



SAULIDITY



DESPACE  
PROTOCOL

2022

SMART CONTRACT  
SECURITY ANALYSIS

PREPARED BY  
Saulidity

PRESENTED TO  
DeSpace Protocol



# SECURITY REPORT



Smart Contract  
Audit



[saulidity.com](http://saulidity.com)  
Saulidity  
@Saulidity

# DISCLAIMER

This report does not constitute financial advice, and Saulidity is not accountable or liable for any negative consequences resulting from this report, nor may Saulidity be held liable in any way. You agree to the terms of this disclaimer by reading any part of the report. If you do not agree to the terms, please stop reading this report immediately and delete and destroy any and all copies of this report that you have downloaded and/or printed. This report was entirely based on information given by the audited party and facts that existed prior to the audit. Saulidity and its auditors cannot be held liable for any outcome, including modifications (if any) made to the contract(s) for the audit that was completed. No modifications have been made to the contract(s) by the Saulidity team, but if it does, it will be indicated explicitly. The audit does not include the project team, website, logic, or tokenomics, but if it does, it will be indicated explicitly. The security is evaluated only on the basis of smart contracts only. There were no security checks performed on any apps or activities. There hasn't been a review of any product codes. It is assumed by Saulidity that the information and materials given were not tampered with, censored, or misrepresented. Even if this report exists and Saulidity makes every effort to uncover any security flaws, you should not rely completely on it and should conduct your own independent research. Saulidity hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Saulidity, for any amount or kind of loss or damage that may result to you or any other person or any kind of company, community, association and institution. Saulidity is the exclusive owner of this report, and it is published by Saulidity. Without Saulidity's express written authorization, use of this report for any reason other than a security interest in the individual contacts, or use of sections of this report, is forbidden.

# Table of Contents

- 02 Saulidity**
- 03 Introduction**
- 04 Scope**
- 05 Audit & Project Information**
- 06 Summary Table**
- 07 Executive Summary**
- 08 DeSwapMigrator.sol**
- 12 DeSwapRouter.sol**
- 15 DeSwapFactory.sol**
- 19 DeSwapERC20.sol**
- 21 DeSwapPair.sol**
- 25 Testing Standards**

# Saulidity

Saulidity is a renowned cybersecurity firm specializing in the analysis and development of Smart contracts. Saulidity, as a full-service security organization, can help with a variety of audits and project development.

In a market where confidence and trust are key, a genuine project may simply increase its user base enormously with an official audit performed by Saulidity.

# Introduction

For a thorough understanding of the audit, please read the entire document.

The goal of the audit was to find any potential smart contract security problems and vulnerabilities.

The information in this report should be used to understand the smart contract's risk exposure and as a guide to improving the smart contract's security posture by addressing the concerns that were discovered.

During our audit, we conducted a thorough inquiry using automated analysis and manual review approaches.

The security specialists did a complete study independently of one another in order to uncover any security issues in the contracts as comprehensively as feasible. For optimum security and professionalism, all of our audits are undertaken by at least two independent auditors.

The audit was carried out on contracts that had not yet been deployed. The project's website, logic, or tokenomics have not been vetted by the Saulidity team.

# Scope

We analyze smart contracts for both well-known and more specific vulnerabilities.

Here are some of the most well-known vulnerabilities that are taken into account but not limited to:

- Reentrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with (Unexpected) Throw
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Style guide violation
- Transfer forwards all gas
- ERC20 API violation
- Compiler version not fixed
- Unchecked external call - Unchecked math
- Unsafe type inference
- Implicit visibility level

# Audit & Project Information

	Project Name	DeSpace Protocol
	Contract Name	DeSwapMigrator.sol DeSwapRouter.sol DeSwapERC20.sol DeSwapFactory.sol DeSwapPair.sol
	Report ID	SAUL40100 V1.0
	Website	despace.io
	Contact	Obasi Co-Founder
	Contact Information	TG @francisobasil
	Code language	Solidity

# Summary Table

SEVERITY	FOUND
Critical	0
High	0
Medium	0
Low	0
Lowest / Code Style / Optimized Practice	0

# Executive Summary

ACCORDING TO THE ANALYSIS, **THERE ARE NO SECURITY VULNERABILITIES.**

ALL ISSUES FOUND DURING AUTOMATED ANALYSIS WERE MANUALLY REVIEWED, AND 162 FALSE POSITIVES WERE ELIMINATED.

# Inheritance

## DeSwapMigrator.sol

### IPancakePair

Public Functions:

```
name()
symbol()
decimals()
totalSupply()
balanceOf(address)
allowance(address,address)
approve(address,uint256)
transfer(address,uint256)
transferFrom(address,address,uint256)
DOMAIN_SEPARATOR()
PERMIT_TYPEHASH()
nonces(address)
permit(address,address,uint256,uint256,uint8,bytes32,bytes32)
MINIMUM_LIQUIDITY()
factory()
token0()
token1()
getReserves()
price0CumulativeLast()
price1CumulativeLast()
kLast()
mint(address)
burn(address)
swap(uint256,uint256,address,bytes)
skim(address)
sync()
initialize(address,address)
```

### IDeSwapRouter

Public Functions:

```
addLiquidityETH(address,uint256,uint256,uint256,address,uint256)
```

### IDeSwapFactory

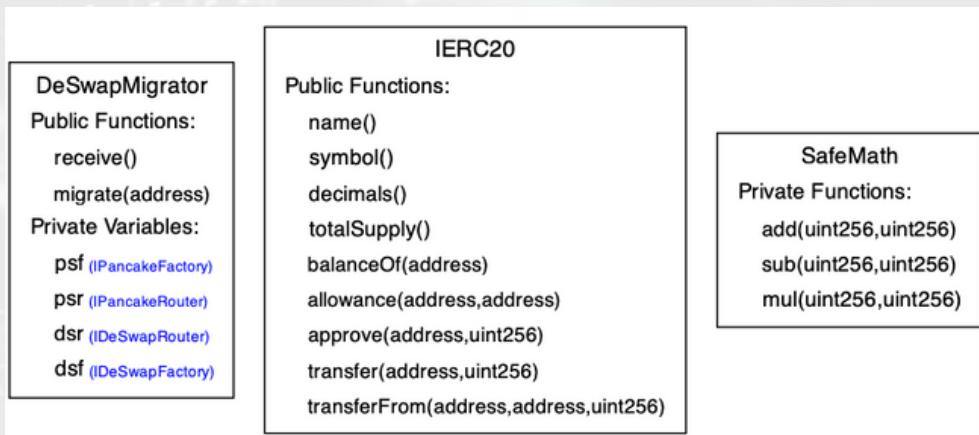
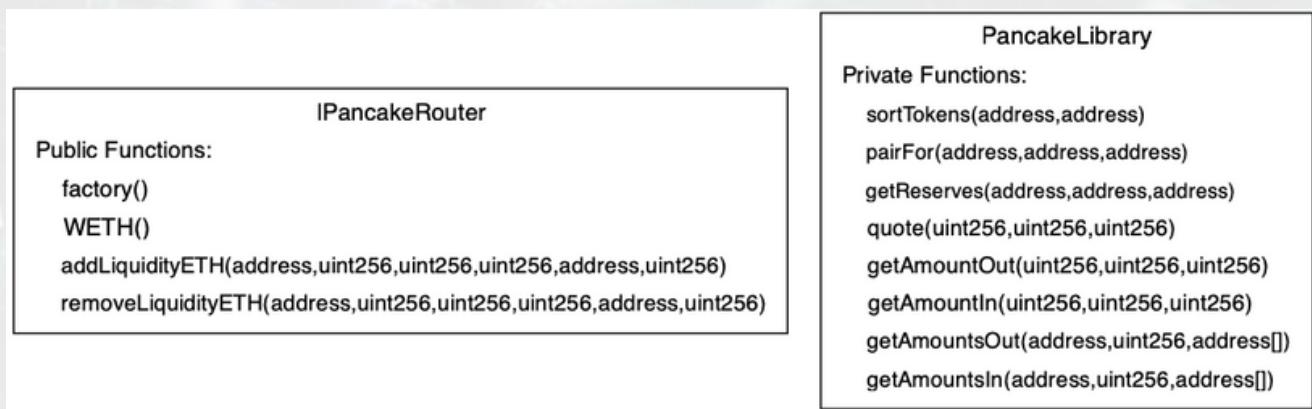
Public Functions:

```
getPair(address,address)
```

