Smart Contract
Security Audit
V1

MyToken Smart Contract

21/1/2022

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**: Polygon
- **Contract Address**: 0x696EAed50c129ed08D125Ead47c4CDF1a466da7b
- **Code:**

https://mumbai.polygonscan.com/address/0xd4c8581265e0b3b7f2a59a8740cffb632d4be06c#code

NFT Information

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

Contracts address deployed to test net (Polygonscan)

MyToken Smart contract on Mumbai test net.

https://mumbai.polygonscan.com/address/0x696eaed50c129ed08d125ead47c4cdf1a466da7b

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 0 notes in all solidity files of the contract

The files:

MyToken.sol

File and Function Level Report

File in Scope:

Contract Name	SHA 256 hash	Contract Address
MyToken.sol	**dca899a93c10d8302e4ed5603bbbaa8692bec1fc6340c5970aa9fd348334a423**	0x696EAed50c129ed08D125Ead47c4CDF1a466da7b

- Contract: MyToken
- Inherit: Initializable, ERC721Upgradeable, ERC721EnumerableUpgradeable, PausableUpgradeable, OwnableUpgradeable, ERC721BurnableUpgradeable, UUPSUpgradeable, ContextMixin
- 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**
isWhitListed	✓	Read / public	**Passed**
supportsInterface	✓	Read / public	**Passed**
price	✓	Read / public	**Passed**
balanceOf	✓	Read / public	**Passed**
Owner	✓	Read / public	**Passed**
baseTokenURI	✓	Read / public	**Passed**
minting	✓	Read / public	**Passed**
getApprovedForAll	✓	Read / public	**Passed**
paused	✓	Read / public	**Passed**
getApproved	✓	Read / public	**Passed**

ownerOf	✓	Read / public	**Passed**
tokenURI	✓	Read / public	**Passed**
tokenByIndex	✓	Read / public	**Passed**
tokenOfOwnerByIndex	✓	Read / public	**Passed**
MAX_PER_MINT	✓	Read / public	**Passed**
max_Supply	✓	Read / public	**Passed**
totalSupply	✓	Read / public	**Passed**
whiteList	✓	Read / public	**Passed**
setMaxSupply	✓	Write / public	**Passed**
approve	✓	Write / public	**Passed**
safeTransferFrom	✓	Write / public	**Passed**
safeTransferFrom	✓	Write / public	**Passed**
setBaseURI	✓	Write / public	**Passed**
paused	✓	Write / public	**Passed**
mintNFT	✓	Write / payable	**Passed**
setPrice	✓	Write / public	**Passed**
transferOwnership	✓	Write / public	**Passed**
setApprovalForAll	✓	Write / public	**Passed**
transferFrom	✓	Write / public	**Passed**
unPaused	✓	Write / public	**Passed**
withdrawAll	✓	Write / payable	**Passed**
setMinting	✓	Write / public	**Passed**
renounceOwnership	✓	Write / public	**Passed**
burn	✓	Write / public	**Passed**
initialize	✓	Write / public	**Passed**
mintForOwner	✓	Write / public	**Passed**
safeMint	✓	Write / public	**Passed**

upgradeTo	✓	Write / payable	**Passed**
setWhiteList	✓	Write / public	**Passed**
upgradeToAndCall	✓	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	Front running.	**Passed**
6	Timestamp dependence.	**Passed**
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**
18	Design Logic.	**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 0.8.7 instead of ^0.8.7). 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:

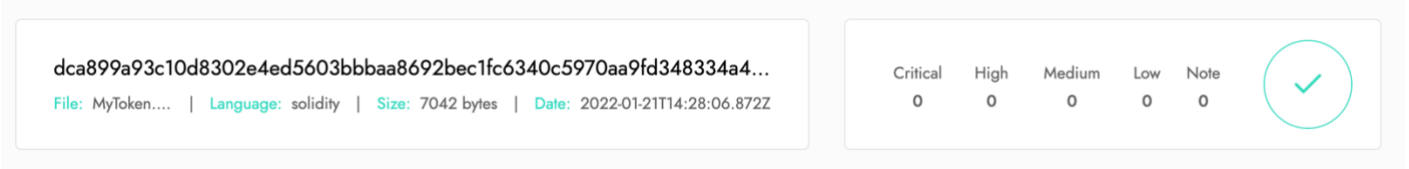
No Very Low severity vulnerabilities were found.

