SOLIDITY AUDIT

Smart Contract Solidity Audit

Audit Details:

Audited project: **Skibididopdop Token** (DOPDOP)
Deployer Address: 0x2e833bf1c19719fea836434151f20902b00e991c
Blockchain: Ethereum

21/05/2023

AUDIT

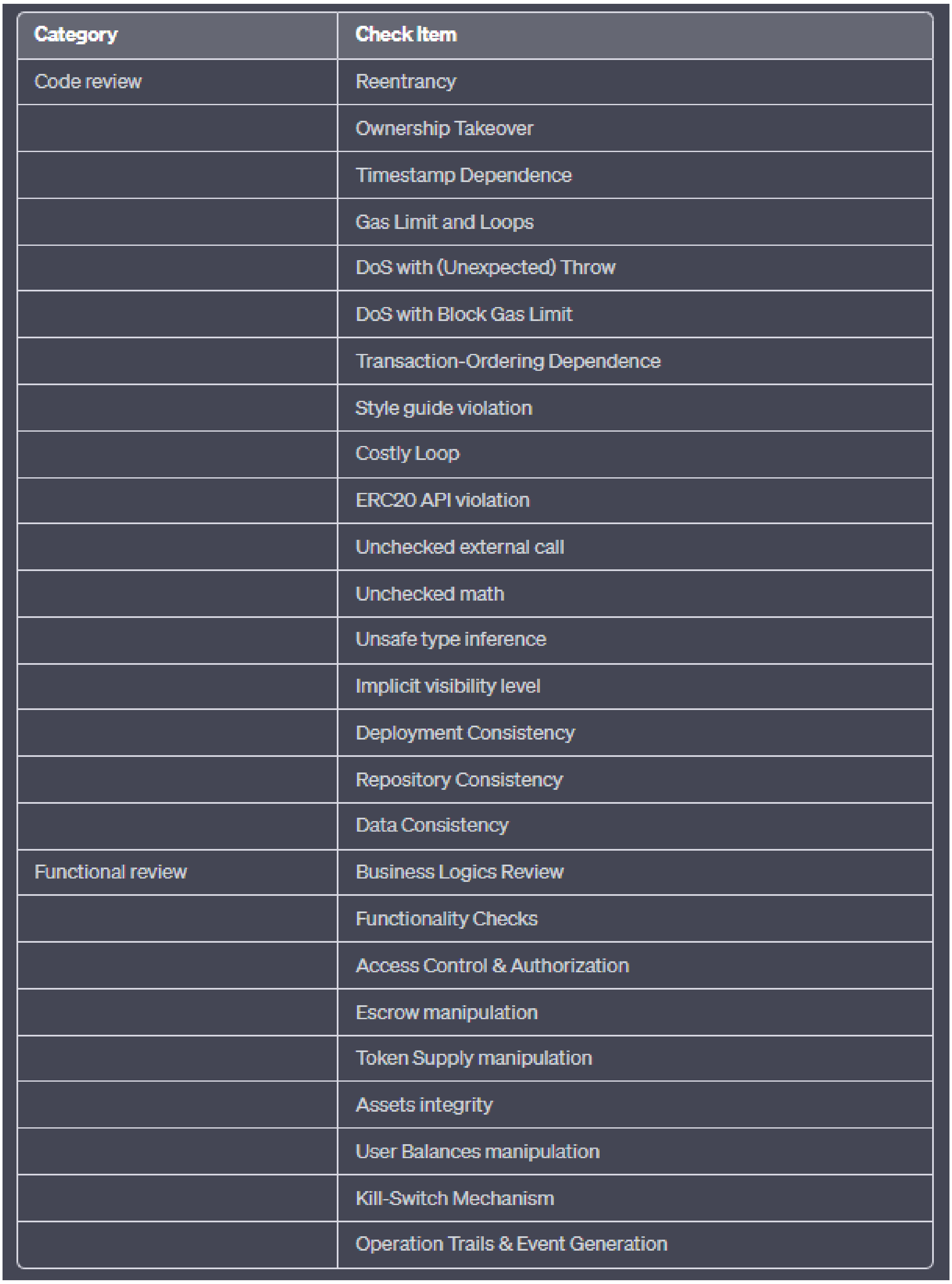
This document may contain confidential information about IT systems and the intellectual property of the Customer as well as information about potential vulnerabilities and methods of their exploitation. The report containing confidential information can be used internally by the Customer, or it can be disclosed publicly after all vulnerabilities are fixed - upon a decision of the Customer.

