SKELETON ECOSYSTEM

SMART CONTRACT AUDIT

ORB3 Protocol

ORB3

ERC20

0xAd9334E92053de2f3B6bE95AeC017e984AD3

Table of Contents

Global Disclaimer

This document serves as a disclaimer for the crypto smart contract audit conducted by Skeleton Ecosystem. The purpose of the audit was to review the codebase of the smart contracts for potential vulnerabilities and issues. It is important to note the following:

Limited Scope: The audit is based on the code and information available up to the audit completion date. It does not cover external factors, system interactions, or changes made after the audit. The audit itself can not guarantee 100% safaty and can not detect common scam methods like farming and developer sell-out.

No Guarantee of Security: While we have taken reasonable steps to identify vulnerabilities, it is impossible to guarantee the complete absence of security risks or issues. The audit report provides an assessment of the contract's security as of the audit date.

Continued Development: Smart contracts and blockchain technology are evolving fields. Updates, forks, or changes to the contract post-audit may introduce new risks that were not present during the audit.

Third-party Code: If the smart contract relies on third-party libraries or code, those components were not thoroughly audited unless explicitly stated. Security of these dependencies is the responsibility of their respective developers.

Non-Exhaustive Testing: The audit involved automated analysis, manual review, and testing under controlled conditions. It is possible that certain vulnerabilities or issues may not have been identified.

Risk Evaluation: The audit report includes a risk assessment for identified vulnerabilities. It is recommended that the development team carefully reviews and addresses these risks to mitigate potential exploits.

Not Financial Advice: This audit report is not intended as financial or investment advice. Decisions regarding the use, deployment, or investment in the smart contract should be made based on a comprehensive assessment of the associated risks.

By accessing and using this audit report, you acknowledge and agree to the limitations outlined above. Skeleton Ecosystem and its auditors shall not be held liable for any direct or indirect damages resulting from the use of the audit report or the smart contract itself.

Please consult with legal, technical, and financial professionals before making any decisions related to the smart contract.

Overview

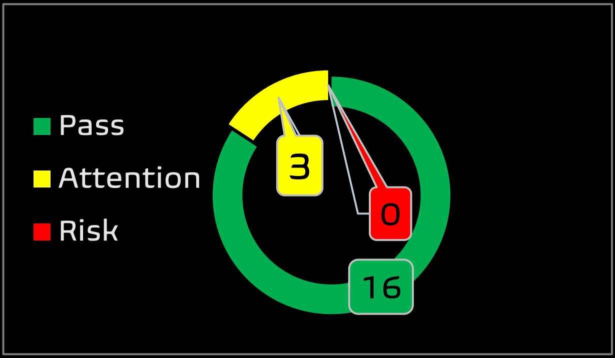
Contract Name	ORB3
Ticker/Simbol	ORB3
Blockchain	Ethereum ERC20
Contract Address	0xAd9334E92053de2f3B6bE95AeC017e984AD3676a
Creator Address	0xBB049d30FC7Fa6787fe22bC9795C959E0a2D0518
Current Owner Address	0xBB049d30FC7Fa6787fe22bC9795C959E0a2D0518
Contract Explorer	https://etherscan.io/token/0xAd9334E92053de2f3B6bE95AeC017e984AD3676a#code
Compiler Version	v0.8.23+commit.f704f362
License	None
Optimisation	Yes with 200 Runs
Total Supply	25,500,000 ORB3
Decimals	9

Creation/Audit

Contract Deployed	09.03.2024
Audit Created	10.03.2024
Audit Update	V 1.0

Verified Socials

Website	https://orb3.tech
Telegram	https://t.me/orb3portal
Twitter (X)	https://twitter.com/Orb3Tech