### IPancakeFactory

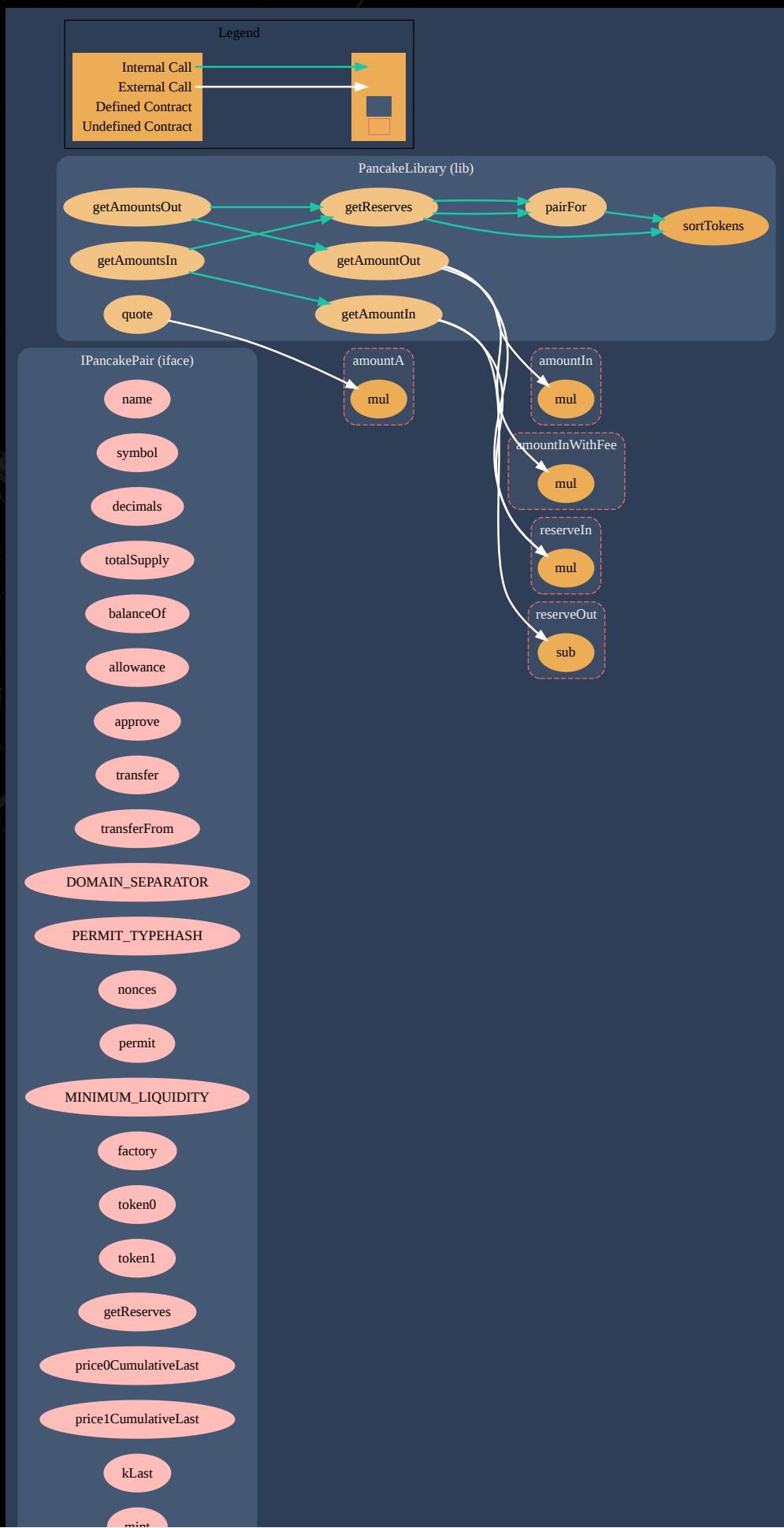
Public Functions:

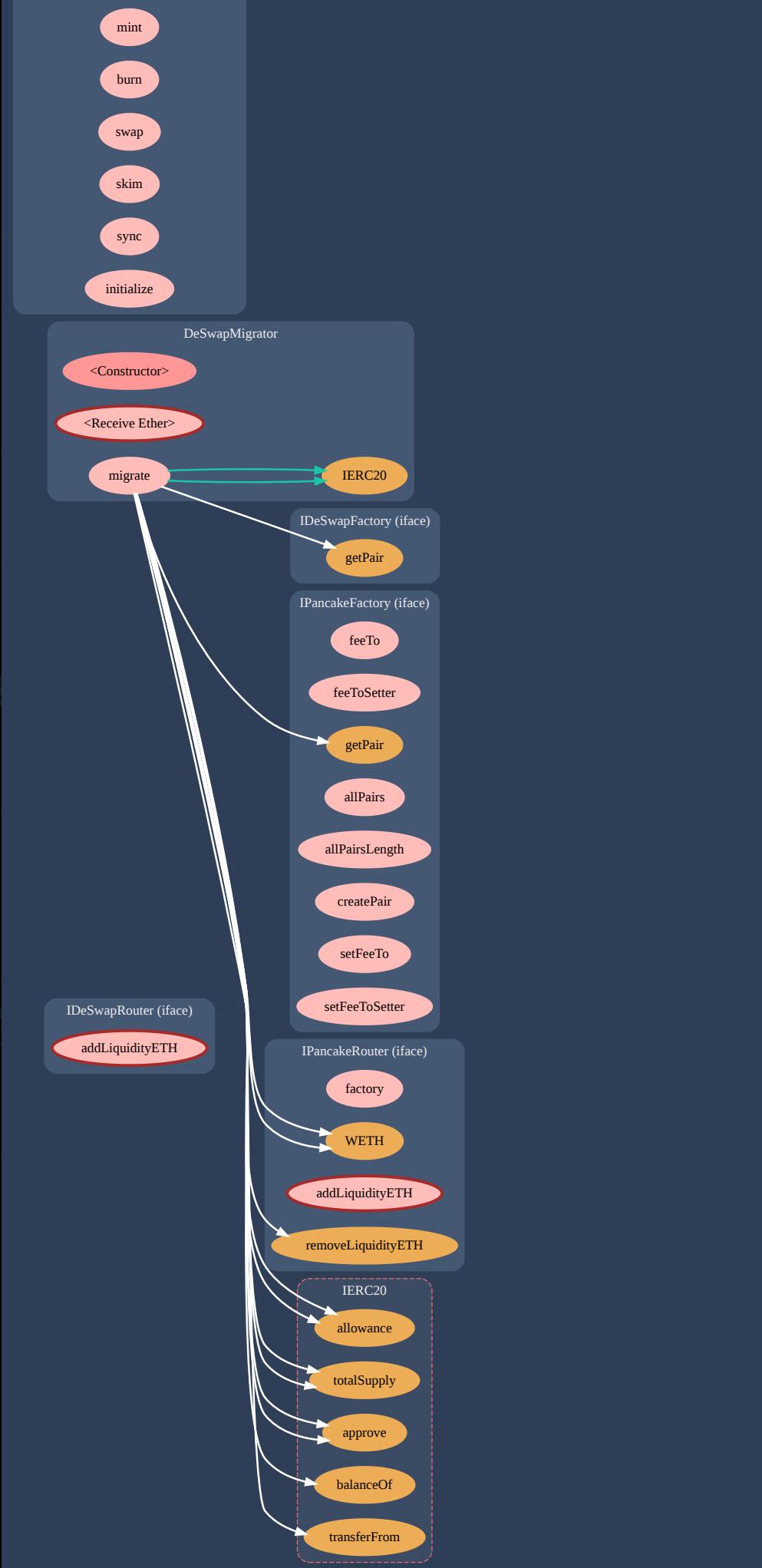
```
feeTo()
feeToSetter()
getPair(address,address)
allPairs(uint256)
allPairsLength()
createPair(address,address)
setFeeTo(address)
setFeeToSetter(address)
```