INTRODUCTION

Solidity Audit (Consultant) was contracted by **Skibididopdop Token** (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of Customer`s smart contract . Scope The scope of the project is main net smart contracts that can be found on Etherscan:

https://etherscan.io/token/0x2e833bf1c19719fea836434151f20902b00e991c

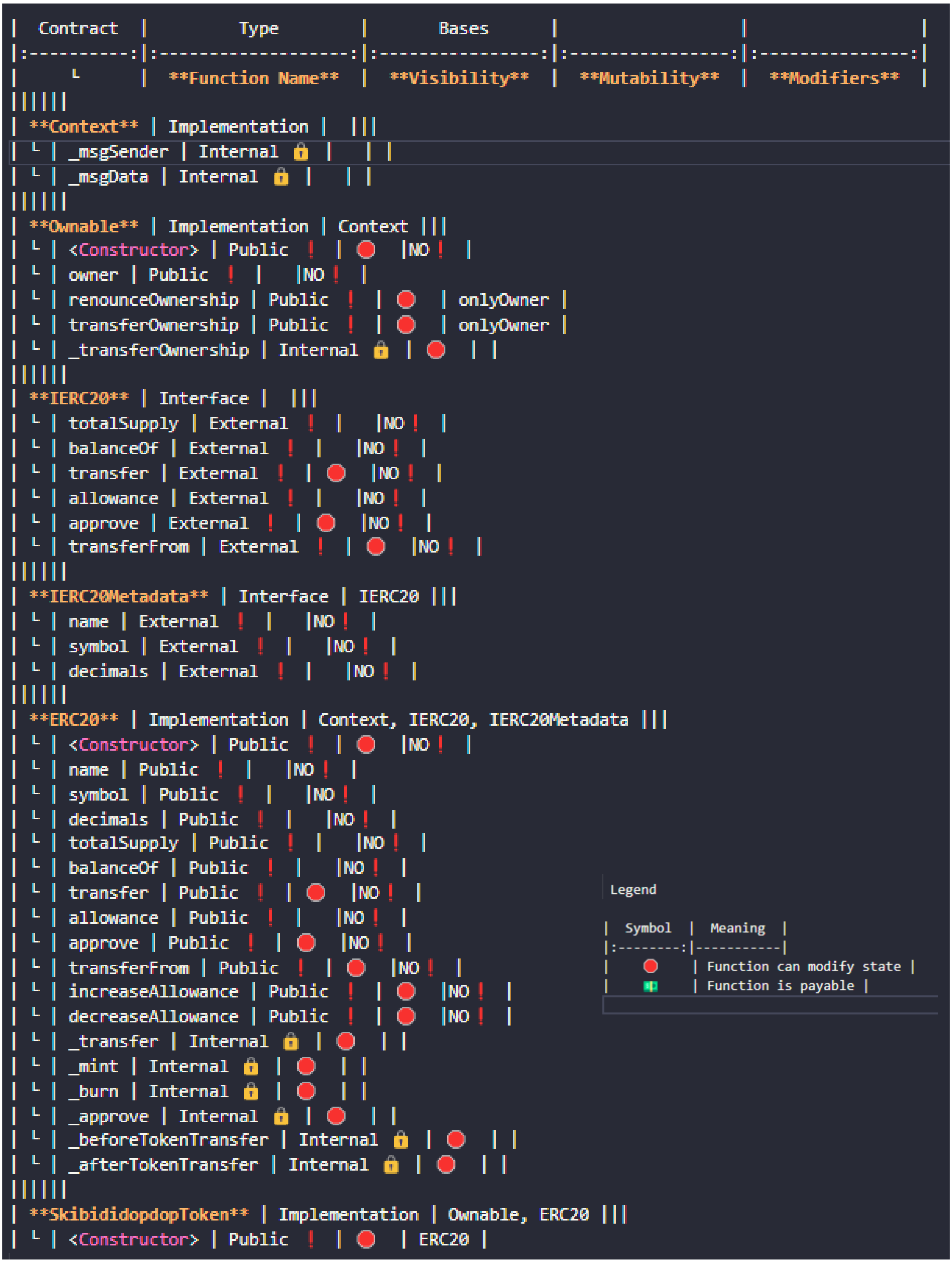
We have scanned this smart contract for commonly known and more specific vulnerabilities. List of the commonly known vulnerabilities that are considered:



Category	Check Item
Code review	Reentrancy
	Ownership Takeover
	Timestamp Dependence
	Gas Limit and Loops
	DoS with (Unexpected) Throw
	DoS with Block Gas Limit
	Transaction-Ordering Dependence
	Style guide violation
	Costly Loop
	ERC20 API violation
	Unchecked external call
	Unchecked math
	Unsafe type inference
	Implicit visibility level
	Deployment Consistency
	Repository Consistency
	Data Consistency
Functional review	Business Logics Review
	Functionality Checks
	Access Control & Authorization
	Escrow manipulation
	Token Supply manipulation
	Assets integrity
	User Balances manipulation
	Kill-Switch Mechanism
	Operation Trails & Event Generation

Our team performed a code functionality analysis, manual auditing and automated checks with Mythril and Slither and a few other tools. All issues found during the automated review were manually reviewed and important vulnerabilities are listed in the Audit Overview section.

