SKELETON ECOSYSTEM

SMART CONTRACT AUDIT

eXtreme Beans [eXtreme]

BEP 20

0xc91b3b4bc5e91CF5b655b5DCFa5F84b43A3665df

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	eXtremeBeans
Ticker/Simbol	eXtreme
Blockchain	Binance Smart Chain BEP20
Contract Address	0xc91b3b4bc5e91CF5b655b5DCFa5F84b43A3665df
Creator Address	0x3e08993e96c78009b483D454610aa648d641bA01
Current Owner Address	0x3e08993e96c78009b483D454610aa648d641bA01
Contract Explorer	https://bscscan.com/token/0xc91b3b4bc5e91cf5b655b5dcfa5f84b43a3665df
Compiler Version	v0.8.19+commit.7dd6d404
License	MIT
Optimisation	Yes with 10000 Runs
Total Supply	750,000 **eXtreme**
Decimals	18

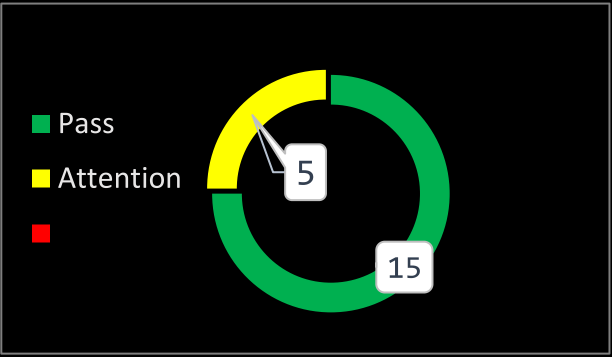
Creation/Audit

Contract Deployed	18 Oct 2023
Audit Created	24 Oct 2023
Audit Update	V 1.0

Verified Socials

Website	https://extremebeans.live/
Telegram	https://t.me/eXtremeBeans
Twitter (X)	https://x.com/eXtremeBeansX

EXTREME BEANS BEP20



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	⚠	0x3e08993e96c78009b483D454610aa648d641bA01
Buy Tax	4 %	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	4 %	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	✓	LP Lock Status on 24.10.2023: Lp Locked: 89.39% Pinklock for 361 days. Lp Locked: 5.12% Pinklock for 357 days. Lp Burned: 5.36%
Trading Disable Function	✓	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	⚠ 4% max	Fee Setting function found The contract owner may 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.

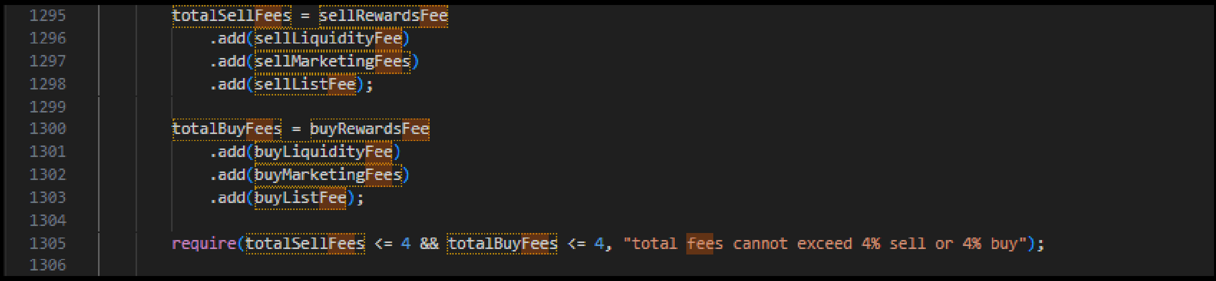
Balance Modifier Function	✓	No Balance Modifier function found. If there is a function for this, the contract owner can have the authority to modify the balance of tokens at other addresses. For example revoke the bought tokens from the holders wallet. Common form of scam: You buy the token, but it's disappearing from your wallet.
Blacklist Function	✓	No Blacklist Setting function 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 but Contract 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 Owner found For contract with a hidden owner, developer can still manipulate the contract even if the ownership has been abandoned. Fake renounce.
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	⚠	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