# Call Graph

## DeSwapMigrator.sol





# Inheritance

## DeSwapRouter.sol

```
DeSwapRouter

Public Functions:
receive()
addLiquidity(address,address,uint256,uint256,uint256,address,uint256)
addLiquidityETH(address,uint256,uint256,uint256,address,uint256)
removeLiquidity(address,address,uint256,uint256,uint256,address,uint256)
removeLiquidityETH(address,uint256,uint256,uint256,address,uint256)
removeLiquidityWithPermit(address,address,uint256,uint256,address,uint256,bool,uint8,bytes32,bytes32)
removeLiquidityETHWithPermit(address,uint256,uint256,uint256,address,uint256,bool,uint8,bytes32,bytes32)
removeLiquidityETHSupportingFeeOnTransferTokens(address,uint256,uint256,uint256,address,uint256)
removeLiquidityETHWithPermitSupportingFeeOnTransferTokens(address,uint256,uint256,uint256,address,uint256,bool,uint8,bytes32,bytes32)
swapExactTokensForTokens(uint256,uint256,address[],address,uint256)
swapTokensForExactTokens(uint256,uint256,address[],address,uint256)
swapExactETHForTokens(uint256,address[],address,uint256)
swapTokensForExactETH(uint256,uint256,address[],address,uint256)
swapExactTokensForETH(uint256,uint256,address[],address,uint256)
swapETHForExactTokens(uint256,address[],address,uint256)
swapExactTokensForTokensSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)
swapExactETHForTokensSupportingFeeOnTransferTokens(uint256,address[],address,uint256)
swapExactTokensForETHSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)
quote(uint256,uint256,uint256)
getAmountOut(uint256,uint256,uint256)
getAmountIn(uint256,uint256,uint256)
getAmountsOut(uint256,address[])
getAmountsIn(uint256,address[])

Private Functions:
__addLiquidity(address,address,uint256,uint256,uint256,uint256)
__swap(uint256[],address[],address)
__swapSupportingFeeOnTransferTokens(address[],address)

Modifiers:
ensure(uint256)

Public Variables:
factory
WBNB
```

↓

```
IUniswapV2Router02

Public Functions:
removeLiquidityETHSupportingFeeOnTransferTokens(address,uint256,uint256,uint256,address,uint256)
removeLiquidityETHWithPermitSupportingFeeOnTransferTokens(address,uint256,uint256,uint256,address,uint256,bool,uint8,bytes32,bytes32)
swapExactTokensForTokensSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)
swapExactETHForTokensSupportingFeeOnTransferTokens(uint256,address[],address,uint256)
swapExactTokensForETHSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)
```

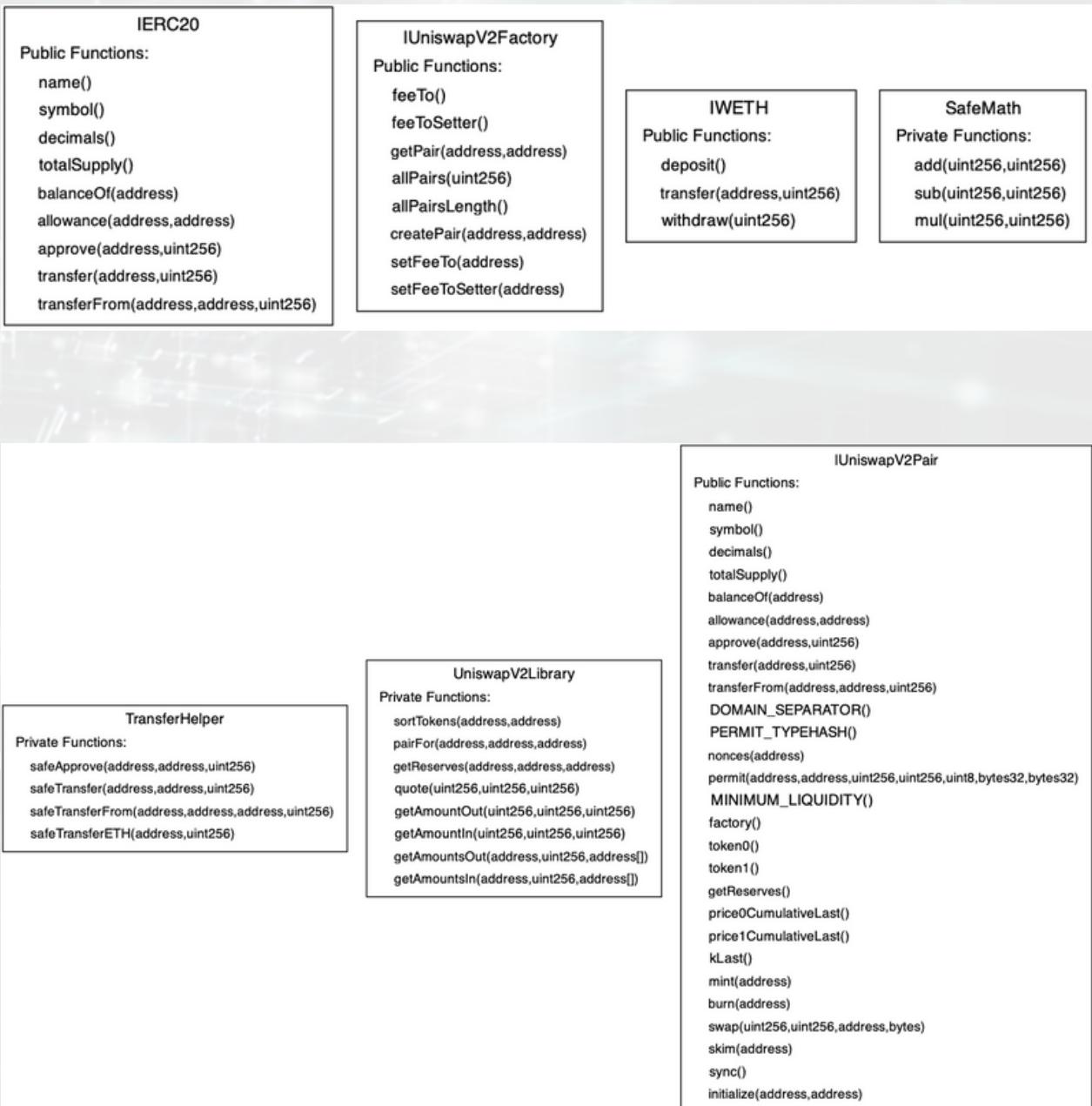
↓

```
IUniswapV2Router01

Public Functions:
factory()
WBNB()
addLiquidity(address,address,uint256,uint256,uint256,address,uint256)
addLiquidityETH(address,uint256,uint256,uint256,address,uint256)
removeLiquidity(address,address,uint256,uint256,uint256,address,uint256)
removeLiquidityETH(address,uint256,uint256,uint256,address,uint256)
removeLiquidityWithPermit(address,address,uint256,uint256,uint256,address,uint256,bool,uint8,bytes32,bytes32)
removeLiquidityETHWithPermit(address,uint256,uint256,uint256,address,uint256,bool,uint8,bytes32,bytes32)
swapExactTokensForTokens(uint256,uint256,address[],address,uint256)
swapTokensForExactTokens(uint256,uint256,address[],address,uint256)
swapExactETHForTokens(uint256,address[],address,uint256)
swapTokensForExactETH(uint256,uint256,address[],address,uint256)
swapExactTokensForETH(uint256,uint256,address[],address,uint256)
swapETHForExactTokens(uint256,address[],address,uint256)
quote(uint256,uint256,uint256)
getAmountOut(uint256,uint256,uint256)
getAmountIn(uint256,uint256,uint256)
getAmountsOut(uint256,address[])
getAmountsIn(uint256,address[])
```