Contracts Description Table

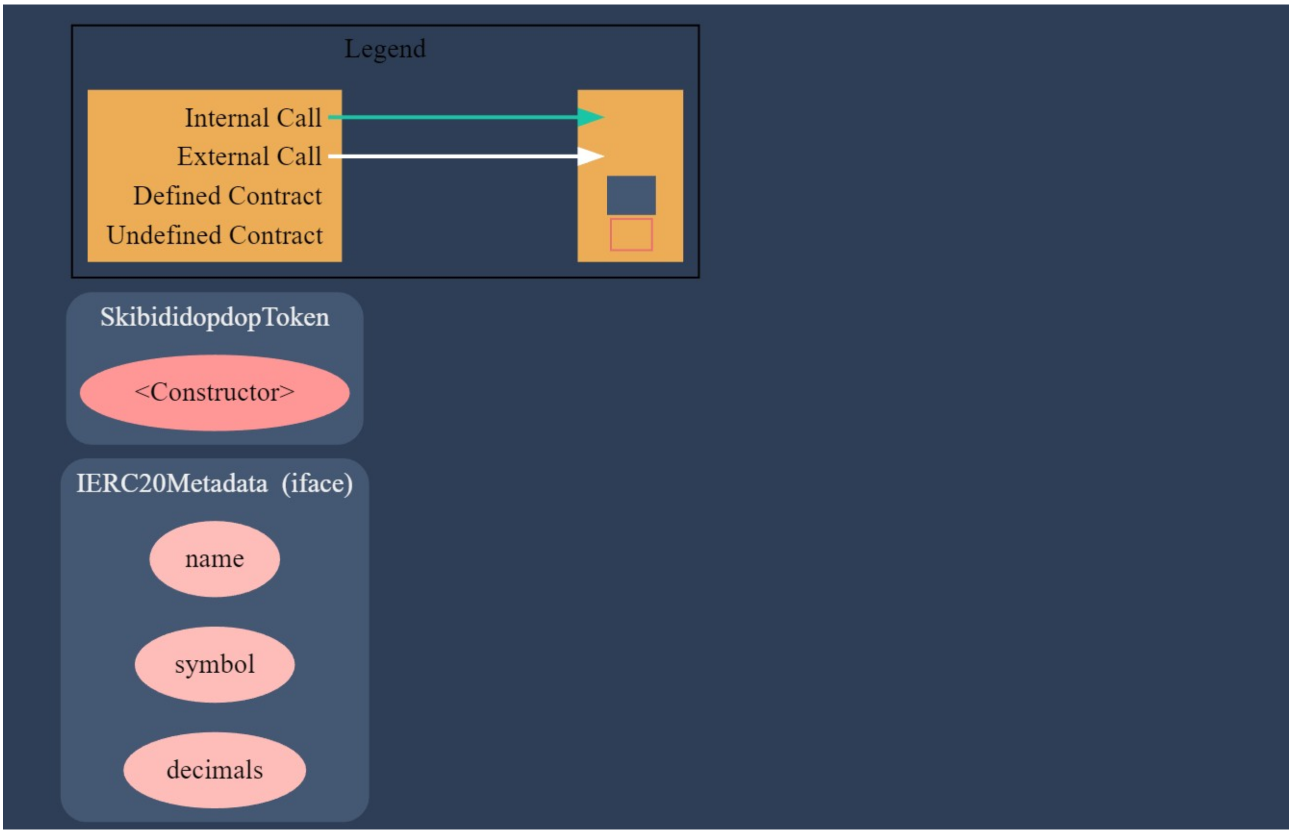


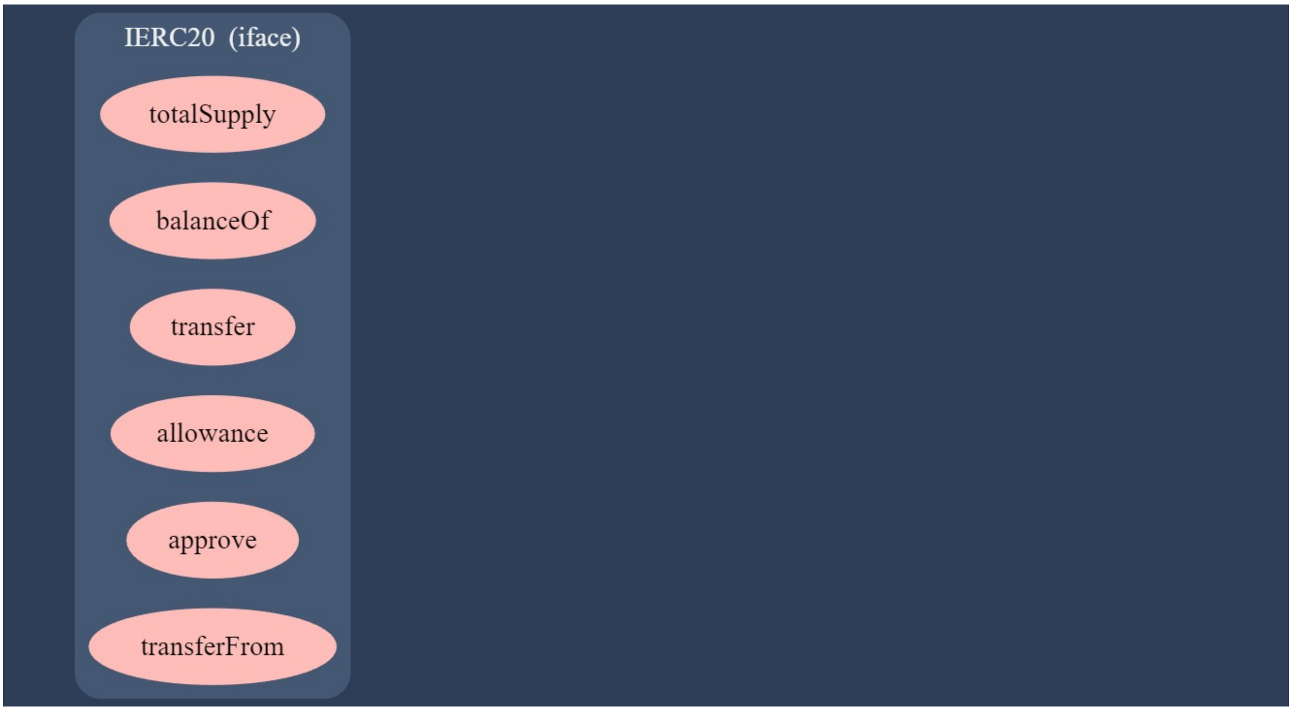
```
|  Contract  |        Type        |       Bases      |                  |                 |
|:----------:|:-------------------:|:----------------:|:----------------:|:---------------:|
|     └      |  **Function Name**  |  **Visibility**  |  **Mutability**  |  **Modifiers**  |
||||||
| **Context** | Implementation |  |||
| └ | _msgSender | Internal 🔒 |   | |
| └ | _msgData | Internal 🔒 |   | |
||||||
| **Ownable** | Implementation | Context |||
| └ | <Constructor> | Public ❗️ | 🔴  |NO❗️ |
| └ | owner | Public ❗️ |   |NO❗️ |
| └ | renounceOwnership | Public ❗️ | 🔴  | onlyOwner |
| └ | transferOwnership | Public ❗️ | 🔴  | onlyOwner |
| └ | _transferOwnership | Internal 🔒 | 🔴  | |
||||||
| **IERC20** | Interface |  |||
| └ | totalSupply | External ❗️ |   |NO❗️ |
