Smart Contract Security Audit V1

NFTME Token

https://www.nftme.com.co/

6/1/2022

https://saferico.com/

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

Table of Contents

Table of Contents

Background

Project Information

NFTME Token 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

- **Website**: https://www.nftme.com.co/
- **Telegram :** https://t.me/+xydoejEi2S84Yzgx
- **Twitter**: https://twitter.com/The_NFTME_Token
- **WhitePaper**: https://www.instagram.com/the_nftme
- **Discord**: https://discord.gg/P3mmNnFsE7
- **Platform**: Ethereum

Token Information

- Name: NFTME
- Total Supply: 2,222,222,222
- Holders: address
- Total transactions:

Contracts address deployed to test net (ETH)

NFTME Token Smart contract on rinkeby (ETH Test Net)
https://rinkeby.etherscan.io/address/0x353d1d33eacb6ab73da2e87af313abb4b608d3e1

Executive Summary

According to our assessment, the customer`s solidity smart contract is **Secured**.because most of low issues and notes fixed in version 2.

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, 3 low, very low-level issues and 3 notes in all solidity files of the contract

The files:

NFTME.sol

File and Function Level Report

File in Scope:

Contract Name	SHA 256 hash	Contract Address
NFTME.sol	**0ea1cd09b5557a17e8d0fa2821b71454bb020a532de70b7a57823ec814b56407**	0x353d1d33EAcb6ab73da2e87AF313ABb4B608d3e1

- Contract: NFTME
- Inherit: Context, IERC20, 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**
decimals	✔	Read / public	**Passed**
totalSupply	✔	Read / public	**Passed**
allowance	✔	Read / public	**Passed**
balanceOf	✔	Read / public	**Passed**
Owner	✔	Read / public	**Passed**
geUnlockTime	✔	Read / public	**Passed**
swapAndLiquifyEnabled	✔	Read / public	**Passed**
rewards	✔	Read / public	**Passed**
reflectionFromToken	✔	Read / public	**Passed**
isExcludedFromReward	✔	Read / public	**Passed**

tokenFromReflection	✔	Read / public	**Passed**
_maxTxAmount	✔	Read / private	**Passed**
_taxFee	✔	Read / private	**Passed**
isExcludedFromFees	✔	Read / public	**Passed**
_liquidityFee	✔	Read / private	**Passed**
_marketingFee	✔	Read / private	**Passed**
totalFees	✔	Read / public	**Passed**
uniswapV2Pair	✔	Read / public	**Passed**
uniswapV2Router	✔	Read / public	**Passed**
CommunityWallet	✔	Read / public	**Passed**
unLock	✔	Write / public	**Passed**
approve	✔	Write / public	**Passed**
transferFrom	✔	Write / public	**Passed**
transfer	✔	Write / public	**Passed**
deliver	✔	Write / public	**Passed**
excludeFromFees	✔	Write / public	**Passed**
excludeFromReward	✔	Write / public	**Passed**
includeInFee	✔	Write / public	**Passed**
renounceOwnership	✔	Write / public	**Passed**
transferOwnership	✔	Write / public	**Passed**
includeInReward	✔	Write / public	**Passed**
setLiquiditFeePercent	✔	Write / public	**Passed**
decreaseAllowance	✔	Write / public	**Passed**
setTaxFeePercent	✔	Write / public	**Passed**
lock	✔	Write / public	**Passed**
setMaxAmount	✔	Write / public	**Passed**
setMaxTxAmount	✔	Write / public	**Passed**
setSwapAndLiquifyEnabled	✔	Write / public	**Passed**

increaseAllowance	✔	Write / public	**Passed**
setCommunityWallet	✔	Write / public	**Passed**
setCommunityFees	✔	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:

#Use of block.timestamp for comparisons
Description
The value of block.timestamp can be manipulated by the miner.
And conditions with strict equality is difficult to achieve - block.timestamp

Remediation
Avoid use of block.timestamp

Status: Acknowledged

#Owner privileges (In the period when the owner isn't renounced)
Description
Owner can change tax, burn, market, community and liquidity fees.
Owner can enable and disable the trading.
Owner can lock the contract for 7 days.
Owner can exclude and include any address from fees and rewards

```
function excludeFromFee(address account) public onlyOwner {
        _isExcludedFromFee[account] = true;
    }

    function includeInFee(address account) public onlyOwner {
        _isExcludedFromFee[account] = false;
    }


    function setMaketingWallet(address newWallet) external onlyOwner() {
        CommunityWallet = newWallet;
    }

    function setMaxTxPercent(uint256 maxTxPercent) external onlyOwner() {
```

```
        maxTxAmount = _tTotal.mul(maxTxPercent).div(
            10**2
        );
    }

    function setliquidityFee(uint256 liqFee) external onlyOwner() {
        liquidityFee = _tTotal.mul(liqFee).div(
            10**2
        );
    }

    function setCommunityFee(uint256 CommunityFee) external onlyOwner() {
        fundingFee = _tTotal.mul(CommunityFee).div(
            10**2
        );
    }

    function setCommunityWallet(address newAdd) external onlyOwner() {
       CommunityWallet = newAdd;
    }

    function setBurnFee(uint256 BurnFee) external onlyOwner() {
        burnFee = _tTotal.mul(BurnFee).div(
            10**2
        );
    }
    function setSwapAndLiquifyEnabled(bool _enabled) public onlyOwner {
        swapAndLiquifyEnabled = _enabled;
        emit SwapAndLiquifyEnabledUpdated(_enabled);
    }
```