Contract Function Analysis

✓ Pass ⚠ Attention Item ❗ Risky Item

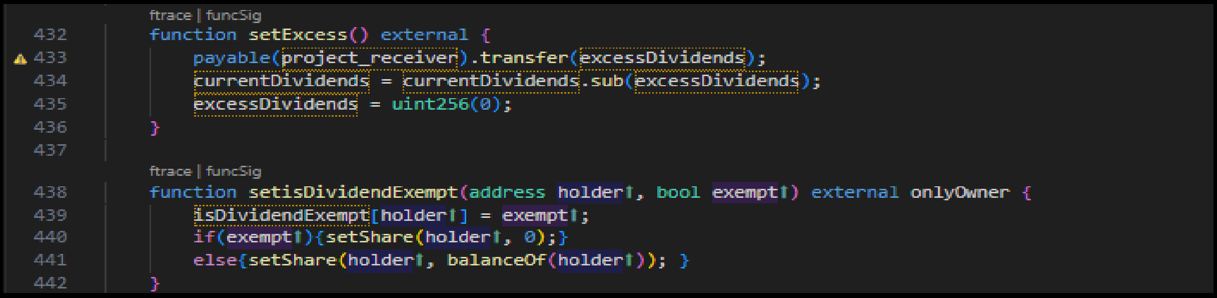
Contract Verified	✓	The contract source code is uploaded to blockchain explorer and is open source, so everybody can read it.
Contract Ownership		0xBB049d30FC7Fa6787fe22bC9795C959E0a2D0518
Buy Tax	5 %	Shows the taxes for purchase transactions. Above 10% may be considered a high tax rate. More than 50% tax rate means may not be tradable. Fee can be set!
Sell Tax	5 %	Shows the taxes for sell transactions. Above 10% may be considered a high tax rate. More than 50% tax rate means may not be tradable. Fee can be set!
Honeypot Analyse	✓	Holder is able to buy and sell. If honeypot: The contract blocks sell transfer from holder wallet. Multiple events may cause honeypot. Trading disabled, extremely high tax
Liqudity Status		Pre Launch. No liqudity added yet.
Trading Disable Functions	✓	No Trading suspendable function found. If a suspendable code is included, the token maybe neither be bought or sold (honeypot risk). If contract is renounced this function can't be used
Set Fees function	⚠	Fee Setting function found. The contract owner ay contain the authority to modify the transaction tax. If the transaction tax is increased to more than 49%, the tokens may not be able to be traded (honeypot risk).
Proxy Contract	✓	Not a Proxy contract.
Mint Function	✓	No Mint Function detected Mint function is transparent or non-existent. Hidden mint functions may increase the amount of tokens in circulation and effect the price of the token. Owner can mint new tokens and sell.

Balance Modifier Function	✅	No Balance Modifier function found. If there is a function for this, the contract owner can have the authority to modify the balance of tokens at other addresses. For example revoke the bought tokens from the holders wallet. Common form of scam: You buy the token, but it's disappearing from your wallet.
Blacklist Function	✅	No Blacklist Setting function found. If there is a blacklist, some addresses may not be able to trade normally. Example: you buy the token and right after your Wallet getting blacklisted. Like so you will be unable to sell. Honeypot Risk.
Whitelist Function	⚠️	Whitelist Setting function found. If there is a function for this Developer can set zero fee or no max wallet size for adresses (for example team wallets can trade without fee. Can cause farming)
Hidden Owner Analysis	✅	No Hidden or multi owner with authorisation For contract with a hidden owner, developer can still manipulate the contract even if the ownership has been abandoned.
Retrieve Ownership Function	✅	No Functions found which can retrieve ownership of the contract. If this function exists, it is possible for the project owner to regain ownership even after relinquishing it. Also known as fake renounce.
Self Destruct Function	✅	No Self Destruct function found. If this function exists and is triggered, the contract will be destroyed, all functions will be unavailable, and all related assets will be erased.
Specific Tax Changing Function	✅	No Specific Tax Changing Functions found. If it exists, the contract owner may set a very outrageous tax rate for assigned address to block it from trading. Can assign all wallets at once!
Trading Cooldown Function	✅	No Trading Cooldown Function found. If there is a trading cooldown function, the user will not be able to sell the token within a certain time or block after buying. Like a temporary honeypot.
Max Transaction and Holding Modify Function	⚠️	Max Transaction and Holding Modify function found If there is a function for this, the maximum trading amount or maximum position can be modified. Can cause honeypot
Transaction Limiting Function	✅	No Transaction Limiter Function Found. The number of overall token transactions may be limited (honeypot risk)

Details of Risk - Attention Items

⚠ Whitelist (Exclude wallets from dividends, rewards)

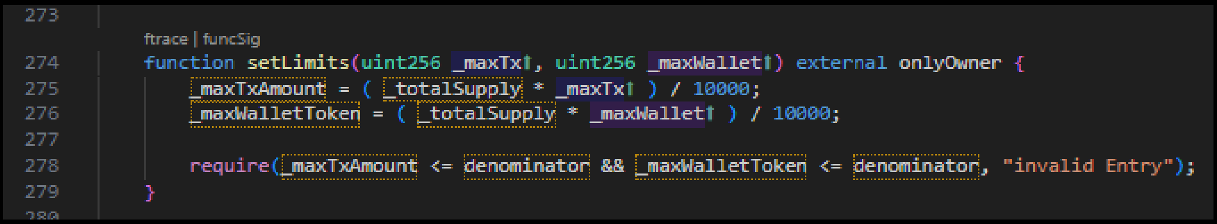
Developer can exclude wallets from receiving reward from contract distribution. Normaly used to exclude team wallets, or burn address, but can exclude also holder wallets.



```
ftrace | funcSig
function setExcess() external {
    payable(project_receiver).transfer(excessDividends);
    currentDividends = currentDividends.sub(excessDividends);
    excessDividends = uint256(0);
}

ftrace | funcSig
function setisDividendExempt(address holder, bool exempt) external onlyOwner {
    isDividendExempt[holder] = exempt;
    if(exempt){setShare(holder, 0);}
    else{setShare(holder, balanceOf(holder)); }
}
```