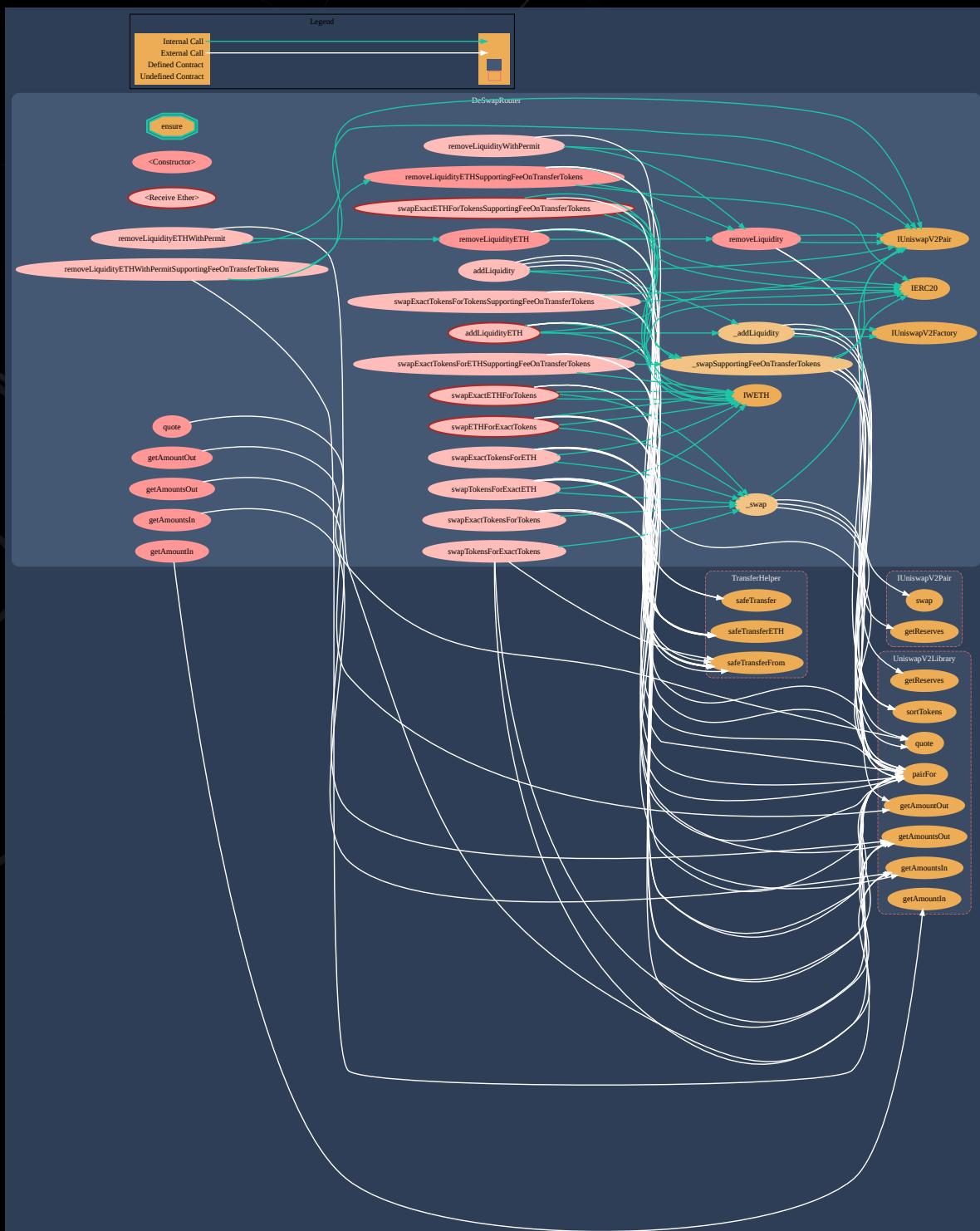
# Inheritance

## DeSwapRouter.sol



# Call Graph

## DeSwapRouter.sol



# Inheritance

## DeSwapFactory.sol

### DeSwapPair

#### Public Functions:

```
getReserves()  
initialize(address,address)  
mint(address)  
burn(address)  
swap(uint256,uint256,address,bytes)  
skim(address)  
sync()
```

#### Private Functions:

```
_safeTransfer(address,address,uint256)  
_update(uint256,uint256,uint112,uint112)  
_mintFee(uint112,uint112)
```

#### Modifiers:

```
lock()
```

#### Public Variables:

```
MINIMUM_LIQUIDITY  
factory  
token0  
token1  
price0CumulativeLast  
price1CumulativeLast  
kLast
```