Notes:

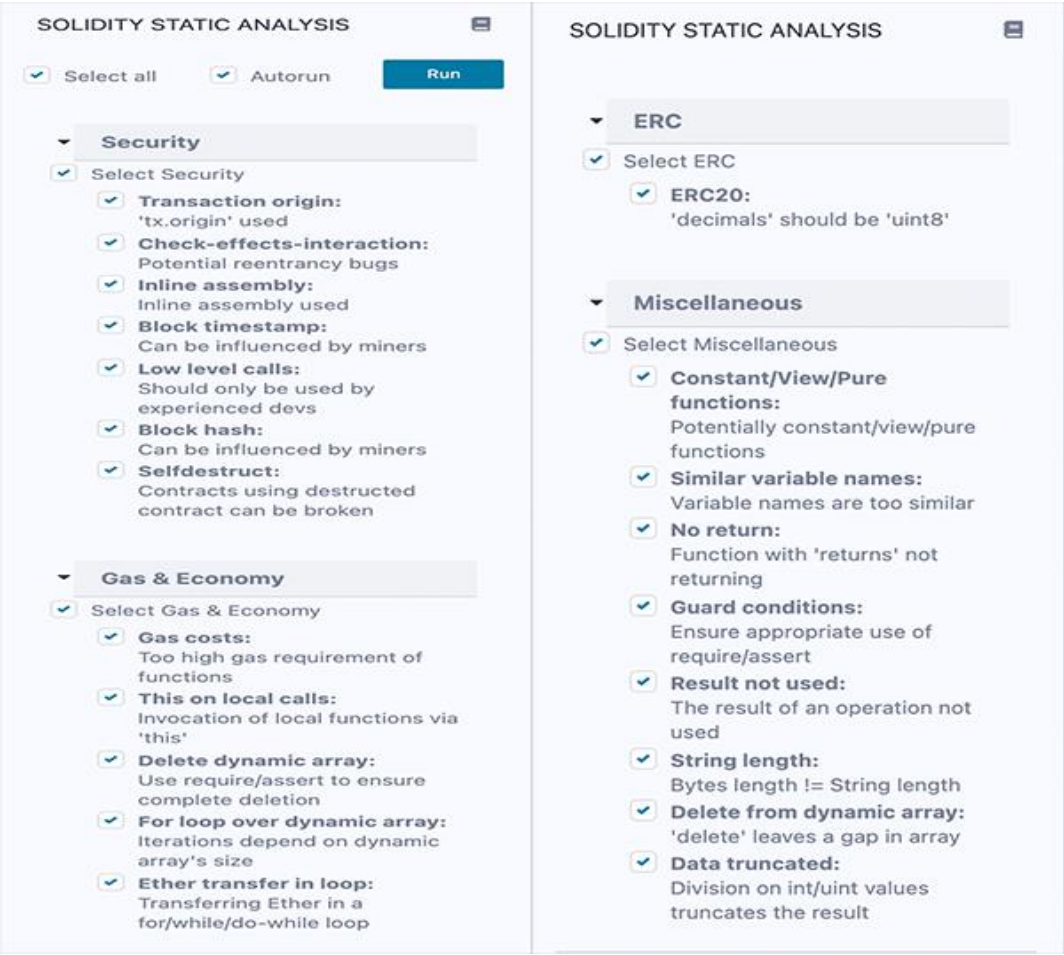
No Notes were found

Automatic Testing

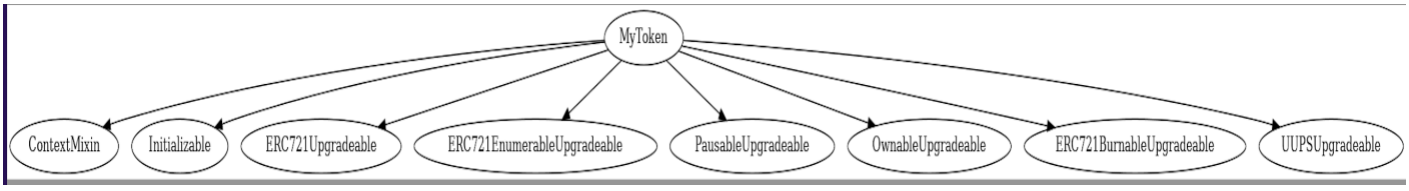
1- Check for security



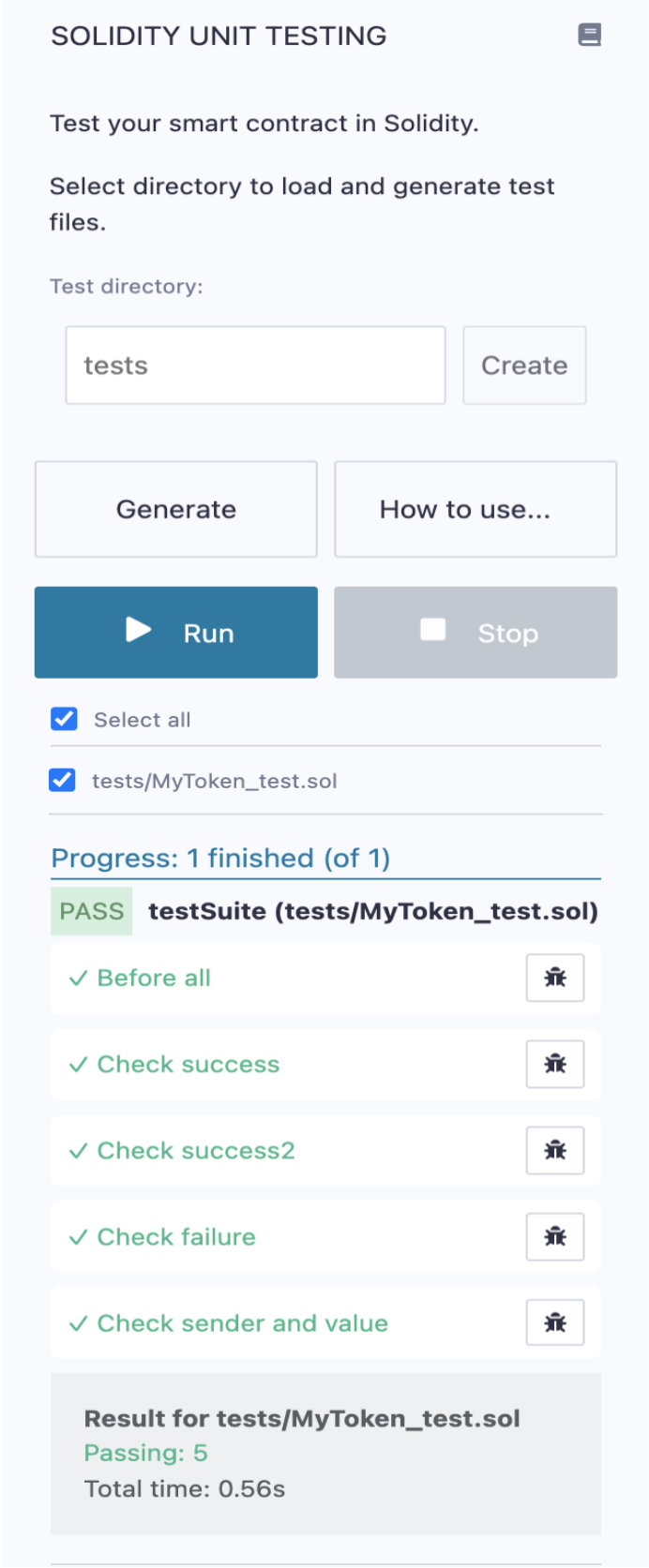
2- SOLIDITY STATIC ANALYSIS



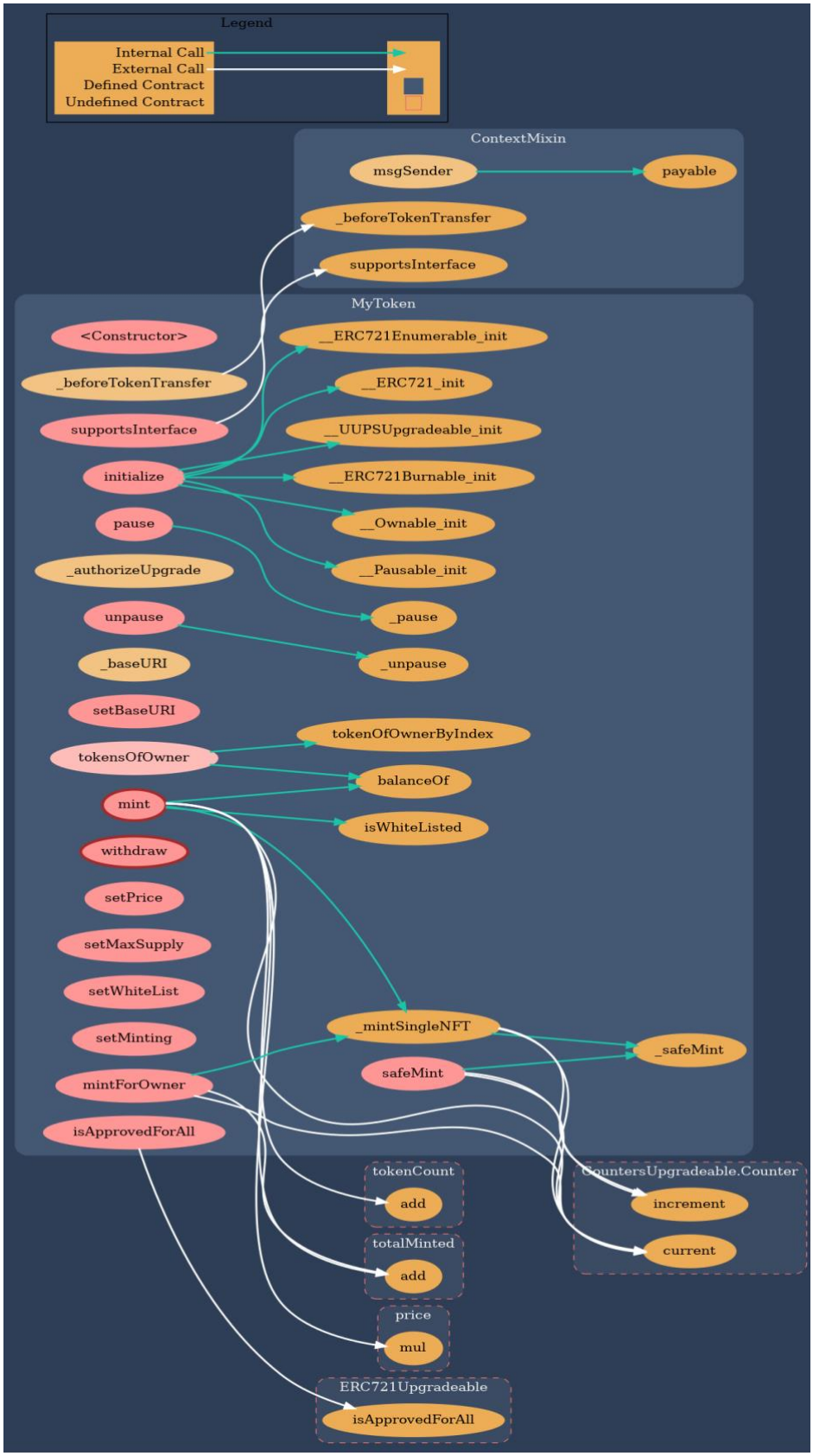