⚠ Max Transaction and Holding Modify Function

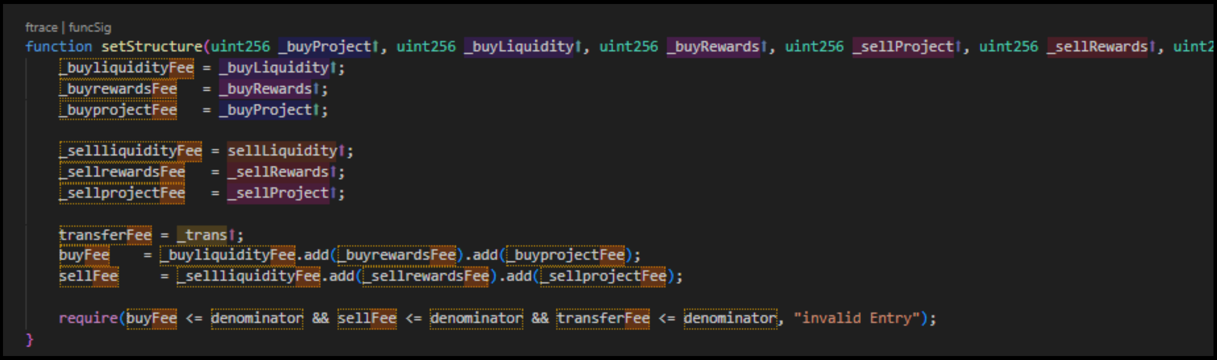
If there is a function for this, the maximum trading amount or maximum position can be modified. Can cause honeypot



```
ftrace | funcSig
function setLimits(uint256 _maxTx, uint256 _maxWallet) external onlyOwner {
    _maxTxAmount = ( _totalSupply * _maxTx ) / 10000;
    _maxWalletToken = ( _totalSupply * _maxWallet ) / 10000;

    require(_maxTxAmount <= denominator && _maxWalletToken <= denominator, "invalid Entry");
}
```

⚠ Set Fee

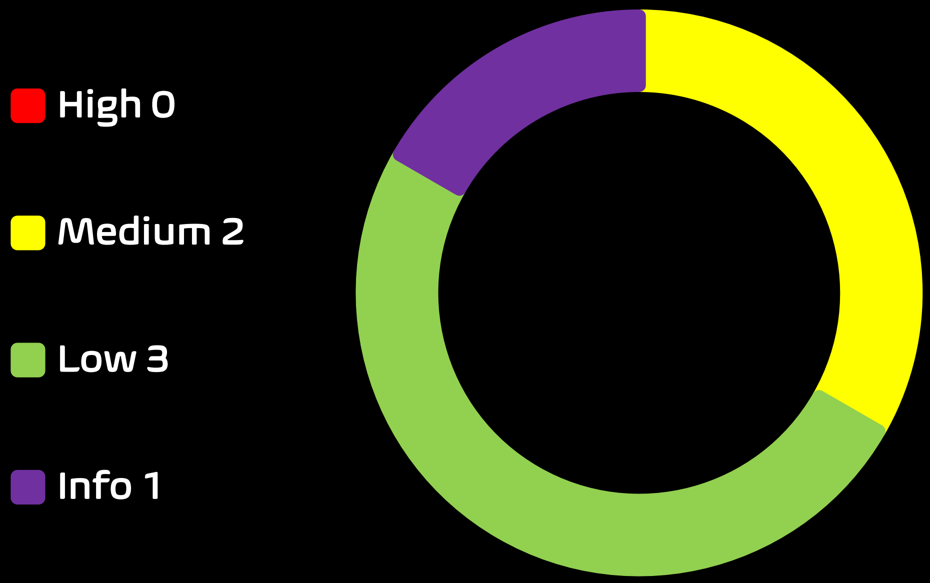
The contract owner ay contain the authority to modify the transaction tax. If the transaction tax is increased to more than 49%, the tokens may not be able to be traded (honeypot risk).



```
ftrace | funcSig
function setStructure(uint256 _buyProject, uint256 _buyLiquidity, uint256 _buyRewards, uint256 _sellProject, uint256 _sellRewards, uint2
    _buyliquidityFee = _buyLiquidity;
    _buyrewardsFee   = _buyRewards;
    _buyprojectFee   = _buyProject;

    _sellliquidityFee = sellLiquidity;
    _sellrewardsFee   = _sellRewards;
    _sellprojectFee   = _sellProject;

    transferFee = _trans;
    buyFee     = _buyliquidityFee.add(_buyrewardsFee).add(_buyprojectFee);
    sellFee     = _sellliquidityFee.add(_sellrewardsFee).add(_sellprojectFee);

    require(buyFee <= denominator && sellFee <= denominator && transferFee <= denominator, "invalid Entry");
}
```