Remediation

Make these functions internal in next version or the team should announce the investors before change the fees and give them time if they want to use the old fees.

Status: Acknowledged

#Pragam version not fixed

Description

It is a good practice to lock the solidity version for a live deployment (use 0.6.12 instead of ^0.6.12). 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

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 "__maxTxAmount" which is public

Remediation

Remove "_" from external variable names and add it to private variable names
Status: Closed. fixed in version 2

 # Constant calculations in the contract

Description

recalculated initialization will save 2847 units of gas in deployment

```
uint256 private _tTotal = 2222222222 * 10**18;
uint256 public _maxTxAmount = 222222222 * 10**18;
uint256 private numTokensSellToAddToLiquidity = 22222222 * 10**18;
```

Recommendation

Replace the initialization as

```
uint256 private _tTotal = 22222222220000000000000000000;
uint256 public _maxTxAmount = 2222222220000000000000000000;
uint256 private numTokensSellToAddToLiquidity = 222222222000000000000000000;
```

Status: Closed. fixed in version 2

#Compiler version is old

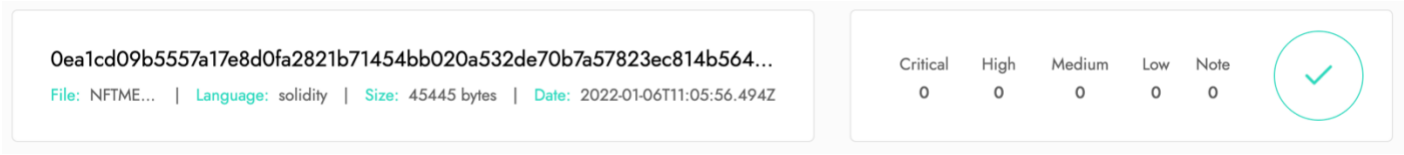
Description

The compiler being used was released a 3 years ago. It's recommended to use more recent compiler version, there can be benefits like reduction in bytecode size etc.

