(address,uint256)
6a4f832b  =>  _safeMint(address,uint256,bytes)
b3e1c718  =>  _safeMint(address,uint256)
9b1f9e74  =>  _burn(uint256)
834a9477  =>  _burn(uint256,bool)
bd3cdd6d  =>  _setExtraDataAt(uint256,uint24)
fc37bbd3  =>  _extraData(address,address,uint24)
5afe32e4  =>  _nextExtraData(address,address,uint256)
b60986df  =>  _msgSenderERC721A()
f832e238  =>  _toString(uint256)
c23dc68f  =>  explicitOwnershipOf(uint256)
5bbb2177  =>  explicitOwnershipsOf(uint256[])
99a2557a  =>  tokensOfOwnerIn(address,uint256,uint256)
8462151c  =>  tokensOfOwner(address)
0ca6fb20  =>  mint(bytes32[],uint256,uint256,uint256)
32d0cdd0  =>  _ensure_mint_limits(bytes32[],uint256,uint256,uint256)
2a152e7f  =>  _wei_per_usd()
4f71715e  =>  _usd_per_eth_times_1e8()
af1c5211  =>  set_base_uri(string)
a5799292  =>  set_whitelist(bytes32)
3ccfd60b  =>  withdraw()
f4eeb9a0  =>  _is_on_whitelist(bytes32[],uint256,uint256)
```

Automatic general report

Files Description Table

File Name	SHA-1 Hash
/Users/macbook/Desktop/smart contracts/Hoodies.sol	79929fa5ba58afd43eab8d71b69d4c8ac8ec119b

Contracts Description Table

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
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 🔒	🛑	
MerkleProof	Library			
└	verify	Internal 🔒		
└	verifyCalldata	Internal 🔒		
└	processProof	Internal 🔒		
└	processProofCalldata	Internal 🔒		
└	multiProofVerify	Internal 🔒		
└	multiProofVerifyCalldata	Internal 🔒		
└	processMultiProof	Internal 🔒		
└	processMultiProofCalldata	Internal 🔒		
└	_hashPair	Private 🔐		
└	_efficientHash	Private 🔐		
ReentrancyGuard	Implementation			
└	<Constructor>	Public ❗	🛑	NO❗
└	_nonReentrantBefore	Private 🔐	🛑	
└	_nonReentrantAfter	Private 🔐	🛑	
AggregatorV3Interface	Interface			
└	decimals	External ❗		NO❗
└	description	External ❗		NO❗
└	version	External ❗		NO❗
└	getRoundData	External ❗		NO❗

└	latestRoundData	External ❗		NO❗
IERC721A	Interface			
└	totalSupply	External ❗		NO❗
└	supportsInterface	External ❗		NO❗
└	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❗
└	name	External ❗		NO❗
└	symbol	External ❗		NO❗
└	tokenURI	External ❗		NO❗
ERC721A__IERC721Receiver	Interface			
└	onERC721Received	External ❗	🛑	NO❗
ERC721A	Implementation	IERC721A		
└	<Constructor>	Public ❗	🛑	NO❗
└	_startTokenId	Internal 🔒		
└	_nextTokenId	Internal 🔒		
└	totalSupply	Public ❗		NO❗
└	_totalMinted	Internal 🔒		
└	_totalBurned	Internal 🔒		
└	balanceOf	Public ❗		NO❗
└	_numberMinted	Internal 🔒		
└	_numberBurned	Internal 🔒		
└	_getAux	Internal 🔒		
└	_setAux	Internal 🔒	🛑	
└	supportsInterface	Public ❗		NO❗
└	name	Public ❗		NO❗
└	symbol	Public ❗		NO❗
└	tokenURI	Public ❗		NO❗
└	_baseURI	Internal 🔒		
└	ownerOf	Public ❗		NO❗
└	_ownershipOf	Internal 🔒		
└	_ownershipAt	Internal 🔒		
└	_initializeOwnershipAt	Internal 🔒	🛑	
└	_packedOwnershipOf	Private 🔐		
└	_unpackedOwnership	Private 🔐		
└	_packOwnershipData	Private 🔐		
└	_nextInitializedFlag	Private 🔐		
└	approve	Public ❗	💵	NO❗
└	getApproved	Public ❗		NO❗
└	setApprovalForAll	Public ❗	🛑	NO❗
└	isApprovedForAll	Public ❗		NO❗
└	_exists	Internal 🔒		
└	_isSenderApprovedOrOwner	Private 🔐		
└	_getApprovedSlotAndAddress	Private 🔐		
└	transferFrom	Public ❗	💵	NO❗
└	safeTransferFrom	Public ❗	💵	NO❗

└	safeTransferFrom	Public ❗	💵	NO❗
└	_beforeTokenTransfers	Internal 🔒	🛑	
└	_afterTokenTransfers	Internal 🔒	🛑	
└	_checkContractOnERC721Received	Private 🔐	🛑	
└	_mint	Internal 🔒	🛑	
└	_mintERC2309	Internal 🔒	🛑	
└	_safeMint	Internal 🔒	🛑	
└	_safeMint	Internal 🔒	🛑	
└	_burn	Internal 🔒	🛑	
└	_burn	Internal 🔒	🛑	
└	_setExtraDataAt	Internal 🔒	🛑	
└	_extraData	Internal 🔒		
└	_nextExtraData	Private 🔐		
└	_msgSenderERC721A	Internal 🔒		
└	_toString	Internal 🔒		
IERC721AQueryable	Interface	IERC721A		
└	explicitOwnershipOf	External ❗		NO❗
└	explicitOwnershipsOf	External ❗		NO❗
└	tokensOfOwnerIn	External ❗		NO❗
└	tokensOfOwner	External ❗		NO❗
ERC721AQueryable	Implementation	ERC721A, IERC721AQueryable		
└	explicitOwnershipOf	Public ❗		NO❗
└	explicitOwnershipsOf	External ❗		NO❗
└	tokensOfOwnerIn	External ❗		NO❗
└	tokensOfOwner	External ❗		NO❗
Hoodies2	Implementation	ERC721A, ERC721AQueryable, Ownable, ReentrancyGuard		
└	<Constructor>	Public ❗	🛑	ERC721A
└	mint	Public ❗	💵	nonReentrant
└	_ensure_mint_limits	Internal 🔒	🛑	
└	_wei_per_usd	Internal 🔒		
└	_usd_per_eth_times_1e8	Internal 🔒		
└	_baseURI	Internal 🔒		
└	tokenURI	Public ❗		NO❗
└	set_base_uri	Public ❗	🛑	onlyOwner
└	set_whitelist	Public ❗	🛑	onlyOwner
└	withdraw	Public ❗	🛑	onlyOwner
└	_is_on_whitelist	Internal 🔒		

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.

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.