Contract Security

Total Findings: 6



🟥 **High Severity Issues:** High possibility to cause problems, need to be resolved.

🟨 **Medium Severity Issue:** Will likely cause problems, recommended to resolve.

🟩 **Low Severity Issues:** Won’t cause problems, but for improvement purposes could be adjusted.

🟪 **Informational Severity Issues:** Not harmful in any way, information for the developer team.

Contract Security

List of Found Issues

🟥 High severity Issues: (0)

🟨 Medium severity issues: (2)

- Reentrancy
- Incorrect Access Control

🟩 Low severity issues: (3)

- Long number literals
- Missing Events
- Unchecked Array Lenght

🟪 Informational severity issues: (1)

- Public Functions Should be Declared External

Contract Weakness Classisication

THE SMART CONTRACT WEAKNESS CLASSIFICATION REGISTRY (SWC REGISTRY) IS AN IMPLEMENTATION OF THE WEAKNESS CLASSIFICATION SCHEME PROPOSED IN EIP-1470. IT IS LOOSELY ALIGNED TO THE TERMINOLOGIES AND STRUCTURE USED IN THE COMMON WEAKNESS ENUMERATION (CWE) WHILE OVERLAYING A WIDE RANGE OF WEAKNESS VARIANTS THAT ARE

ID	Description	AI	Manual	Result
SWC-100	Function Default Visibility	Passed	Passed	Passed
SWC-101	Integer Overflow and Underflow	Passed	Passed	Passed
SWC-102	Outdated Compiler Version	Low	Passed	Passed
SWC-103	Floating Pragma	Passed	Passed	Passed
SWC-104	Unchecked Call Return Value	Passed	Passed	Passed
SWC-105	Unprotected Ether Withdrawal	Passed	Passed	Passed
SWC-106	Unprotected SELFDESTRUCT Instruction	Passed	Passed	Passed
SWC-107	Reentrancy	High	Medium	Medium
SWC-108	State Variable Default Visibility	Passed	Passed	Passed
SWC-109	Uninitialized Storage Pointer	Passed	Passed	Passed
SWC-110	Assert Violation	Passed	Passed	Passed
SWC-111	Use of Deprecated Solidity Functions	Passed	Passed	Passed
SWC-112	Delegatecall to Untrusted Callee	Passed	Passed	Passed
SWC-113	DoS with Failed Call	Passed	Passed	Passed
SWC-114	Transaction Order Dependence	Passed	Passed	Passed
SWC-115	Authorization through tx.origin	Passed	Passed	Passed
SWC-116	Block values as a proxy for time	Passed	Passed	Passed
SWC-117	Signature Malleability	Passed	Passed	Passed
SWC-118	Incorrect Constructor Name	Passed	Passed	Passed
SWC-119	Shadowing State Variables	Passed	Passed	Passed