⚠ Set Fee (remedation: renounce ownership)

The contract owner may 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

Lowering risk: 4% max fee setting function found



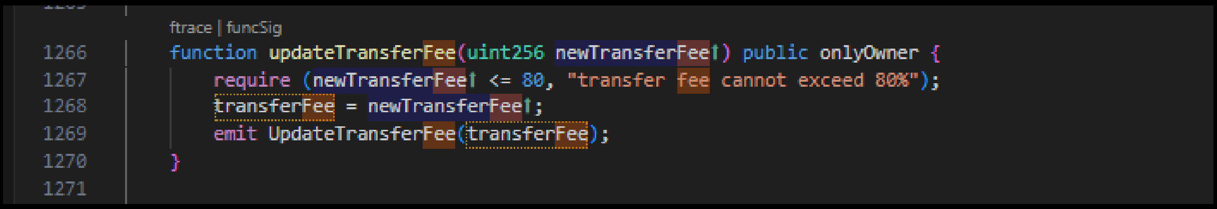
```
        totalSellFees = sellRewardsFee
            .add(sellLiquidityFee)
            .add(sellMarketingFees)
            .add(sellListFee);

        totalBuyFees = buyRewardsFee
            .add(buyLiquidityFee)
            .add(buyMarketingFees)
            .add(buyListFee);

        require(totalSellFees <= 4 && totalBuyFees <= 4, "total fees cannot exceed 4% sell or 4% buy");
```

⚠ Set Transfer Fee (remedation: renounce ownership)

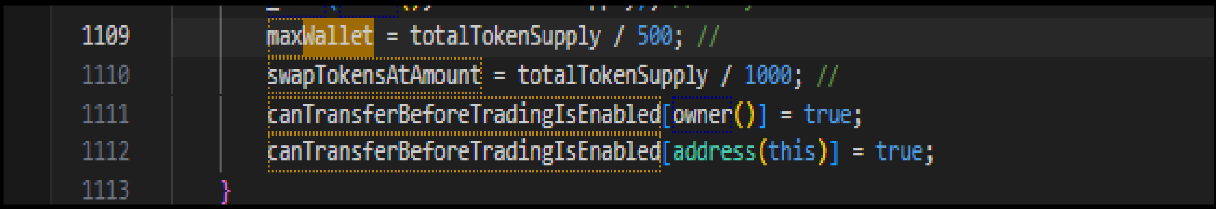
The transfer between wallets has a high fee! 80% (Max 80% setting)



```
    ftrace | funcSig
    function updateTransferFee(uint256 newTransferFee) public onlyOwner {
        require (newTransferFee <= 80, "transfer fee cannot exceed 80%");
        transferFee = newTransferFee;
        emit UpdateTransferFee(transferFee);
    }
```

⚠ Max Transaction and Holding Modify Function. Max transfer and wallet can set to zero! (remedation: renounce ownership)

If there is a function for this, the maximum trading amount or maximum position can be modified. At zero transfer ammount, its honeypot riks



```
        maxWallet = totalTokenSupply / 500; //
        swapTokensAtAmount = totalTokenSupply / 1000; //
        canTransferBeforeTradingIsEnabled[owner()] = true;
        canTransferBeforeTradingIsEnabled[address(this)] = true;
    }
```