| └ | balanceOf | External ❗️ |   |NO❗️ |
| └ | transfer | External ❗️ | 🔴  |NO❗️ |
| └ | allowance | External ❗️ |   |NO❗️ |
| └ | approve | External ❗️ | 🔴  |NO❗️ |
| └ | transferFrom | External ❗️ | 🔴  |NO❗️ |
||||||
| **IERC20Metadata** | Interface | IERC20 |||
| └ | name | External ❗️ |   |NO❗️ |
| └ | symbol | External ❗️ |   |NO❗️ |
| └ | decimals | External ❗️ |   |NO❗️ |
||||||
| **ERC20** | Implementation | Context, IERC20, IERC20Metadata |||
| └ | <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❗️ |
| └ | _transfer | Internal 🔒 | 🔴  | |
| └ | _mint | Internal 🔒 | 🔴  | |
| └ | _burn | Internal 🔒 | 🔴  | |
| └ | _approve | Internal 🔒 | 🔴  | |
| └ | _beforeTokenTransfer | Internal 🔒 | 🔴  | |
| └ | _afterTokenTransfer | Internal 🔒 | 🔴  | |
||||||
| **SkibididopdopToken** | Implementation | Ownable, ERC20 |||
| └ | <Constructor> | Public ❗️ | 🔴  | ERC20 |
```