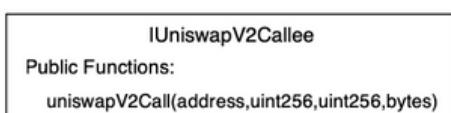
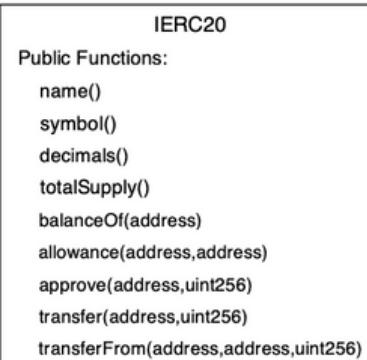
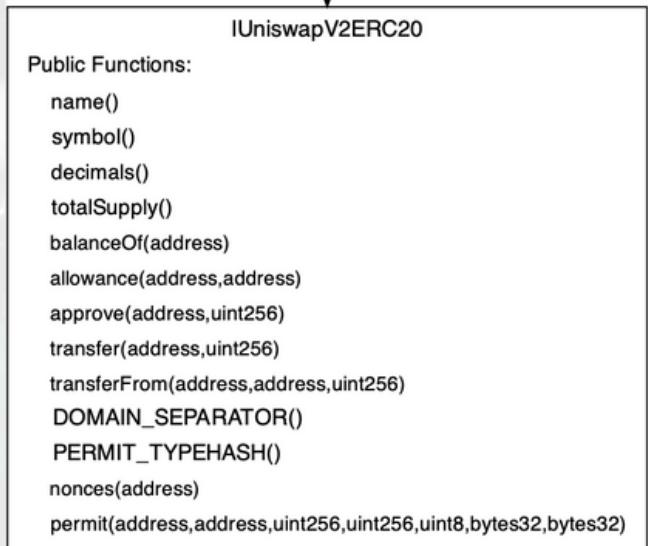
#### Private Variables:

```
SELECTOR  
reserve0  
reserve1  
blockTimestampLast  
unlocked
```

```
'name()' collides in inherited contracts IUniswapV2Pair, IUniswapV2ERC20 where IUniswapV2ERC20 is chosen.  
'symbol()' collides in inherited contracts IUniswapV2Pair, IUniswapV2ERC20 where IUniswapV2ERC20 is chosen.  
'decimals()' collides in inherited contracts IUniswapV2Pair, IUniswapV2ERC20 where IUniswapV2ERC20 is chosen.  
'totalSupply()' collides in inherited contracts IUniswapV2Pair, IUniswapV2ERC20 where IUniswapV2ERC20 is chosen.  
'balanceOf(address)' collides in inherited contracts IUniswapV2Pair, IUniswapV2ERC20 where IUniswapV2ERC20 is chosen.  
'allowance(address,address)' collides in inherited contracts IUniswapV2Pair, IUniswapV2ERC20 where IUniswapV2ERC20 is chosen.  
'approve(address,uint256)' collides in inherited contracts IUniswapV2Pair, DeSwapERC20 where DeSwapERC20 is chosen.  
'transfer(address,uint256)' collides in inherited contracts IUniswapV2Pair, DeSwapERC20 where DeSwapERC20 is chosen.  
'transferFrom(address,address,uint256)' collides in inherited contracts IUniswapV2Pair, DeSwapERC20 where DeSwapERC20 is chosen.  
'DOMAIN_SEPARATOR()' collides in inherited contracts IUniswapV2Pair, IUniswapV2ERC20 where IUniswapV2ERC20 is chosen.  
'PERMIT_TYPEHASH()' collides in inherited contracts IUniswapV2Pair, IUniswapV2ERC20 where IUniswapV2ERC20 is chosen.  
'nonces(address)' collides in inherited contracts IUniswapV2Pair, IUniswapV2ERC20 where IUniswapV2ERC20 is chosen.  
'permit(address,address,uint256,uint8,bytes32,bytes32)' collides in inherited contracts IUniswapV2Pair, DeSwapERC20 where DeSwapERC20 is chosen.
```

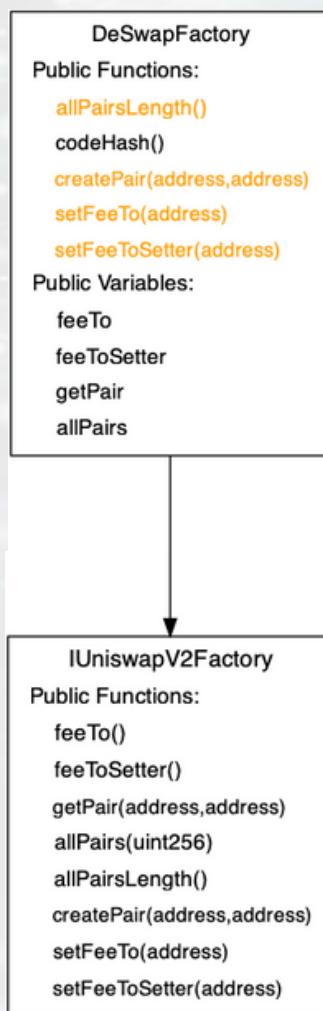
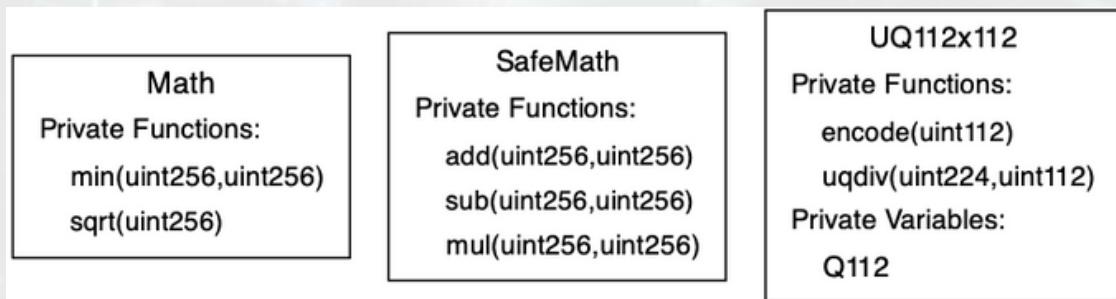
2