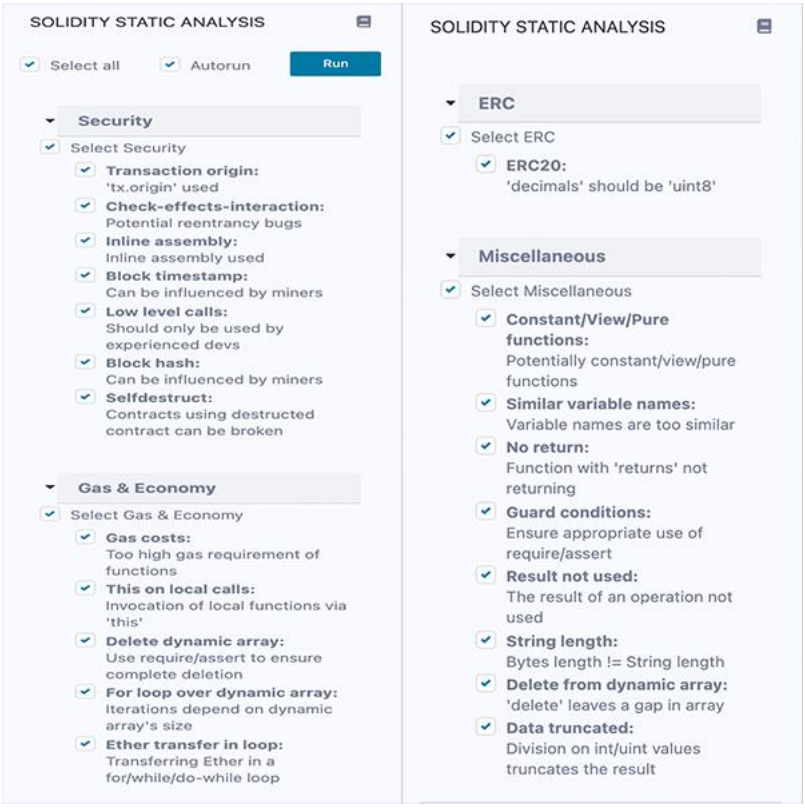
Status: Acknowledged

Automatic Testing

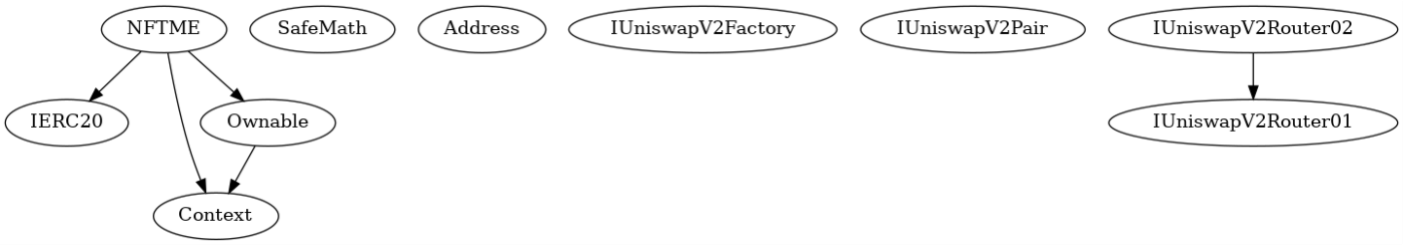