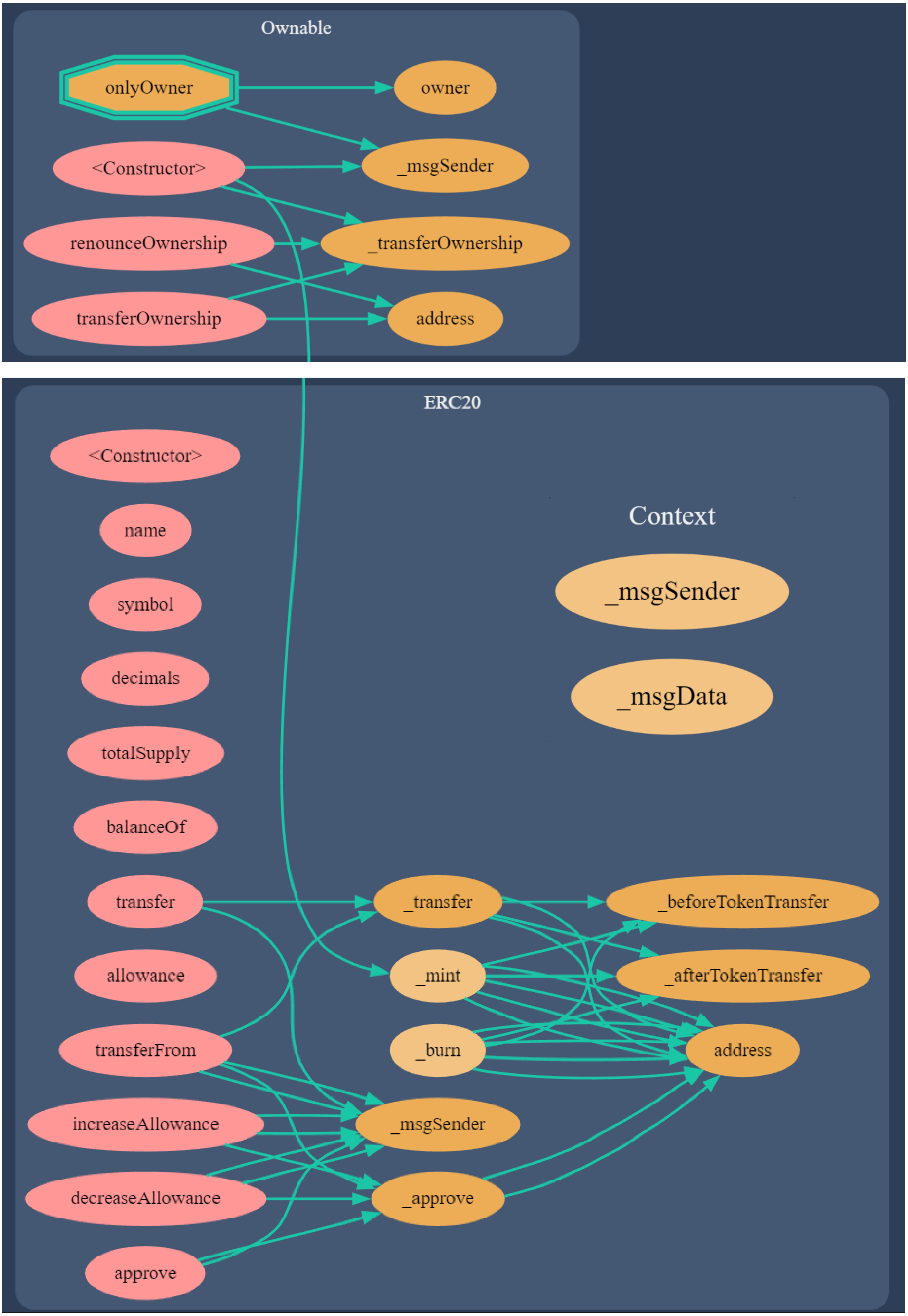
```
Legend

|  Symbol  |  Meaning  |
|:--------:|-----------|
|    🔴    | Function can modify state |
|    💵    | Function is payable |
```