1



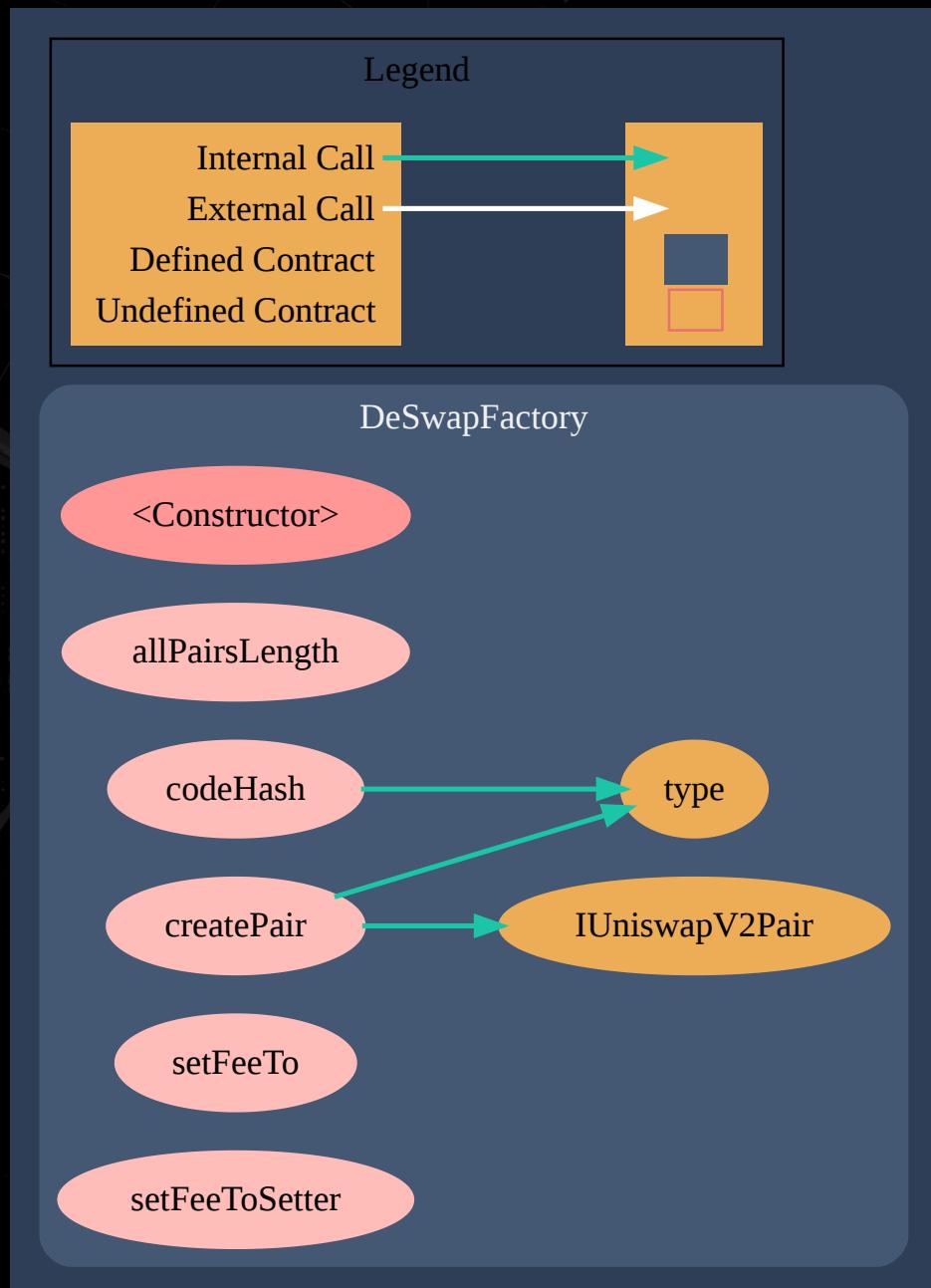
# Inheritance

## DeSwapFactory.sol



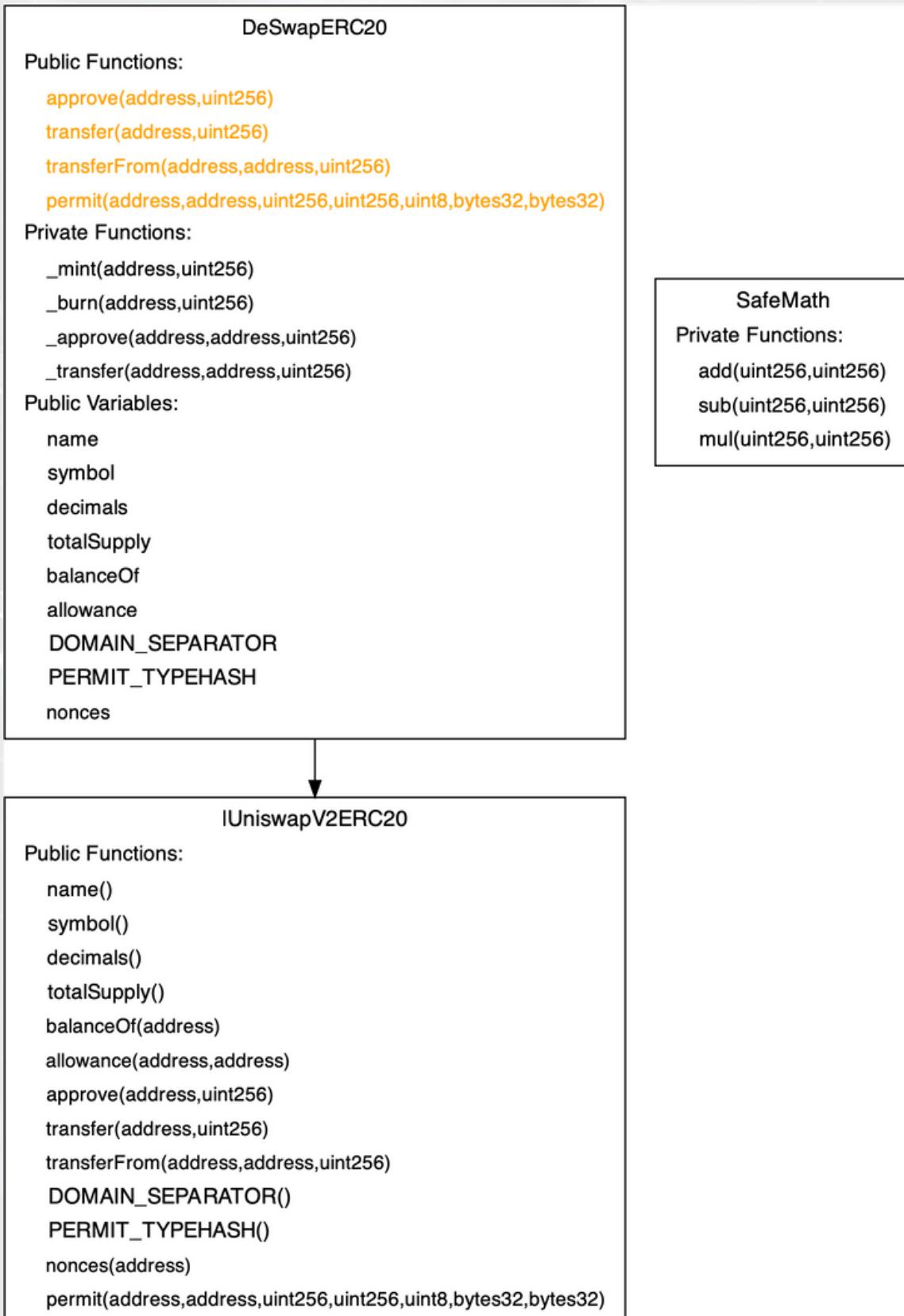
# Call Graph

DeSwapFactory.sol



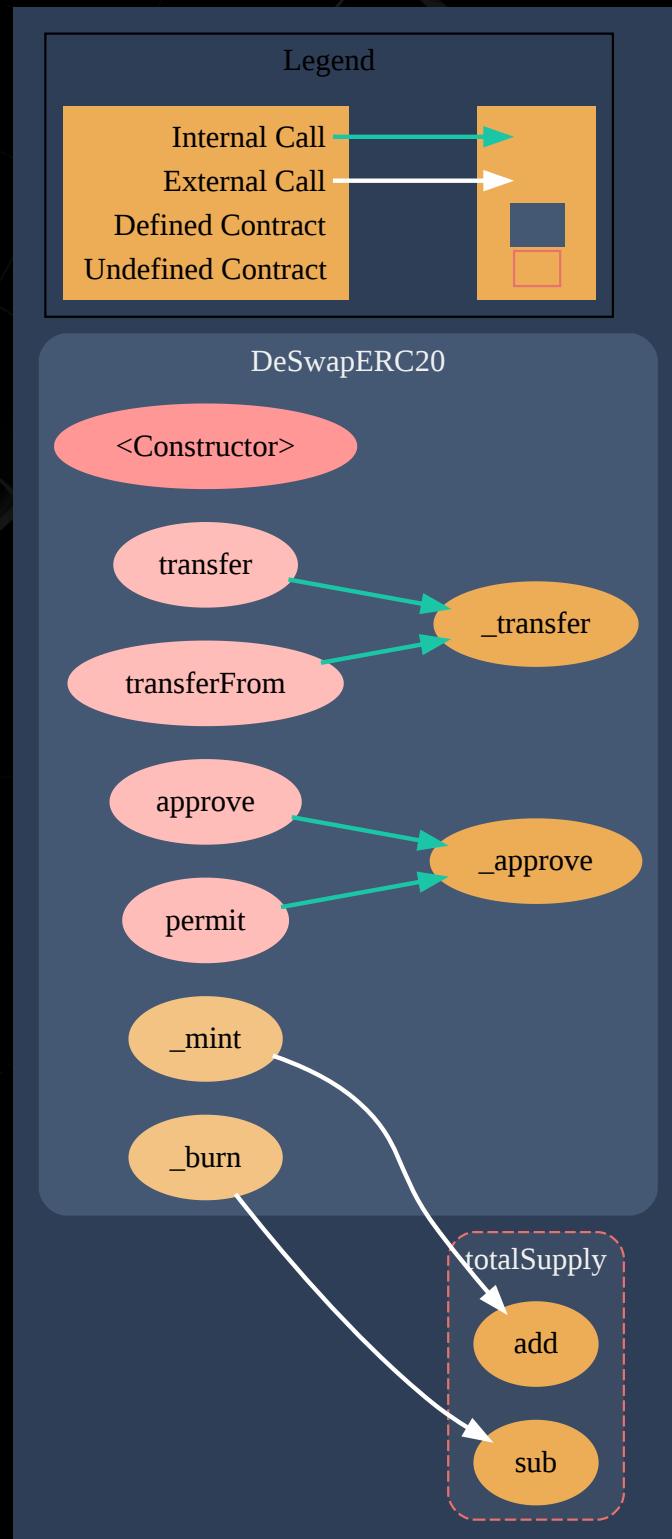
# Inheritance

## DeSwapERC20.sol



# Call Graph

DeSwapERC20.sol



# Inheritance

## DeSwapPair.sol

### DeSwapPair

#### Public Functions:

```
getReserves()
initialize(address,address)
mint(address)
burn(address)
swap(uint256,uint256,address,bytes)
skim(address)
sync()
```

#### Private Functions:

```
_safeTransfer(address,address,uint256)
_update(uint256,uint256,uint112,uint112)
_mintFee(uint112,uint112)
```

#### Modifiers:

```
lock()
```

#### Public Variables:

```
MINIMUM_LIQUIDITY
factory
token0
token1
price0CumulativeLast
price1CumulativeLast
kLast
```

#### Private Variables:

```
SELECTOR
reserve0
reserve1
blockTimestampLast
unlocked
```

```
'name()' collides in inherited contracts IUniswapV2Pair, IUniswapV2ERC20 where IUniswapV2ERC20 is chosen.
'symbol()' collides in inherited contracts IUniswapV2Pair, IUniswapV2ERC20 where IUniswapV2ERC20 is chosen.
'decimals()' collides in inherited contracts IUniswapV2Pair, IUniswapV2ERC20 where IUniswapV2ERC20 is chosen.