1- Check for security



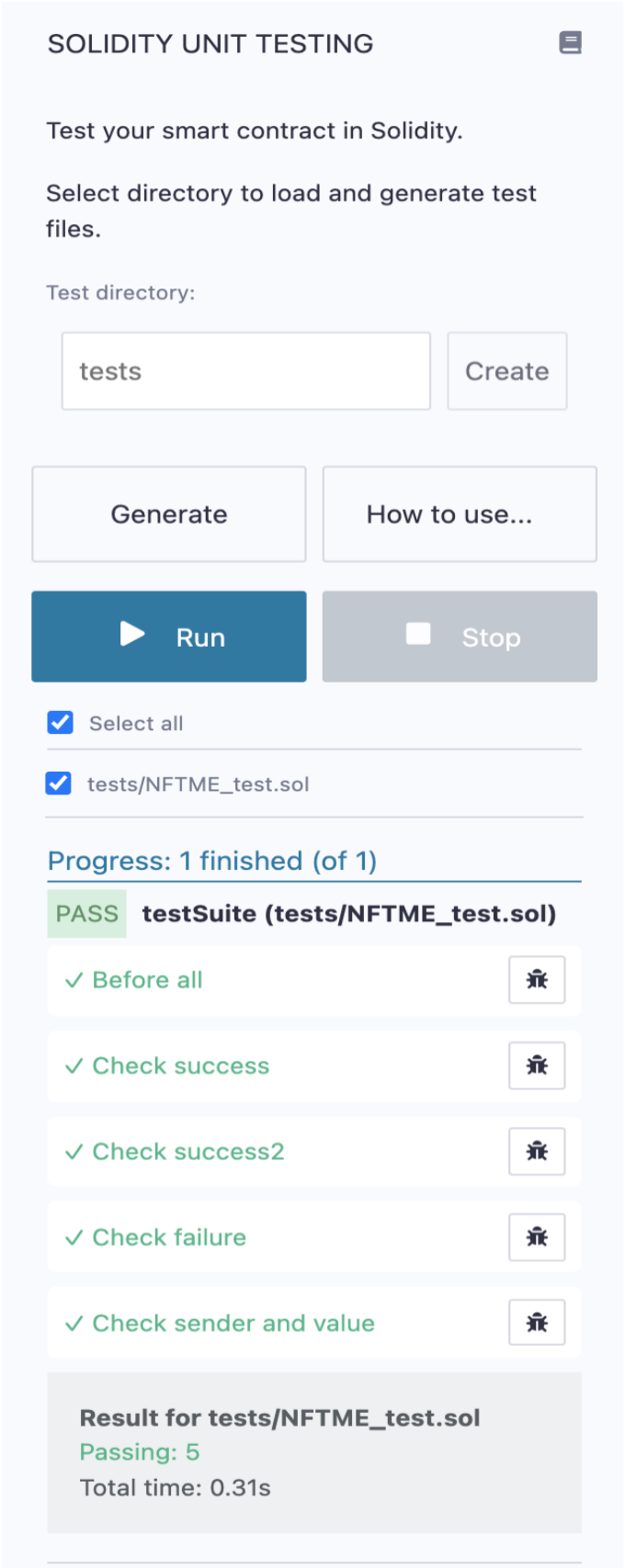
2- SOLIDITY STATIC ANALYSIS



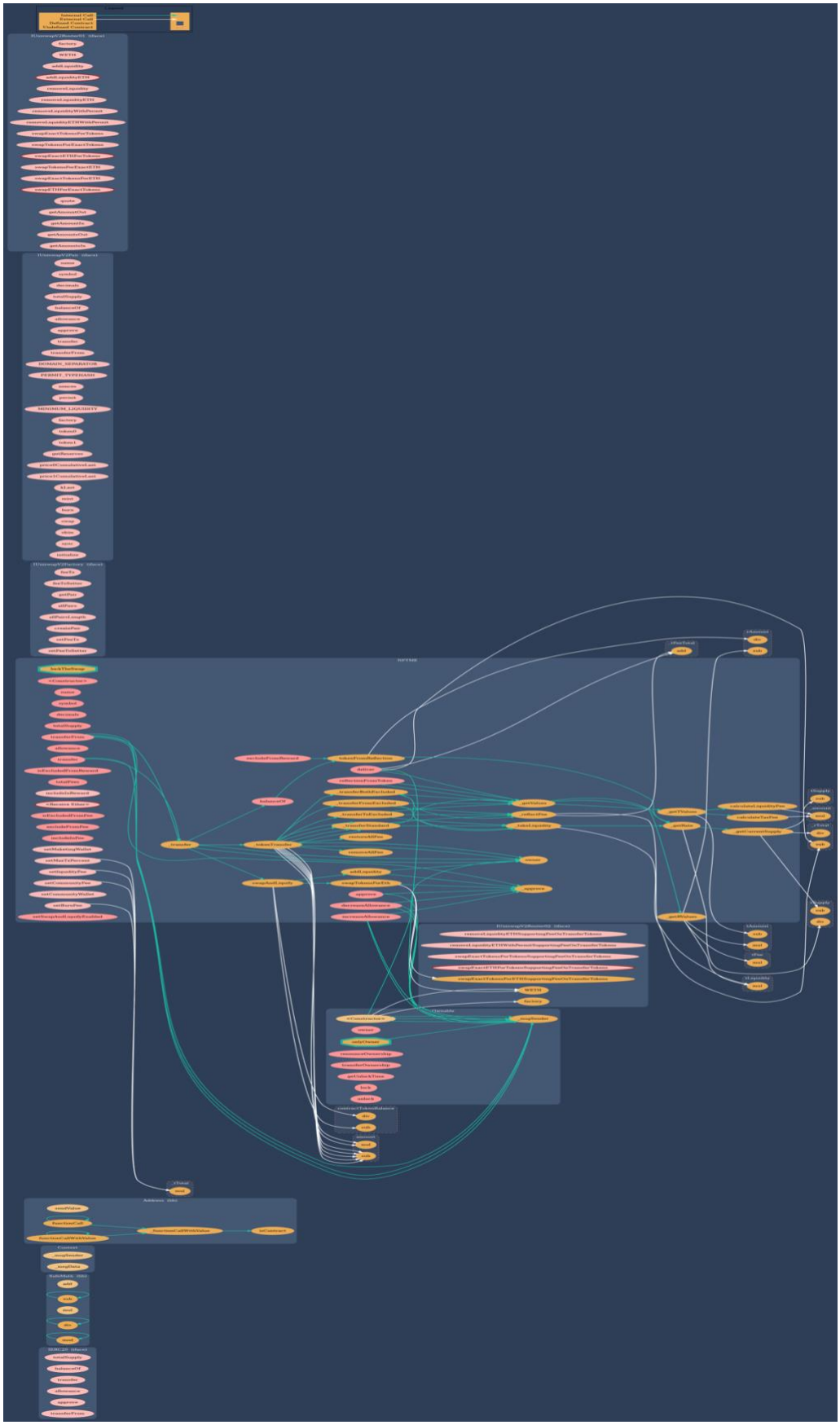