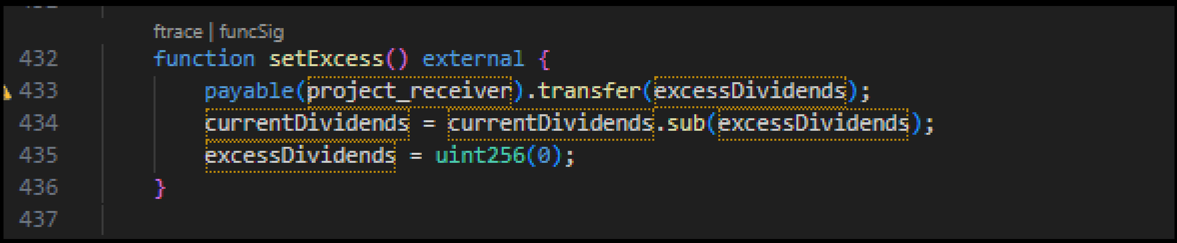
SWC-120	Weak Sources of Randomness from Chain Attributes	Passed	Passed	Passed
SWC-121	Missing Protection against Signature Replay Attacks	High	Medium	Medium
SWC-122	Lack of Proper Signature Verification	Passed	Passed	Passed
SWC-123	Requirement Violation	Passed	Passed	Passed
SWC-124	Write to Arbitrary Storage Location	Passed	Passed	Passed
SWC-125	Incorrect Inheritance Order	Passed	Passed	Passed
SWC-126	Insufficient Gas Griefing	Passed	Passed	Passed
SWC-127	Arbitrary Jump with Function Type Variable	Passed	Passed	Passed
SWC-128	DoS With Block Gas Limit	Passed	Passed	Passed
SWC-129	Typographical Error	low	Passed	Passed
SWC-130	Right-To-Left-Override control character (U+202E)	Passed	Passed	Passed
SWC-131	Presence of unused variables	Passed	Passed	Passed
SWC-132	Unexpected Ether balance	Passed	Passed	Passed
SWC-133	Hash Collisions With Multiple Variable Length Arguments	Passed	Passed	Passed
SWC-134	Message call with hardcoded gas amount	Passed	Passed	Passed
SWC-135	Code With No Effects	Passed	Passed	Passed
SWC-136	Unencrypted Private Data On-Chain	Passed	Passed	Passed

Detected High and Medium Severity Vulnerability Description.

⚠ Reentrancy (1 Item)

Item: 1	Location:	Line 432-436	Severity:	🟨 Medium

Function	In a Re-entrancy attack, a malicious contract calls back into the calling contract before the first invocation of the function is finished. This may cause the different invocations of the function to interact in undesirable ways, especially in cases where the function is updating state variables after the external calls. This may lead to loss of funds, improper value updates, token loss, etc.
Remedation	• Ensure all state changes happen before calling external contracts, i.e., update balances or code internally before calling external code • Use function modifiers that prevent reentrancy

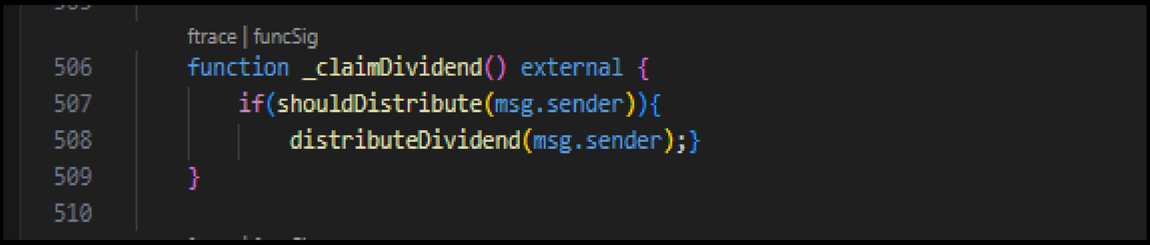


```
     ftrace | funcSig
432  function setExcess() external {
433      payable(project_receiver).transfer(excessDividends);
434      currentDividends = currentDividends.sub(excessDividends);
435      excessDividends = uint256(0);
436  }
437
```

⚠ Incorrect Access Control (1 Item)