'totalSupply()' collides in inherited contracts IUniswapV2Pair, IUniswapV2ERC20 where IUniswapV2ERC20 is chosen.
'balanceOf(address)' collides in inherited contracts IUniswapV2Pair, IUniswapV2ERC20 where IUniswapV2ERC20 is chosen.
'allowance(address,address)' collides in inherited contracts IUniswapV2Pair, IUniswapV2ERC20 where IUniswapV2ERC20 is chosen.
'approve(address,uint256)' collides in inherited contracts IUniswapV2Pair, DeSwapERC20 where DeSwapERC20 is chosen.
'transfer(address,uint256)' collides in inherited contracts IUniswapV2Pair, DeSwapERC20 where DeSwapERC20 is chosen.
'transferFrom(address,address,uint256)' collides in inherited contracts IUniswapV2Pair, DeSwapERC20 where DeSwapERC20 is chosen.
'DOMAIN_SEPARATOR()' collides in inherited contracts IUniswapV2Pair, IUniswapV2ERC20 where IUniswapV2ERC20 is chosen.
'PERMIT_TYPEHASH()' collides in inherited contracts IUniswapV2Pair, IUniswapV2ERC20 where IUniswapV2ERC20 is chosen.
'nonces(address)' collides in inherited contracts IUniswapV2Pair, IUniswapV2ERC20 where IUniswapV2ERC20 is chosen.
'permit(address,address,uint256,uint256,uint8,bytes32,bytes32)' collides in inherited contracts IUniswapV2Pair, DeSwapERC20 where DeSwapERC20 is chosen.
```