3- Inheritance graph



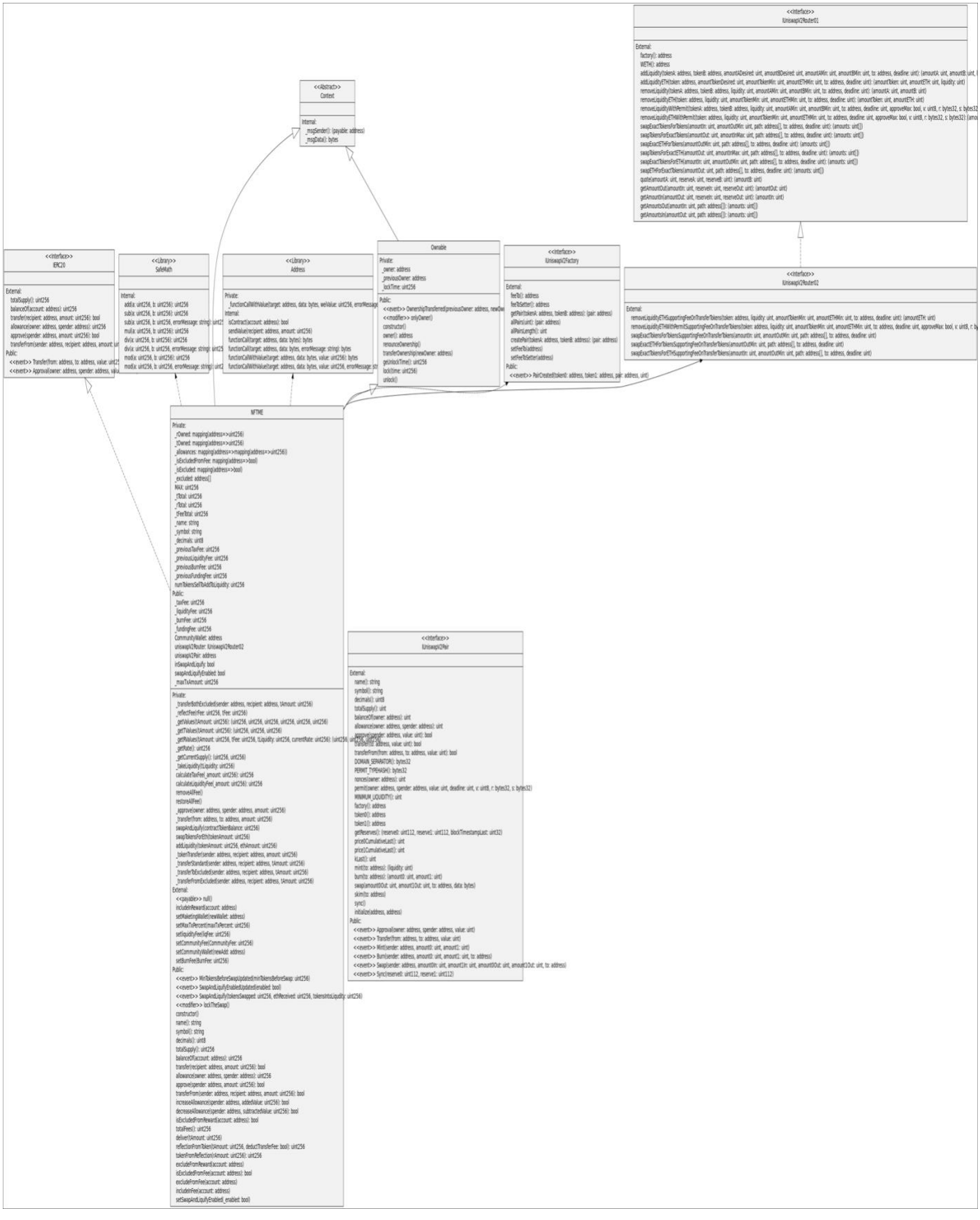
4- SOLIDITY UNIT TESTING



5- Call graph



Unified Modeling Language (UML)



