Smart Contract

Security Assessment

For DOGEUM

29 Aug 2022

Ascendant
@ascendantproj
www.ascendant.finance

Table of Contents

DISCLAIMER

Ascendant Finance ("Ascendant") has conducted an independent audit to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the codes that were provided for the scope of this audit. This audit report does not constitute agreement, acceptance or advocation for the Project that was audited, and users relying on this audit report should not consider this as having any merit for financial advice in any shape, form or nature. The contracts audited do not account for any economic developments that may be pursued by the Project in question, and that the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are completely free of exploits, bugs, vulnerabilities or deprecation of technologies. Further, this audit report shall not be disclosed nor transmitted to any persons or parties on any objective, goal or justification without due written assent, acquiescence or approval by Ascendant.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence regardless of the findings presented in this report. Information is provided 'as is', and Ascendant is under no covenant to the completeness, accuracy or solidity of the contracts audited. In no event will Ascendant or its partners, employees, agents or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions and/or actions with regards to the information provided in this audit report.

Cryptocurrencies and any technologies by extension directly or indirectly related to cryptocurrencies are highly volatile and speculative by nature. All reasonable due diligence and safeguards may yet be insufficient, and users should exercise considerable caution when participating in any shape or form in this nascent industry.

The audit report has made all reasonable attempts to provide clear and articulate recommendations to the Project team with respect to the rectification, amendment and/or revision of any highlighted issues, vulnerabilities or exploits within the contracts provided. It is the sole responsibility of the Project team to sufficiently test and perform checks, ensuring that the contracts are functioning as intended, specifically that the functions therein contained within said contracts have the desired intended effects, functionalities and outcomes of the Project team.

Ascendant retains full rights over all intellectual property (including expertise and new attack or exploit vectors) discovered during the audit process. Ascendant is therefore allowed and expected to re-use this knowledge in subsequent audits and to inform existing projects that may have similar vulnerabilities. Ascendant may, at its discretion, claim bug bounties from third-parties while doing so.

Executive Summary

Severity	Found
🔴 High	0
🟠 Medium	0
🟡 Low	6
🟣 Informational	27
Total	33

We performed an independent technical audit to identify Smart Contracts uncertainties. This shall protect the code from illegitimate authorization attempts or external & internal threats of any type. This also ensures end-to-end proofing of the contract from frauds. The audit was performed semi-manually. We analyzed the Smart Contracts code line-by-line and used an automation tool to report any suspicious code.

The following tools were used:

- Truffle
- Remix IDE
- Slither

Overview

This report has been prepared for DOGEUM on the Binance network. Ascendant provides a user-centered examination of the smart contracts to look for vulnerabilities, logic errors or other issues from both an internal and external perspective.

Summary

Project Name	DOGEUM
Platform	Binance
Language	Solidity

Contracts Assessed

Name	**Location**
Dogeum.sol	0xE83981C6E294881D92697FdC887D19Acd9A820E3
Context.sol	In Dogeum Contract
Ownable.sol	In Dogeum Contract
IERC20.sol	In Dogeum Contract
IERC20Metadata.sol	In Dogeum Contract
ERC20.sol	In Dogeum Contract

Findings Summary

Severity	Found
🔴 High	0
🟠 Medium	0
🟡 Low	6
🟣 Informational	27
Total	33

Classification of Issues

🔴 High	Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, or impairment of the contract and its functions. Issues under this classification are recommended to be fixed with utmost urgency.
🟠 Medium	Bugs or issues that may be subject to exploit, though their impact is somewhat limited. Issues under this classification are recommended to be fixed as soon as possible.
🟡 Low	Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless.
🟣 Informational	Consistency, syntax or style best practices, Generally pose a negligible level of risk, if any.

Manual Review

Ascendant

Issues Checking Status

Issue Description	Checking Status
Compiler errors	PASS
Race conditions and Reentrancy. Cross-function race conditions.	PASS
Possible delays in data delivery.	PASS
Oracle calls.	PASS
Front running.	PASS
Timestamp dependence.	PASS
Integer Overflow and Underflow.	PASS
DoS with Revert.	PASS
DoS with block gas limit.	PASS
Methods execution permissions.	PASS
Economy model of the contract.	PASS
The impact of the exchange rate on the logic.	PASS
Private user data leaks.	PASS
Malicious Event log.	PASS
Scoping and Declarations.	PASS
Uninitialized storage pointers.	PASS