Contract Security

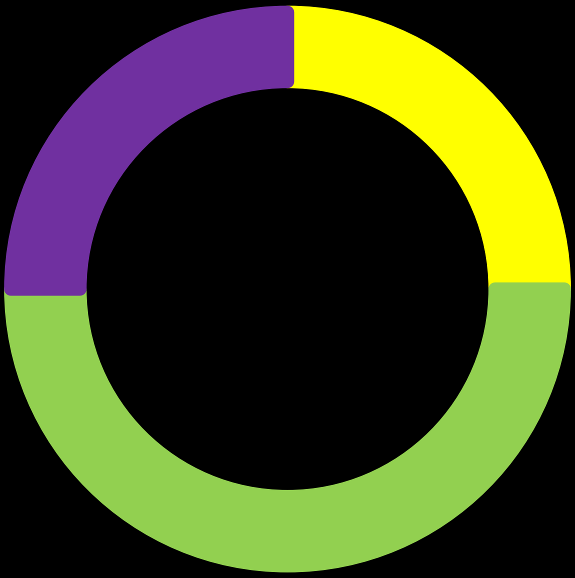
Total Findings: 8

🟥 **High 0**

🟨 **Medium 2**

🟩 **Low 4**

🟪 **Info 2**



🟥 **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)**

- TX Origin used
- Approve front running attack

🟩 **Low severity issues: (4)**

- Missing Events
- Long Number Literals
- Floating Pragma
- Dos with failed Call

🟪 **Informational severity issues: (2)**

- Hard Coded Address
- 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 SPECIFIC TO SMART CONTRACTS.

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	Passed	Passed	Passed
SWC-103	Floating Pragma	Low	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	Passed	Passed	Passed
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	Medium	Low	Low
SWC-114	Transaction Order Dependence	Passed	Passed	Passed
SWC-115	Authorization through tx.origin	High	Medium	Medium
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	**Passed**	**Passed**	**Passed**
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.

⚠ TX Origin Used (6 Items)

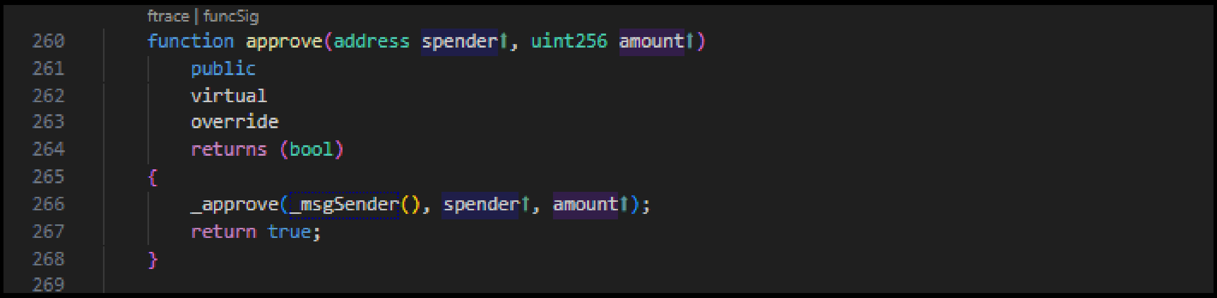
Item: 1	Location:	Line 1395	Severity:	🟨 Medium
Item: 2	Location:	Line 1489	Severity:	🟨 Medium
Item: 3	Location:	Line 1491	Severity:	🟨 Medium
Item: 4	Location:	Line 1505	Severity:	🟨 Medium
Item: 5	Location:	Line 1506	Severity:	🟨 Medium
Item: 6	Location:	Line 1586	Severity:	🟨 Medium

Function	In Solidity, tx.origin is a global variable that returns the address of the account that sent the transaction. Using the variable for authorization could make a contract vulnerable. For example, if an authorized account calls a malicious contract which triggers it to call the vulnerable contract that passes an authorization check since tx.origin returns the original sender of the transaction which in this case is the authorized account.
Remedation	tx.origin should not be used for authorization in smart contracts. It does have some legitimate use cases, for example, To prevent external contracts from calling the current contract, you can implement a require of the form require(tx.origin == msg.sender). This prevents intermediate contracts from calling the current contract, thus limiting the contract to regular codeless addresses.