Functions signature

```
11902160  =>  _getTValues(uint256)
16279055  =>  isContract(address)
39509351  =>  increaseAllowance(address,uint256)
75128141  =>  calculateTaxFee(uint256)
18160ddd  =>  totalSupply()
70a08231  =>  balanceOf(address)
a9059cbb  =>  transfer(address,uint256)
dd62ed3e  =>  allowance(address,address)
095ea7b3  =>  approve(address,uint256)
23b872dd  =>  transferFrom(address,address,uint256)
771602f7  =>  add(uint256,uint256)
b67d77c5  =>  sub(uint256,uint256)
e31bdc0a  =>  sub(uint256,uint256,string)
c8a4ac9c  =>  mul(uint256,uint256)
a391c15b  =>  div(uint256,uint256)
b745d336  =>  div(uint256,uint256,string)
f43f523a  =>  mod(uint256,uint256)
71af23e8  =>  mod(uint256,uint256,string)
119df25f  =>  _msgSender()
8b49d47e  =>  _msgData()
24a084df  =>  sendValue(address,uint256)
a0b5ffb0  =>  functionCall(address,bytes)
241b5886  =>  functionCall(address,bytes,string)
2a011594  =>  functionCallWithValue(address,bytes,uint256)
d525ab8a  =>  functionCallWithValue(address,bytes,uint256,string)
36455e42  =>  _functionCallWithValue(address,bytes,uint256,string)
8da5cb5b  =>  owner()
715018a6  =>  renounceOwnership()
f2fde38b  =>  transferOwnership(address)
b6c52324  =>  geUnlockTime()
dd467064  =>  lock(uint256)
a69df4b5  =>  unlock()
017e7e58  =>  feeTo()
094b7415  =>  feeToSetter()
e6a43905  =>  getPair(address,address)
1e3dd18b  =>  allPairs(uint256)
574f2ba3  =>  allPairsLength()
c9c65396  =>  createPair(address,address)
f46901ed  =>  setFeeTo(address)
a2e74af6  =>  setFeeToSetter(address)
06fdde03  =>  name()
95d89b41  =>  symbol()
313ce567  =>  decimals()
3644e515  =>  DOMAIN_SEPARATOR()
30adf81f  =>  PERMIT_TYPEHASH()
7ecebe00  =>  nonces(address)
d505accf  =>  permit(address,address,uint256,uint256,uint8,bytes32,bytes32)
ba9a7a56  =>  MINIMUM_LIQUIDITY()
c45a0155  =>  factory()
0dfe1681  =>  token0()
d21220a7  =>  token1()
0902f1ac  =>  getReserves()
5909c0d5  =>  price0CumulativeLast()
5a3d5493  =>  price1CumulativeLast()
7464fc3d  =>  kLast()
6a627842  =>  mint(address)
```

```
89afcb44  =>  burn(address)
022c0d9f  =>  swap(uint256,uint256,address,bytes)
bc25cf77  =>  skim(address)
fff6cae9  =>  sync()
485cc955  =>  initialize(address,address)
ad5c4648  =>  WETH()
e8e33700  =>
addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256)
f305d719  =>  addLiquidityETH(address,uint256,uint256,uint256,address,uint256)
baa2abde  =>
removeLiquidity(address,address,uint256,uint256,uint256,address,uint256)
02751cec  =>  removeLiquidityETH(address,uint256,uint256,uint256,address,uint256)
2195995c  =>
removeLiquidityWithPermit(address,address,uint256,uint256,uint256,address,uint256,b
ool,uint8,bytes32,bytes32)
ded9382a  =>
removeLiquidityETHWithPermit(address,uint256,uint256,uint256,address,uint256,bool,u
int8,bytes32,bytes32)
38ed1739  =>  swapExactTokensForTokens(uint256,uint256,address[],address,uint256)
8803dbee  =>  swapTokensForExactTokens(uint256,uint256,address[],address,uint256)
7ff36ab5  =>  swapExactETHForTokens(uint256,address[],address,uint256)
4a25d94a  =>  swapTokensForExactETH(uint256,uint256,address[],address,uint256)
18cbafe5  =>  swapExactTokensForETH(uint256,uint256,address[],address,uint256)
fb3bdb41  =>  swapETHForExactTokens(uint256,address[],address,uint256)
ad615dec  =>  quote(uint256,uint256,uint256)
054d50d4  =>  getAmountOut(uint256,uint256,uint256)
85f8c259  =>  getAmountIn(uint256,uint256,uint256)
d06ca61f  =>  getAmountsOut(uint256,address[])
1f00ca74  =>  getAmountsIn(uint256,address[])
af2979eb  =>
removeLiquidityETHSupportingFeeOnTransferTokens(address,uint256,uint256,uint256,add
ress,uint256)
5b0d5984  =>
removeLiquidityETHWithPermitSupportingFeeOnTransferTokens(address,uint256,uint256,u
int256,address,uint256,bool,uint8,bytes32,bytes32)
5c11d795  =>
swapExactTokensForTokensSupportingFeeOnTransferTokens(uint256,uint256,address[],add
ress,uint256)
b6f9de95  =>
swapExactETHForTokensSupportingFeeOnTransferTokens(uint256,address[],address,uint25
6)
791ac947  =>
swapExactTokensForETHSupportingFeeOnTransferTokens(uint256,uint256,address[],addres
s,uint256)
a457c2d7  =>  decreaseAllowance(address,uint256)
88f82020  =>  isExcludedFromReward(address)
13114a9d  =>  totalFees()
3bd5d173  =>  deliver(uint256)
4549b039  =>  reflectionFromToken(uint256,bool)
2d838119  =>  tokenFromReflection(uint256)
52390c02  =>  excludeFromReward(address)
3685d419  =>  includeInReward(address)
6ff6cdf4  =>  _transferBothExcluded(address,address,uint256)
184d894e  =>  _reflectFee(uint256,uint256)
d4780e36  =>  _getValues(uint256)
1d5671e4  =>  _getRValues(uint256,uint256,uint256,uint256)
94e10784  =>  _getRate()
97a9d560  =>  _getCurrentSupply()
c432df5e  =>  _takeLiquidity(uint256)
```

```
cc126a23  =>  calculateLiquidityFee(uint256)
301370af  =>  removeAllFee()
e7e3e3a7  =>  restoreAllFee()
5342acb4  =>  isExcludedFromFee(address)
104e81ff  =>  _approve(address,address,uint256)
30e0789e  =>  _transfer(address,address,uint256)
173865ad  =>  swapAndLiquify(uint256)
b28805f4  =>  swapTokensForEth(uint256)
9cd441da  =>  addLiquidity(uint256,uint256)
f147aa74  =>  _tokenTransfer(address,address,uint256)
2852df65  =>  _transferStandard(address,address,uint256)
16f1cc83  =>  _transferToExcluded(address,address,uint256)
c7d9be66  =>  _transferFromExcluded(address,address,uint256)
437823ec  =>  excludeFromFee(address)
ea2f0b37  =>  includeInFee(address)
6b3e0cf0  =>  setMaketingWallet(address)
d543dbeb  =>  setMaxTxPercent(uint256)
da6e1b78  =>  setliquidityFee(uint256)
0d12bbdb  =>  setCommunityFee(uint256)
85d6bb81  =>  setCommunityWallet(address)
4bf2c7c9  =>  setBurnFee(uint256)
c49b9a80  =>  setSwapAndLiquifyEnabled(bool)
```