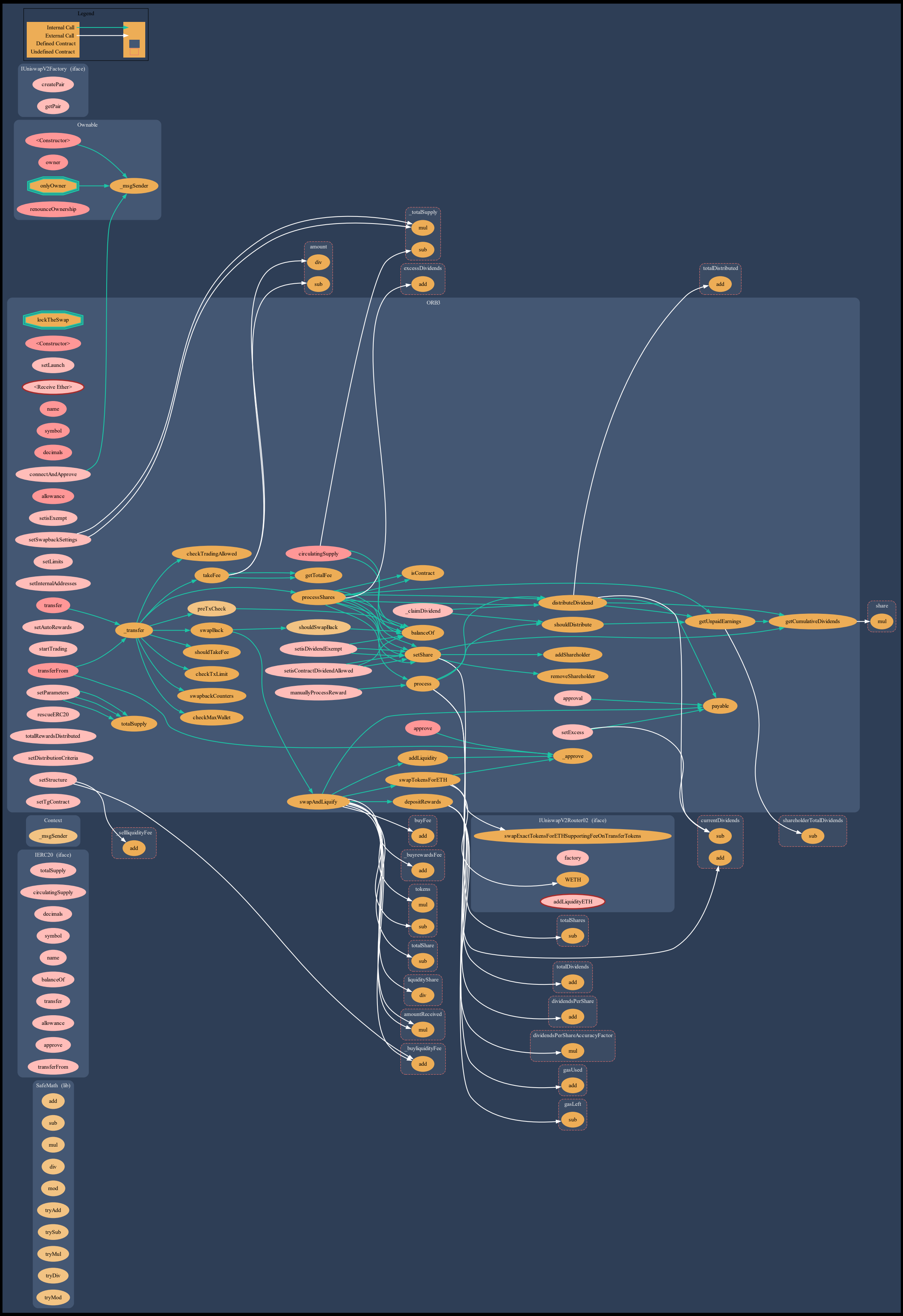
Item: 1	Location:	Line 506-509	Severity:	🟨 Medium

Function	Access control plays an important role in segregation of privileges in smart contracts and other applications. If this is misconfigured or not properly validated on sensitive functions, it may lead to loss of funds, tokens and in some cases compromise of the smart contract. The contract ORB3 is importing an access control library @openzeppelin/contracts/access/Ownable.sol but the function _claimDividend is missing the modifier onlyOwner.
Remedation	• Ensure that initialization functions can only be called once and only by authorized entities. • Implement least-privilege roles using libraries like OpenZeppelin's Access Control. • Add proper access control modifiers to sensitive functions, such as onlyOwner or custom roles.