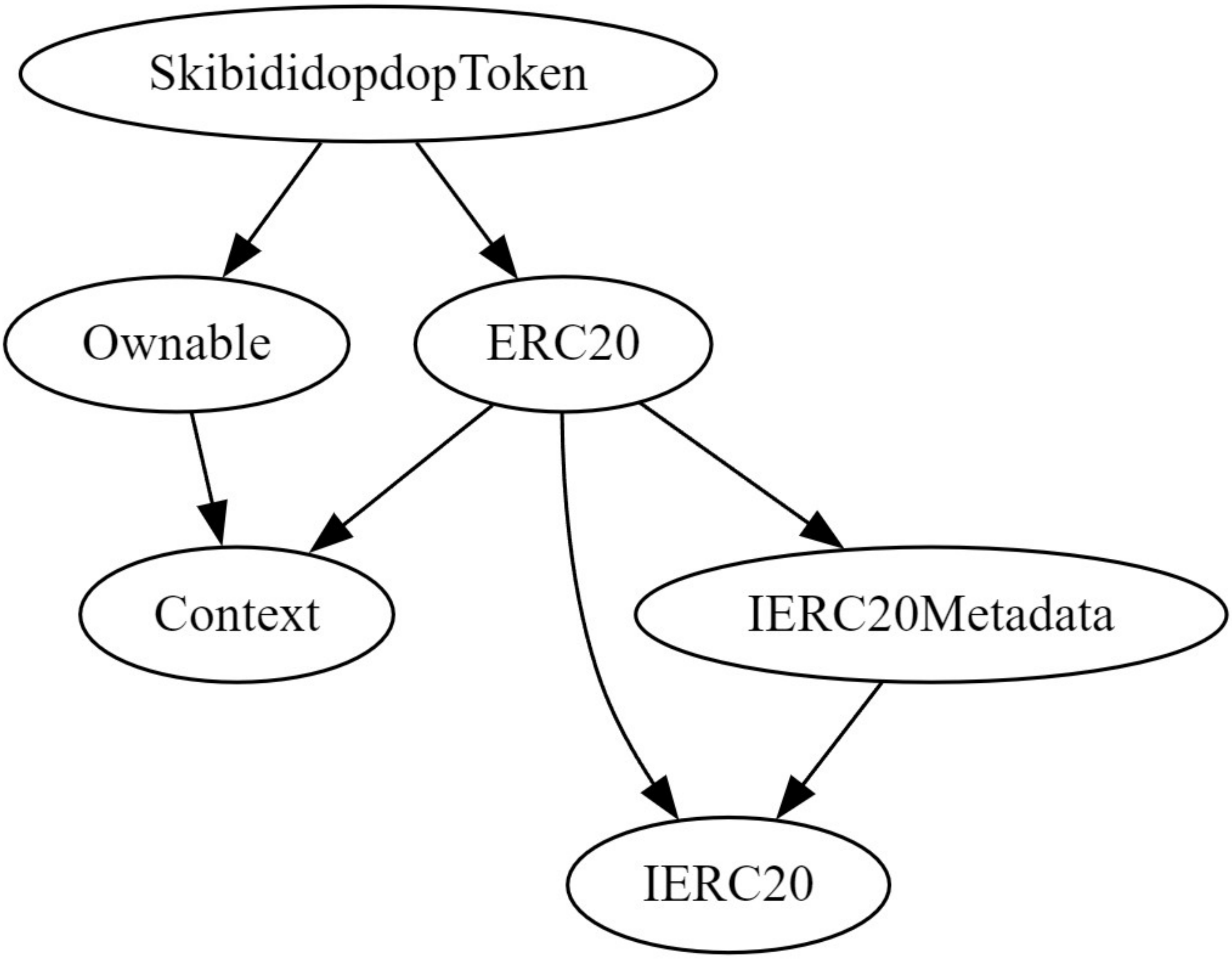
GRAPH


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
SkibididopdopToken
<Constructor>
IERC20Metadata (iface)
name
symbol
decimals