⚠ Approve of Front running Attack. Example Sandwitch bots (2 Items)

Item: 1	Location:	Line 260-268	Severity:	🟨 Medium

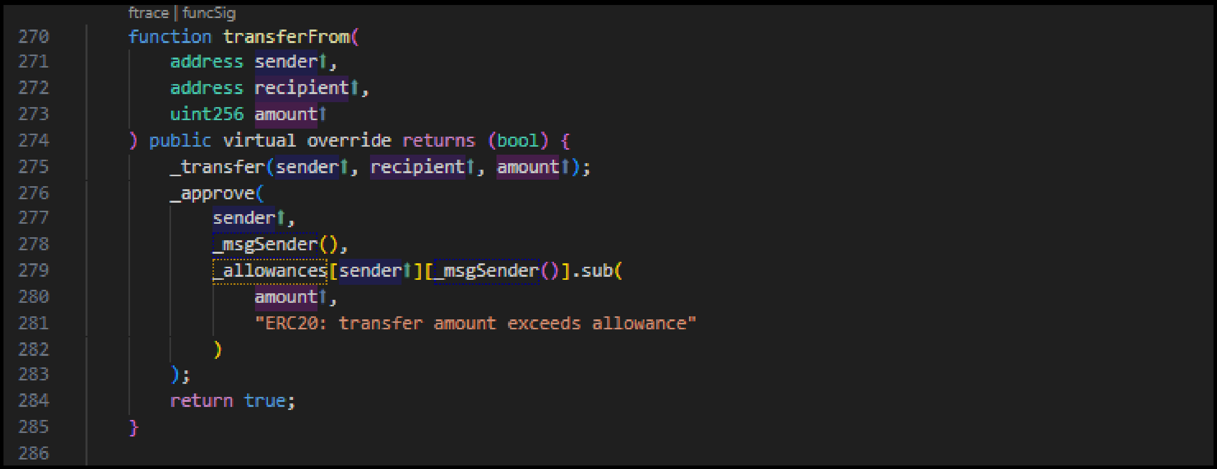
Function	The approve() method overrides current allowance regardless of whether the spender already used it or not, so there is no way to increase or decrease allowance by a certain value atomically unless the token owner is a smart contract, not an account. This can be abused by a token receiver when they try to withdraw certain tokens from the sender's account. Meanwhile, if the sender decides to change the amount and sends another approve transaction, the receiver can notice this transaction before it's mined and can extract tokens from both the transactions, therefore, ending up with tokens from both the transactions. This is a front-running attack affecting the ERC20 Approve function. The function approve can be front-run by abusing the _approve function.
Remedation	1. Introduce mechanisms that limit the maximum acceptable gas price for transactions. This can help prevent front-runners from drastically increasing the gas fees to prioritize their transactions. 2. Use transaction taxes to prevent against front-runattack



```
        ftrace | funcSig
260     function approve(address spender↑, uint256 amount↑)
261         public
262         virtual
263         override
264         returns (bool)
265     {
266         _approve(_msgSender(), spender↑, amount↑);
267         return true;
268     }
269
```