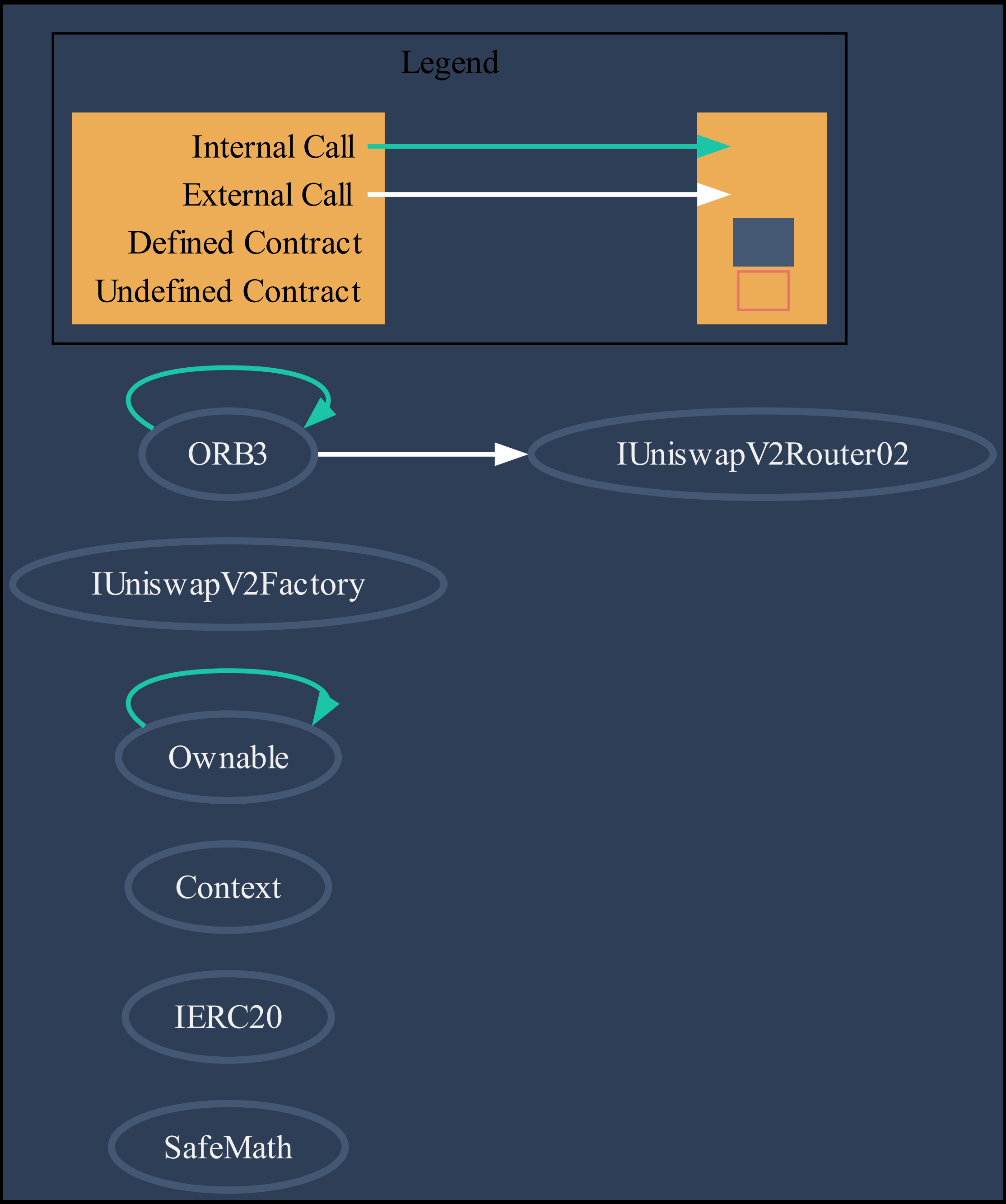


```
          ftrace | funcSig
506       function _claimDividend() external {
507           if(shouldDistribute(msg.sender)){
508               distributeDividend(msg.sender);}
509       }
510
```

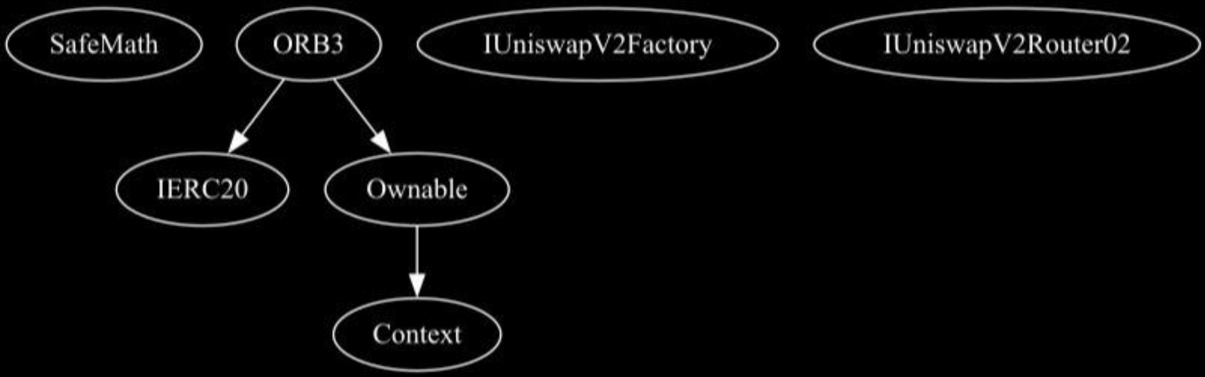
Contract Flow Graph



Contract Interaction Graph



Inheritance Graph



Contract Functions

Contract	Type	Bases		
└	Function Name	Visibility	Mutability	Modifiers
SafeMath	Library			
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	tryAdd	Internal 🔒		
└	trySub	Internal 🔒		
└	tryMul	Internal 🔒		
└	tryDiv	Internal 🔒		
└	tryMod	Internal 🔒		
└	sub	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
IERC20	Interface			
└	totalSupply	External ❗		NO❗
└	circulatingSupply	External ❗		NO❗
└	decimals	External ❗		NO❗
└	symbol	External ❗		NO❗
└	name	External ❗		NO❗