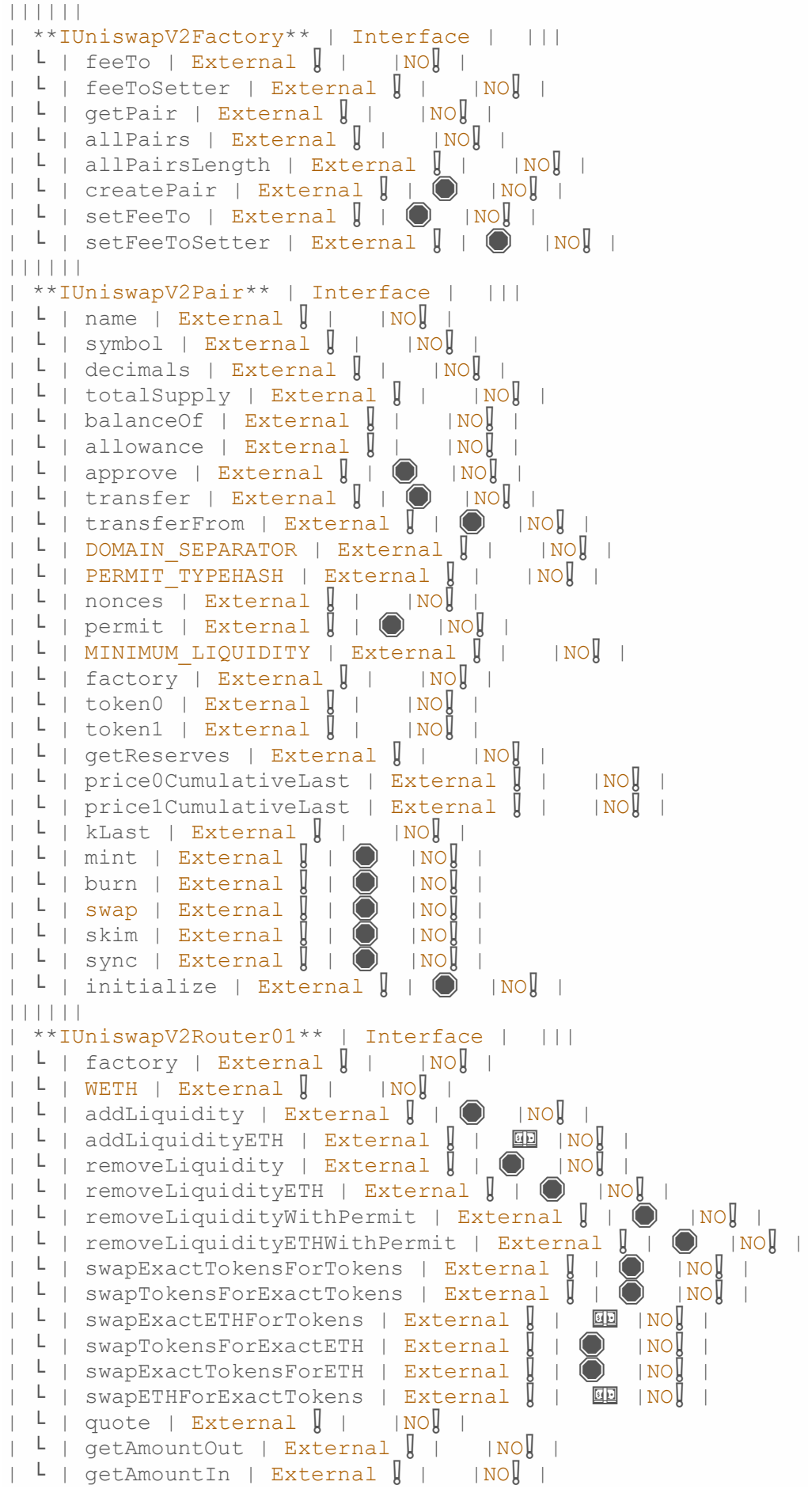
Automatic general report

Files Description Table

File Name	SHA-1 Hash
/Users/macbook/Desktop/smart contracts/NFTME.sol	5dd35612b6be981e9618f5237b3b1a95cf22cf0f

Contracts Description Table

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
IERC20	Interface			
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
SafeMath	Library			
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	mod	Internal 🔒		
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Address	Library			
└	isContract	Internal 🔒		
└	sendValue	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCall	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	functionCallWithValue	Internal 🔒	🛑	
└	_functionCallWithValue	Private 🔐	🛑	
Ownable	Implementation	Context		
└	<Constructor>	Internal 🔒	🛑	
└	owner	Public ❗		NO❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	geUnlockTime	Public ❗		NO❗
└	lock	Public ❗	🛑	onlyOwner
└	unlock	Public ❗	🛑	NO❗