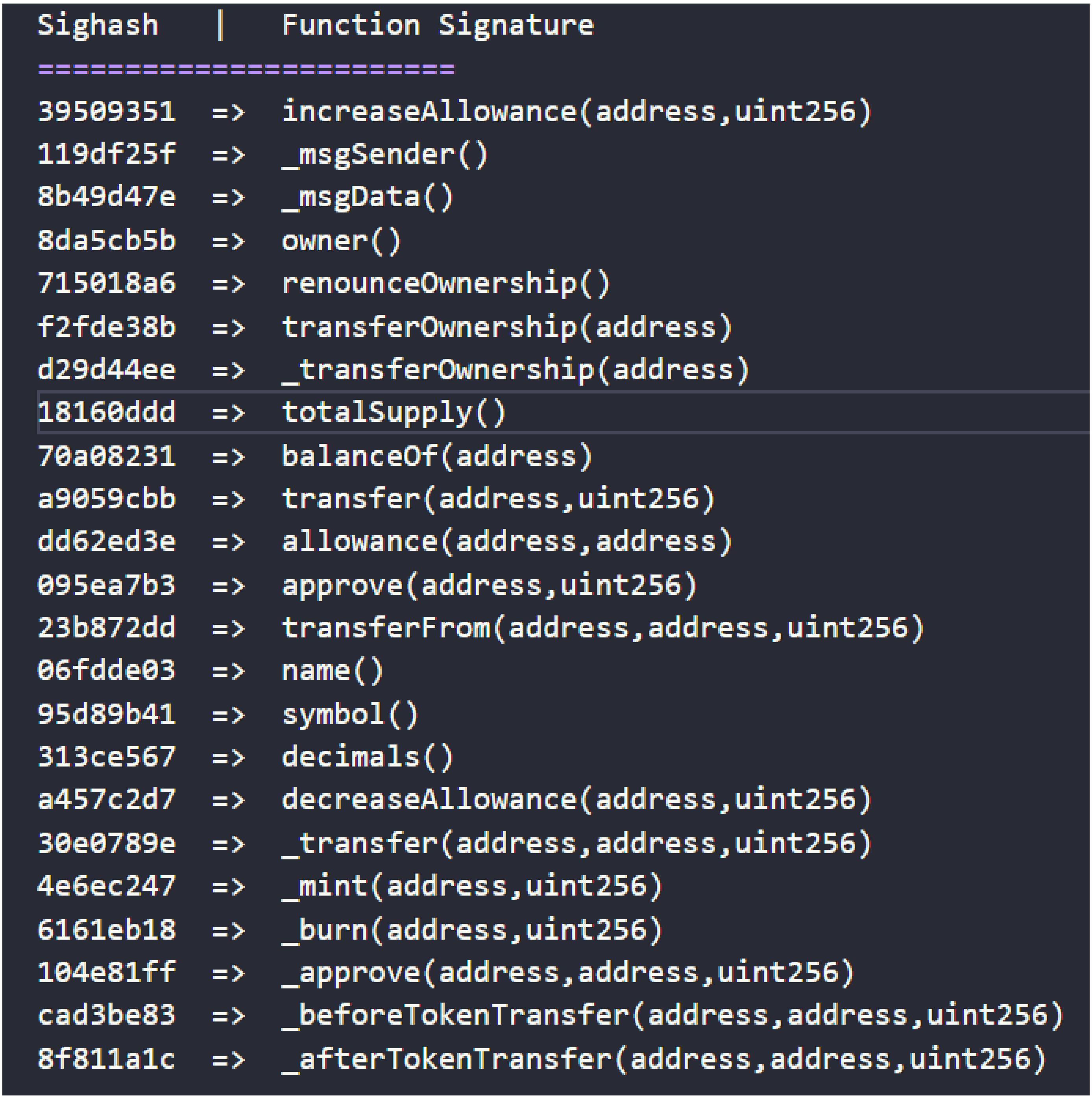

IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom


Ownable
onlyOwner
owner
<Constructor>
_msgSender
renounceOwnership
_transferOwnership
transferOwnership
address
ERC20
<Constructor>
name
symbol
decimals
totalSupply
balanceOf
transfer
allowance
transferFrom
increaseAllowance
decreaseAllowance
approve
Context
_msgSender
_msgData
_transfer
_mint
_burn
_msgSender
_approve
_beforeTokenTransfer
_afterTokenTransfer
address