3- Inheritance graph



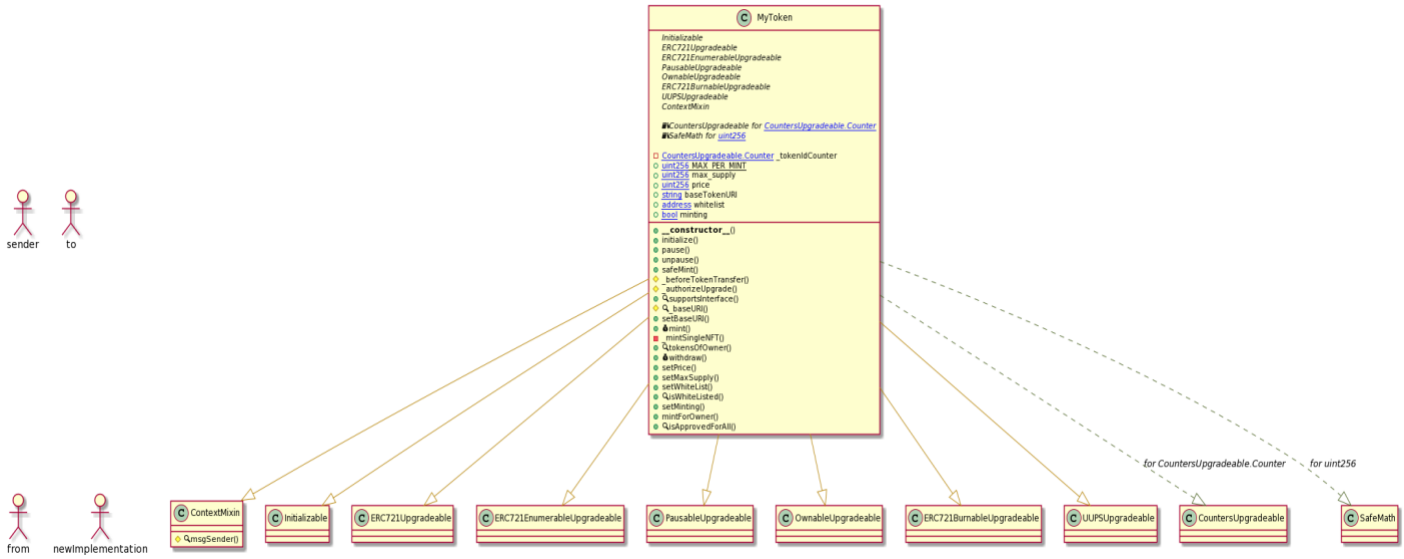
4- SOLIDITY UNIT TESTING



5- Call graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
ContextMixin
msgSender
payable
_beforeTokenTransfer
supportsInterface
MyToken
<Constructor>
_beforeTokenTransfer
supportsInterface
initialize
pause
_authorizeUpgrade
unpause
_baseURI
setBaseURI
tokensOfOwner
mint
withdraw
setPrice
setMaxSupply
setWhiteList
setMinting
mintForOwner
isApprovedForAll
__ERC721Enumerable_init
__ERC721_init
__UUPSUpgradeable_init
__ERC721Burnable_init
__Ownable_init
__Pausable_init
_pause
_unpause
tokenOfOwnerByIndex
balanceOf
isWhiteListed
_mintSingleNFT
safeMint
_safeMint
tokenCount
add
totalMinted
add
price
mul
CountersUpgradeable.Counter
increment
current
ERC721Upgradeable
isApprovedForAll

Unified Modeling Language (UML)