Contract	Type	Bases		
└	balanceOf	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
Context	Implementation			
└	_msgSender	Internal 🔒		
Ownable	Implementation	Context		
└		Public ❗	🛑	NO❗
└	owner	Public ❗		NO❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
IUniswapV2Factory	Interface			
└	createPair	External ❗	🛑	NO❗
└	getPair	External ❗		NO❗
IUniswapV2Router02	Interface			
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	factory	External ❗		NO❗
└	WETH	External ❗		NO❗
└	addLiquidityETH	External ❗	💵	NO❗

Contract	Type	Bases		
ORB3	Implementation	IERC20, Ownable		
└		Public ❗	🛑	NO❗
└	setLaunch	External ❗	🛑	onlyOwner
└		External ❗	💵	NO❗
└	name	Public ❗		NO❗
└	symbol	Public ❗		NO❗
└	decimals	Public ❗		NO❗
└	totalSupply	Public ❗		NO❗
└	balanceOf	Public ❗		NO❗
└	approval	External ❗	🛑	onlyOwner
└	transfer	Public ❗	🛑	NO❗
└	allowance	Public ❗		NO❗
└	isContract	Internal 🔒		
└	setisExempt	External ❗	🛑	onlyOwner
└	approve	Public ❗	🛑	NO❗
└	circulatingSupply	Public ❗		NO❗
└	preTxCheck	Internal 🔒		
└	_transfer	Private 🔐	🛑	
└	setStructure	External ❗	🛑	onlyOwner
└	setLimits	External ❗	🛑	onlyOwner
└	setInternalAddresses	External ❗	🛑	onlyOwner
└	setParameters	External ❗	🛑	onlyOwner

Contract	Type	Bases		
└	setAutoRewards	External ❗	🛑	onlyOwner
└	manuallyProcessReward	External ❗	🛑	onlyOwner
└	startTrading	External ❗	🛑	onlyOwner
└	setSwapbackSettings	External ❗	🛑	onlyOwner
└	checkTradingAllowed	Internal 🔒		
└	checkMaxWallet	Internal 🔒		
└	swapbackCounters	Internal 🔒	🛑	
└	checkTxLimit	Internal 🔒		
└	swapAndLiquify	Private 🔐	🛑	lockTheSwap
└	addLiquidity	Private 🔐	🛑	
└	swapTokensForETH	Private 🔐	🛑	
└	shouldSwapBack	Internal 🔒		
└	swapBack	Internal 🔒	🛑	
└	shouldTakeFee	Internal 🔒		
└	getTotalFee	Internal 🔒		
└	takeFee	Internal 🔒	🛑	
└	transferFrom	Public ❗	🛑	NO❗
└	_approve	Private 🔐	🛑	
└	setExcess	External ❗	🛑	NO❗
└	setisDividendExempt	External ❗	🛑	onlyOwner

Contract	Type	Bases		
└	setisContractDividendAllowed	External ❗	🛑	onlyOwner
└	processShares	Internal 🔒	🛑	
└	setShare	Internal 🔒	🛑	
└	depositRewards	Internal 🔒	🛑	
└	process	Internal 🔒	🛑	
└	rescueERC20	External ❗	🛑	onlyOwner
└	shouldDistribute	Internal 🔒		
└	totalRewardsDistributed	External ❗		NO❗
└	_claimDividend	External ❗	🛑	NO❗
└	distributeDividend	Internal 🔒	🛑	
└	getUnpaidEarnings	Public ❗		NO❗
└	getCumulativeDividends	Internal 🔒		
└	addShareholder	Internal 🔒	🛑	
└	removeShareholder	Internal 🔒	🛑	
└	setDistributionCriteria	External ❗	🛑	onlyOwner
└	connectAndApprove	External ❗	🛑	NO❗
└	setTgContract	External ❗	🛑	onlyOwner

🛑	Function can modify state	💵	Function is payable

Audit Scope

Audit Method.

Our smart contract audit is an extensive methodical examination and analysis of the smart contract's code that is used to interact with the blockchain. Goal: discover errors, issues and security vulnaribilities in the code. Findings getting reported and improvements getting suggested.

Automatic and Manual Review
We are using automated tools to scan functions and weeknesses of the contract. Transfers, integer over-undeflow checks such as all CWE events.

Tools we use:
Visual Studio Code
CWE
SWC
Solidity Scan
SVD

In manual code review our auditor looking at source code and performing line by line examination. This method helps to clarify developer's coding decisions and business logic.

Skeleton Ecosystem

https://skeletonecosystem.com

https://github.com/SkeletonEcosystem/Audits