Arithmetic accuracy.	PASS
Design Logic.	PASS
Cross-function race conditions.	PASS
Safe Open Zeppelin contracts implementation and usage.	PASS
Fallback function security.	PASS

Audit Findings

Severity	**Low x 6**
Contract	Dogeum.sol
Description	Multiplication performed after division

Code Snippet	function _mintTokonomics() internal {
	_mint(_privateSalePool, ((_supply / 100) * 15));
	_mint(_gamePool, ((_supply / 100) * 10));
	_mint(_preSalePool, ((_supply / 100) * 60));
	_mint(_pancakePool, ((_supply / 100) * 10));
	_mint(_marketingPool, ((_supply / 100) * 2));
	_mint(_stakingPool, ((_supply / 100) * 3));
	}

Recommendation	Solidity integer division might truncate. As a result, performing multiplication before division can sometimes avoid loss of precision. Given that the division happens within the constructor and is not a function that will be often used, no mitigating action is necessary.
Status	ACKNOWLEDGED

Severity	Informational x 6
Contract	Dogeum.sol
Description	Lacks Zero-Check

Code Snippet	constructor(
	address preSale,
	address privateSalePool,
	address stakingPool,
	address gamePool,
	address pancakePool,
	address marketingPool
	) ERC20("Dogeum Token", "DOGEUM") {
	_preSalePool = preSale;
	_privateSalePool = privateSalePool;
	_stakingPool = stakingPool;
	_gamePool = gamePool;
	_pancakePool = pancakePool;
	_marketingPool = marketingPool;
	_mintTokonomics();
	}

Recommendation	Constructor does not check whether the addresses used as arguments are the zero-address, meaning a portion of the tokens could theoretically be stuck. However, this is unlikely if the deployer simply does not include the zero address as an argument. Therefore, additional require statements are not necessary to mitigate the potential harm.
Status	ACKNOWLEDGED

Functional Test Status

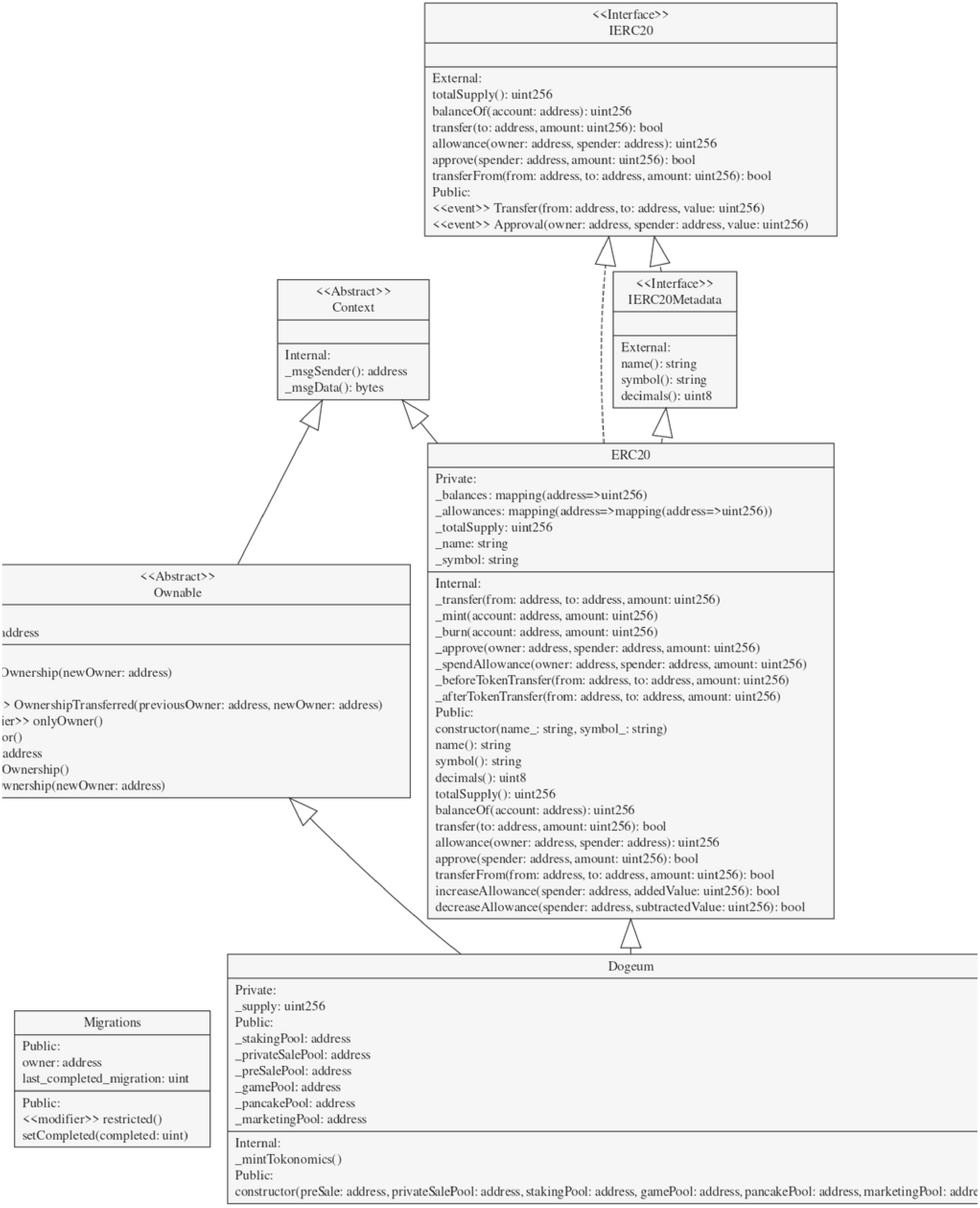
Function Name	Type/Return Type	Score
owner	read/public	PASS
renounceOwnership	write/public	PASS
transferOwnership	write/public	PASS
msgSender	internal	PASS
_msgData	internal	PASS
name	read/public	PASS
symbol	read/public	PASS
decimals	read/public	PASS
totalSupply	read/public	PASS
balanceOf	read/public	PASS
transfer	write/public	PASS
allowance	read/public	PASS
approve	write/public	PASS
transferFrom	write/public	PASS
increaseAllowance	write/public	PASS
decreaseAllowance	write/public	PASS
_transfer	internal	PASS
_mint	internal	PASS