Functions signature

```
Sighash   |   Function Signature
========================
d737d0c7  =>  msgSender()
8129fc1c  =>  initialize()
8456cb59  =>  pause()
3f4ba83a  =>  unpause()
40d097c3  =>  safeMint(address)
cad3be83  =>  _beforeTokenTransfer(address,address,uint256)
5ec29272  =>  _authorizeUpgrade(address)
01ffc9a7  =>  supportsInterface(bytes4)
743976a0  =>  _baseURI()
55f804b3  =>  setBaseURI(string)
a0712d68  =>  mint(uint256)
825afd56  =>  _mintSingleNFT()
8462151c  =>  tokensOfOwner(address)
3ccfd60b  =>  withdraw()
91b7f5ed  =>  setPrice(uint256)
6f8b44b0  =>  setMaxSupply(uint256)
775b9c13  =>  setWhiteList(address[])
6f9170f6  =>  isWhiteListed(address)
b4c7f066  =>  setMinting(bool)
63bc312a  =>  mintForOwner(uint256)
e985e9c5  =>  isApprovedForAll(address,address)
```

Automatic general report

Files Description Table

File Name	SHA-1 Hash
/Users/macbook/Desktop/smart contracts/MyToken.sol	f7e424945bf61caabcb1bf98b922351c6a49b8e7
/Users/macbook/Desktop/smart contracts/GivingBulls.sol	5ac5678c14862fa7a143a44eba17307b053c2556

Contracts Description Table

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
ContextMixin	Implementation			
└	msgSender	Internal 🔒		
MyToken	Implementation	Initializable, ERC721Upgradeable, ERC721EnumerableUpgradeable, PausableUpgradeable, OwnableUpgradeable, ERC721BurnableUpgradeable, UUPSUpgradeable, ContextMixin		
└	<Constructor>	Public ❗	🛑	initializer
└	initialize	Public ❗	🛑	initializer
└	pause	Public ❗	🛑	onlyOwner
└	unpause	Public ❗	🛑	onlyOwner
└	safeMint	Public ❗	🛑	onlyOwner
└	_beforeTokenTransfer	Internal 🔒	🛑	whenNotPaused
└	_authorizeUpgrade	Internal 🔒	🛑	onlyOwner
└	supportsInterface	Public ❗		NO❗
└	_baseURI	Internal 🔒		
└	setBaseURI	Public ❗	🛑	onlyOwner
└	mint	Public ❗	💰	whenNotPaused
└	_mintSingleNFT	Private 🔐	🛑	
└	tokensOfOwner	External ❗		NO❗
└	withdraw	Public ❗	💰	onlyOwner
└	setPrice	Public ❗	🛑	onlyOwner
└	setMaxSupply	Public ❗	🛑	onlyOwner
└	setWhiteList	Public ❗	🛑	onlyOwner
└	isWhiteListed	Public ❗		NO❗
└	setMinting	Public ❗	🛑	onlyOwner
└	mintForOwner	Public ❗	🛑	onlyOwner
└	isApprovedForAll	Public ❗		NO❗
GivingBulls	Implementation	ERC721EnumerableLite, Ownable		
└	<Constructor>	Public ❗	🛑	ERC721B
└	tokenURI	External ❗		NO❗
└	setBaseURI	Public ❗	🛑	onlyOwner
└	flipSaleState	Public ❗	🛑	onlyOwner
└	flipPresaleState	Public ❗	🛑	onlyOwner

└	mintBull	Public ❗	💵	NO❗
└	addUserToPresale	Public ❗	🛑	onlyOwner
└	addUsersToPresale	Public ❗	🛑	onlyOwner
└	isUserInPresale	Public ❗		NO❗
└	setPrice	Public ❗	🛑	onlyOwner
└	withdrawAllToAddress	Public ❗	💵	onlyOwner

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

Conclusion

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

Since possible test cases can be unlimited and developer level documentation (code flow diagram with function level description) not provided, for such an extensive smart contract protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan Everything.

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

✓ No volatile code.
✓ Not many 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.