INHERITANCE

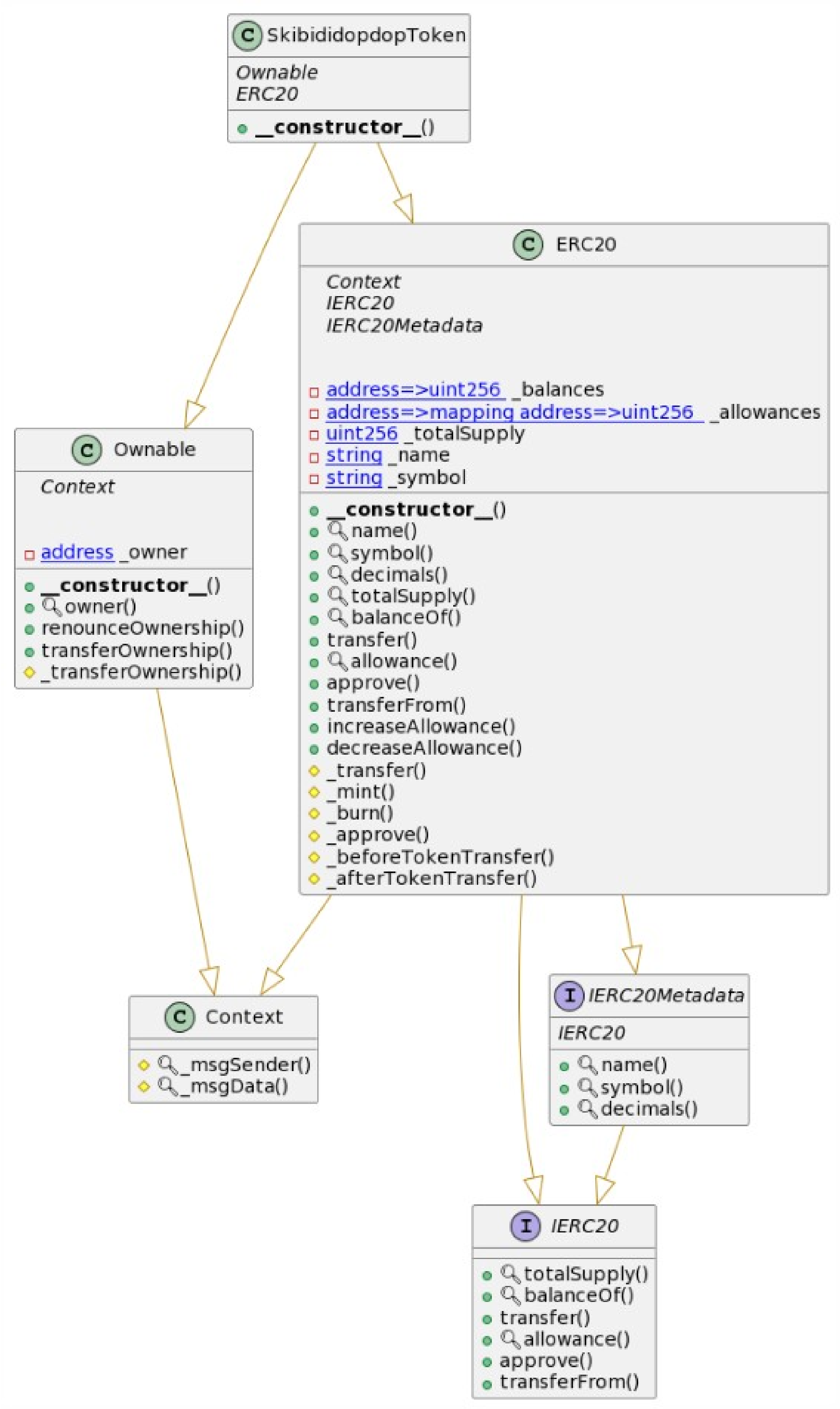


FUNCSIGS



```
Sighash   |   Function Signature
========================
39509351  =>  increaseAllowance(address,uint256)
119df25f  =>  _msgSender()
8b49d47e  =>  _msgData()
8da5cb5b  =>  owner()
715018a6  =>  renounceOwnership()
f2fde38b  =>  transferOwnership(address)
d29d44ee  =>  _transferOwnership(address)
18160ddd  =>  totalSupply()
70a08231  =>  balanceOf(address)
a9059cbb  =>  transfer(address,uint256)
dd62ed3e  =>  allowance(address,address)
095ea7b3  =>  approve(address,uint256)
23b872dd  =>  transferFrom(address,address,uint256)
06fdde03  =>  name()
95d89b41  =>  symbol()
313ce567  =>  decimals()
a457c2d7  =>  decreaseAllowance(address,uint256)
30e0789e  =>  _transfer(address,address,uint256)
4e6ec247  =>  _mint(address,uint256)
6161eb18  =>  _burn(address,uint256)
104e81ff  =>  _approve(address,address,uint256)
cad3be83  =>  _beforeTokenTransfer(address,address,uint256)
8f811a1c  =>  _afterTokenTransfer(address,address,uint256)
```

UML DIAGRAM



SkibididopdopToken
Ownable
ERC20
__constructor__()
ERC20
Context
IERC20
IERC20Metadata
address=>uint256 _balances
address=>mapping address=>uint256 _allowances
uint256 _totalSupply
string _name
string _symbol
__constructor__()
name()
symbol()
decimals()
totalSupply()
balanceOf()
transfer()
allowance()
approve()
transferFrom()
increaseAllowance()
decreaseAllowance()
_transfer()
_mint()
_burn()
_approve()
_beforeTokenTransfer()
_afterTokenTransfer()
Ownable
Context
address _owner
__constructor__()
owner()
renounceOwnership()
transferOwnership()
_transferOwnership()
Context
_msgSender()
_msgData()
IERC20Metadata
IERC20
name()
symbol()
decimals()
IERC20
totalSupply()
balanceOf()
transfer()
allowance()
approve()
transferFrom()

OVERALL

This is a Solidity smart contract for an ERC-20 token called "Skibididopdop". The contract inherits from the OpenZeppelin contracts `Ownable` and `ERC20` and implements the `IERC20Metadata` interface.

Here's a breakdown of the contract:

1. The contract defines the SPDX license identifier for the MIT license.
2. The contract provides an abstract `Context` contract that defines the execution context of the contract.
3. The contract inherits from the `Ownable` contract, which provides basic access control functionality, allowing for an owner to be set and for certain functions to be restricted to the owner.
4. The contract defines an interface `IERC20` that represents the ERC-20 standard functions.
5. The contract defines an interface `IERC20Metadata` that extends `IERC20` and adds optional metadata functions for the token name, symbol, and decimals.
6. The contract implements the `ERC20` contract, which is a basic implementation of the `IERC20` interface. It provides functionality for token transfers, allowances, and balance queries. The `name`, `symbol`, and `decimals` functions are overridden to provide the token-specific information.

Overall, this contract provides the basic functionality for an ERC-20 token and includes access control through the `Ownable` contract. It can be used as a starting point for creating a custom ERC-20 token with additional features.

Technical Disclaimer

Smart contracts are deployed and executed on blockchain platform. The platform, its programming language, and other software related to the smart contract can have its vulnerabilities that can lead to hacks. Thus, the audit can't guarantee the explicit security of the audited smart contracts.