_burn	internal	PASS
_approve	internal	PASS
_spendAllowance	internal	PASS
_beforeTokenTransfer	internal	PASS
_afterTokenTransfer	internal	PASS
_mintTokenomics	internal	PASS

Automated Review

Ascendant

Unified Modeling Language(UML)



<<Interface>>
IERC20

External:
totalSupply(): uint256
balanceOf(account: address): uint256
transfer(to: address, amount: uint256): bool
allowance(owner: address, spender: address): uint256
approve(spender: address, amount: uint256): bool
transferFrom(from: address, to: address, amount: uint256): bool
Public:
<<event>> Transfer(from: address, to: address, value: uint256)
<<event>> Approval(owner: address, spender: address, value: uint256)

<<Abstract>>
Context

Internal:
_msgSender(): address
_msgData(): bytes

<<Interface>>
IERC20Metadata

External:
name(): string
symbol(): string
decimals(): uint8

<<Abstract>>
Ownable

ddress

Ownership(newOwner: address)

> OwnershipTransferred(previousOwner: address, newOwner: address)
ier>> onlyOwner()
or()
address
Ownership()
wnership(newOwner: address)

ERC20

Private:
_balances: mapping(address=>uint256)
_allowances: mapping(address=>mapping(address=>uint256))
_totalSupply: uint256
_name: string
_symbol: string

Internal:
_transfer(from: address, to: address, amount: uint256)
_mint(account: address, amount: uint256)
_burn(account: address, amount: uint256)
_approve(owner: address, spender: address, amount: uint256)
_spendAllowance(owner: address, spender: address, amount: uint256)
_beforeTokenTransfer(from: address, to: address, amount: uint256)
_afterTokenTransfer(from: address, to: address, amount: uint256)
Public:
constructor(name_: string, symbol_: string)
name(): string
symbol(): string
decimals(): uint8
totalSupply(): uint256
balanceOf(account: address): uint256
transfer(to: address, amount: uint256): bool
allowance(owner: address, spender: address): uint256
approve(spender: address, amount: uint256): bool
transferFrom(from: address, to: address, amount: uint256): bool
increaseAllowance(spender: address, addedValue: uint256): bool
decreaseAllowance(spender: address, subtractedValue: uint256): bool