IUniswapV2Factory | Interface
feeTo | External | NO
feeToSetter | External | NO
getPair | External | NO
allPairs | External | NO
allPairsLength | External | NO
createPair | External | NO
setFeeTo | External | NO
setFeeToSetter | External | NO
IUniswapV2Pair | Interface
name | External | NO
symbol | External | NO
decimals | External | NO
totalSupply | External | NO
balanceOf | External | NO
allowance | External | NO
approve | External | NO
transfer | External | NO
transferFrom | External | NO
DOMAIN_SEPARATOR | External | NO
PERMIT_TYPEHASH | External | NO
nonces | External | NO
permit | External | NO
MINIMUM_LIQUIDITY | External | NO
factory | External | NO
token0 | External | NO
token1 | External | NO
getReserves | External | NO
price0CumulativeLast | External | NO
price1CumulativeLast | External | NO
kLast | External | NO
mint | External | NO
burn | External | NO
swap | External | NO
skim | External | NO
sync | External | NO
initialize | External | NO
IUniswapV2Router01 | Interface
factory | External | NO
WETH | External | NO
addLiquidity | External | NO
addLiquidityETH | External | NO
removeLiquidity | External | NO
removeLiquidityETH | External | NO
removeLiquidityWithPermit | External | NO
removeLiquidityETHWithPermit | External | NO
swapExactTokensForTokens | External | NO
swapTokensForExactTokens | External | NO
swapExactETHForTokens | External | NO
swapTokensForExactETH | External | NO
swapExactTokensForETH | External | NO
swapETHForExactTokens | External | NO
quote | External | NO
getAmountOut | External | NO
getAmountIn | External | NO

└	getAmountsOut	External ❗			NO❗
└	getAmountsIn	External ❗			NO❗
IUniswapV2Router02	Interface	IUniswapV2Router01		
└	removeLiquidityETHSupportingFeeOnTransferTokens	External ❗	🛑		NO❗
└	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑		NO❗
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💸		NO❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑		NO❗
NFTME	Implementation	Context, IERC20, Ownable		
└	<Constructor>	Public ❗	🛑		NO❗
└	name	Public ❗			NO❗
└	symbol	Public ❗			NO❗
└	decimals	Public ❗			NO❗
└	totalSupply	Public ❗			NO❗
└	balanceOf	Public ❗			NO❗
└	transfer	Public ❗	🛑		NO❗
└	allowance	Public ❗			NO❗
└	approve	Public ❗	🛑		NO❗
└	transferFrom	Public ❗	🛑		NO❗
└	increaseAllowance	Public ❗	🛑		NO❗
└	decreaseAllowance	Public ❗	🛑		NO❗
└	isExcludedFromReward	Public ❗			NO❗
└	totalFees	Public ❗			NO❗
└	deliver	Public ❗	🛑		NO❗
└	reflectionFromToken	Public ❗			NO❗
└	tokenFromReflection	Public ❗			NO❗
└	excludeFromReward	Public ❗	🛑	onlyOwner
└	includeInReward	External ❗	🛑	onlyOwner
└	_transferBothExcluded	Private 🔒	🛑	
└	<Receive Ether>	External ❗	💸		NO❗
└	_reflectFee	Private 🔒	🛑	
└	_getValues	Private 🔒		
└	_getTValues	Private 🔒		
└	_getRValues	Private 🔒		
└	_getRate	Private 🔒		
└	_getCurrentSupply	Private 🔒		
└	_takeLiquidity	Private 🔒	🛑	
└	calculateTaxFee	Private 🔒		
└	calculateLiquidityFee	Private 🔒		
└	removeAllFee	Private 🔒	🛑	
└	restoreAllFee	Private 🔒	🛑	
└	isExcludedFromFee	Public ❗			NO❗
└	_approve	Private 🔒	🛑	
└	_transfer	Private 🔒	🛑	
└	swapAndLiquify	Private 🔒	🛑	lockTheSwap
└	swapTokensForEth	Private 🔒	🛑	
└	addLiquidity	Private 🔒	🛑	
└	_tokenTransfer	Private 🔒	🛑	
└	_transferStandard	Private 🔒	🛑	
└	_transferToExcluded	Private 🔒	🛑	
└	_transferFromExcluded	Private 🔒	🛑	
└	excludeFromFee	Public ❗	🛑	onlyOwner
└	includeInFee	Public ❗	🛑	onlyOwner
└	setMaketingWallet	External ❗	🛑	onlyOwner

└	setMaxTxPercent	External ❗	🛑	onlyOwner
└	setliquidityFee	External ❗	🛑	onlyOwner
└	setCommunityFee	External ❗	🛑	onlyOwner
└	setCommunityWallet	External ❗	🛑	onlyOwner
└	setBurnFee	External ❗	🛑	onlyOwner
└	setSwapAndLiquifyEnabled	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 “secured”.

- ✔ No mint function.
- ✔ No volatile code.
- ✔ Not many high severity issues were found.
- ✔ Contract Ownership Renounced.

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.