2

1

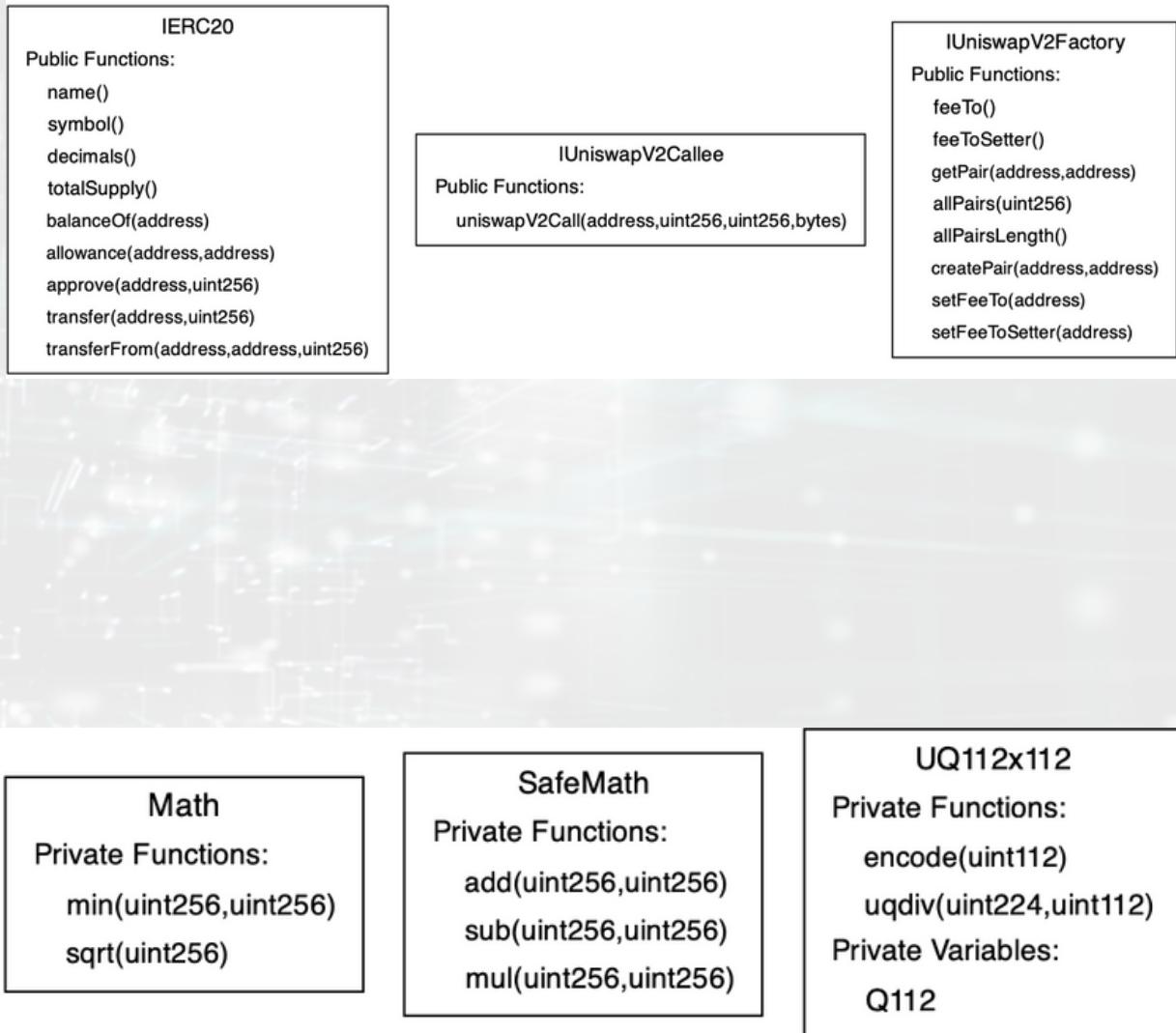
# Inheritance

## DeSwapPair.sol



# Inheritance

## DeSwapPair.sol



**IUniswapV2Callee**  
Public Functions:  
uniswapV2Call(address,uint256,uint256,bytes)

**IUniswapV2Factory**  
Public Functions:  
feeTo()  
feeToSetter()  
getPair(address,address)  
allPairs(uint256)  
allPairsLength()  
createPair(address,address)  
setFeeTo(address)  
setFeeToSetter(address)

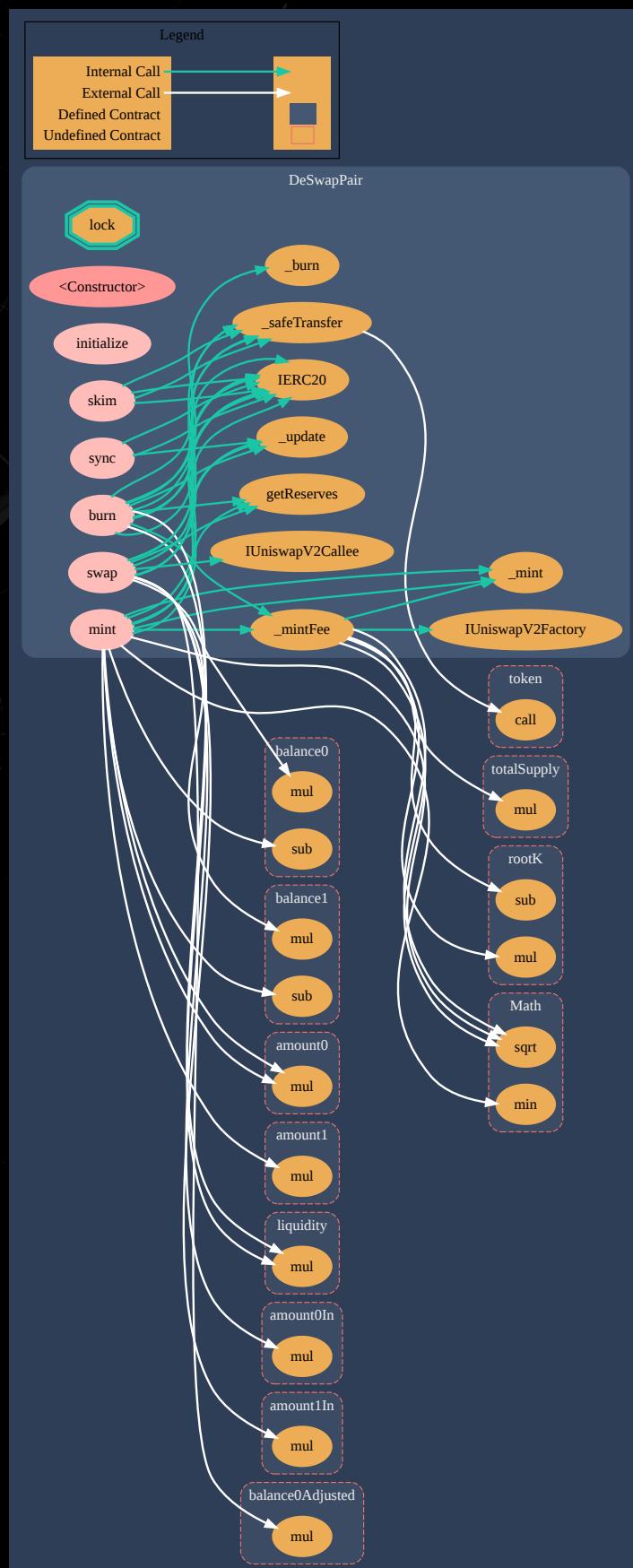
**Math**  
Private Functions:  
min(uint256,uint256)  
sqrt(uint256)

**SafeMath**  
Private Functions:  
add(uint256,uint256)  
sub(uint256,uint256)  
mul(uint256,uint256)

**UQ112x112**  
Private Functions:  
encode(uint112)  
uqdiv(uint224,uint112)  
Private Variables:  
Q112

# Call Graph

DeSwapPair.sol



# Testing Standards

The goal of the audit was to find any potential smart contract security problems and vulnerabilities.

The information in this report should be used to understand the smart contract's risk exposure and as a guide to improving the smart contract's security posture by addressing the concerns that were discovered.

The blockchain platform is used to deploy and execute smart contracts. The platform, its programming language, and other smart contract-related applications all have vulnerabilities that may be exploited. As a result, the audit cannot ensure the audited smart contract(s) explicit security. Audits can't make statements or warranties on security of the code. It also cannot be deemed an adequate assessment of the code's utility and safety, bug-free status, or any statements of the smart contract.

While we did our best in completing the study and publishing this report, it is crucial to emphasize that you should not rely only on it; we advocate all projects doing many independent audits and participating in a public bug bounty program to assure smart contract security.

# Testing Standards

1. Gather all relevant data.
2. Perform a preliminary visual examination of all documents and contracts.
3. Find security holes with specialist tools & manual review with independent experts.
4. Create and distribute a report.



# SAULIDITY



Smart Contract  
Audit



[saulidity.com](http://saulidity.com)  
Saulidity  
@Saulidity