Migrations

Public:
owner: address
last_completed_migration: uint

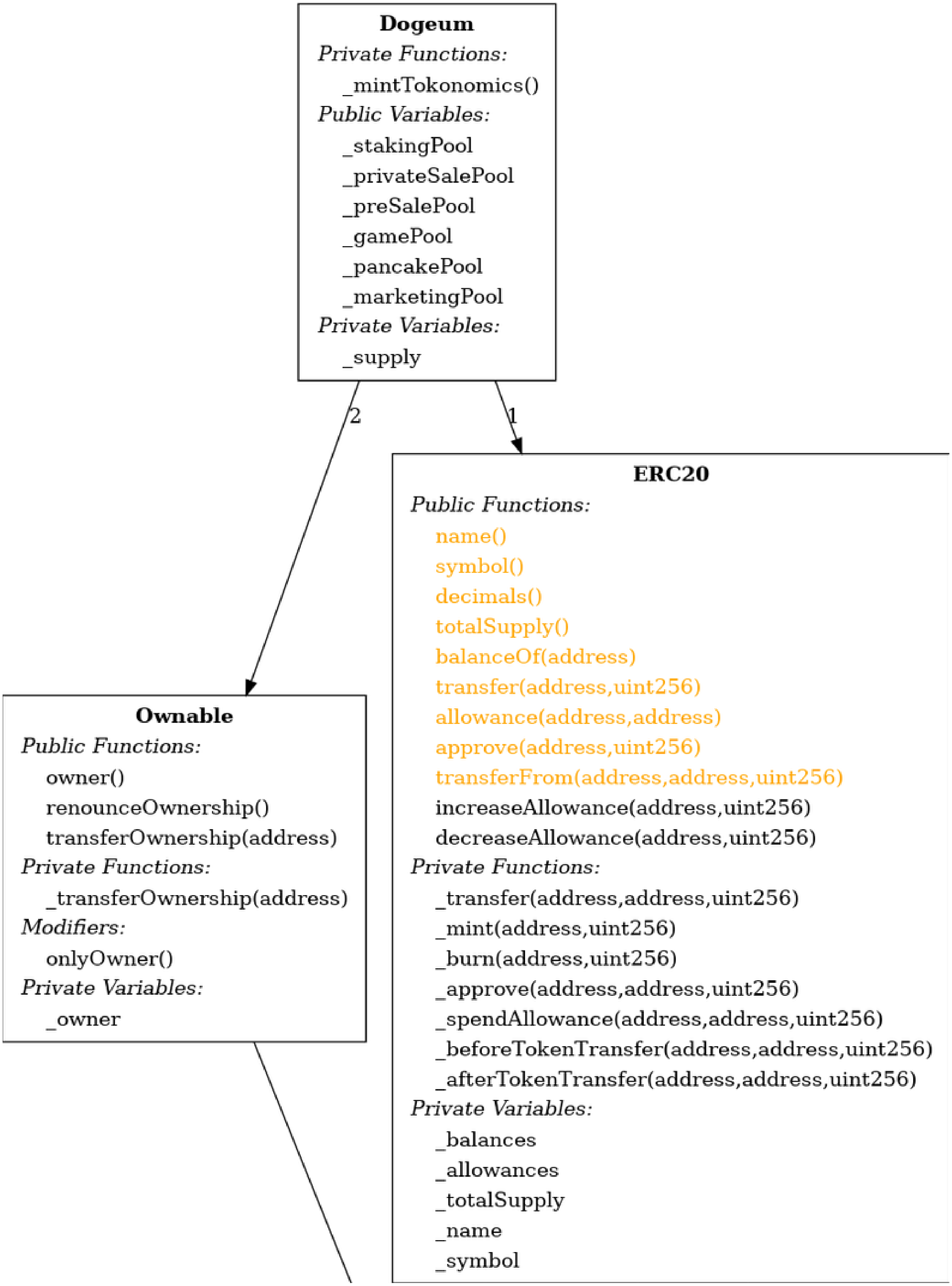
Public:
<<modifier>> restricted()
setCompleted(completed: uint)

Dogeum

Private:
_supply: uint256
Public:
_stakingPool: address
_privateSalePool: address
_preSalePool: address
_gamePool: address
_pancakePool: address
_marketingPool: address

Internal:
_mintTokonomics()
Public:
constructor(preSale: address, privateSalePool: address, stakingPool: address, gamePool: address, pancakePool: address, marketingPool: addre

Inheritance Graph



Conclusion

The smart contracts reviewed in this audit contain no critical severity issues and all Medium to Low issues have either been corrected or acknowledged.

Please check the disclaimer above and note, the audit makes no statements or warranties on business model, investment attractiveness or code sustainability. The report is provided for the only contract mentioned in the report and does not include any other potential contracts deployed by Owner.

Ascendant

Ascendant

@ascendantproj

www.ascendant.finance

Ascendant