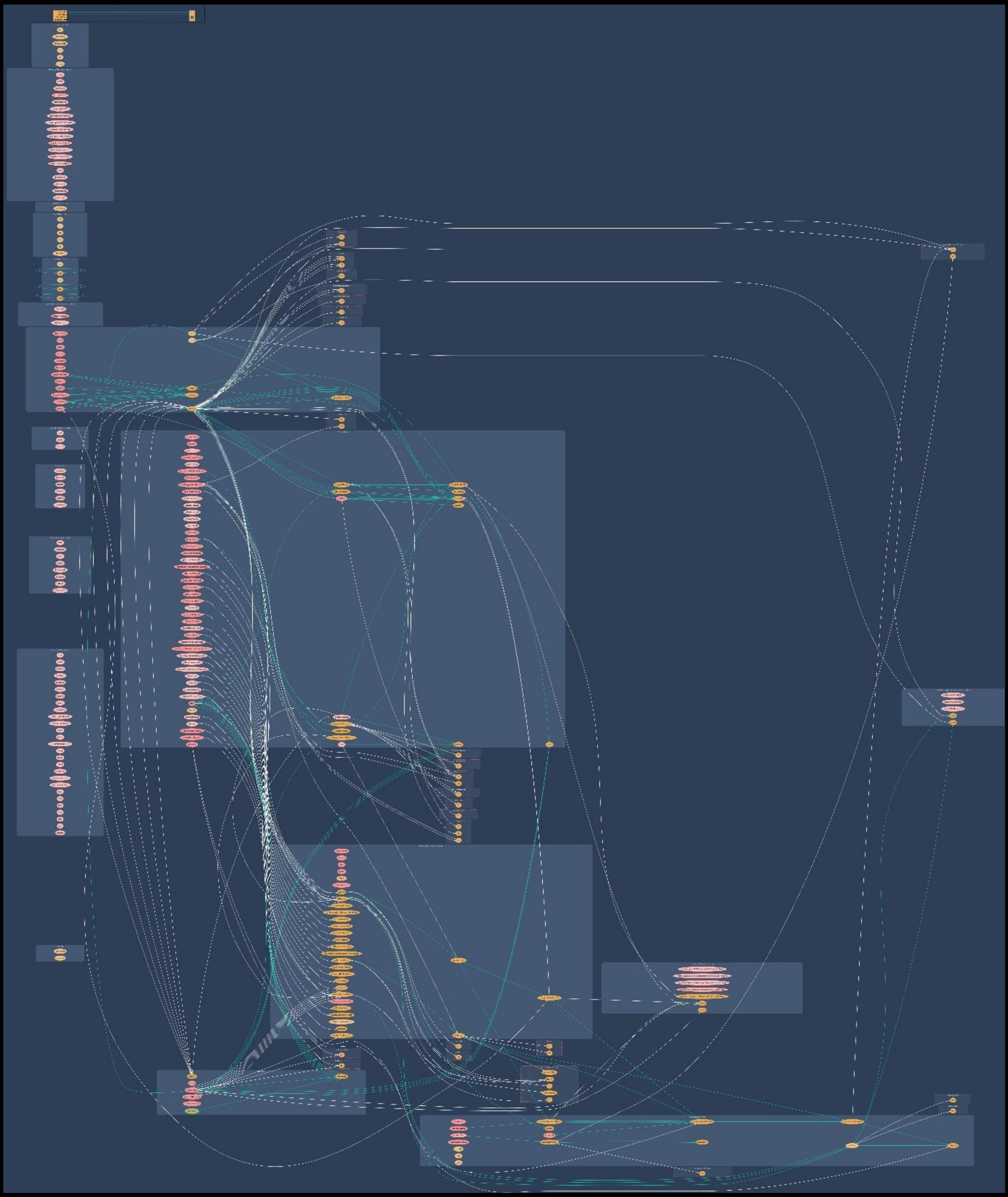
Item: 2	Location:	Line 270-285	Severity:	🟨 Medium

Function	The transferFrom() method overrides current allowance regardless of whether the spender already used it or not, so there is no way to increase or decrease allowance by a certain value atomically unless the token owner is a smart contract, not an account. This can be abused by a token receiver when they try to withdraw certain tokens from the sender's account. Meanwhile, if the sender decides to change the amount and sends another approve transaction, the receiver can notice this transaction before it's mined and can extract tokens from both the transactions, therefore, ending up with tokens from both the transactions. This is a front-running attack affecting the ERC20 Approve function. The function transferFrom can be front-run by abusing the _approve function.
Remedation	1. Introduce mechanisms that limit the maximum acceptable gas price for transactions. This can help prevent front-runners from drastically increasing the gas fees to prioritize their transactions. 2. Use transaction taxes to prevent against front-runattack

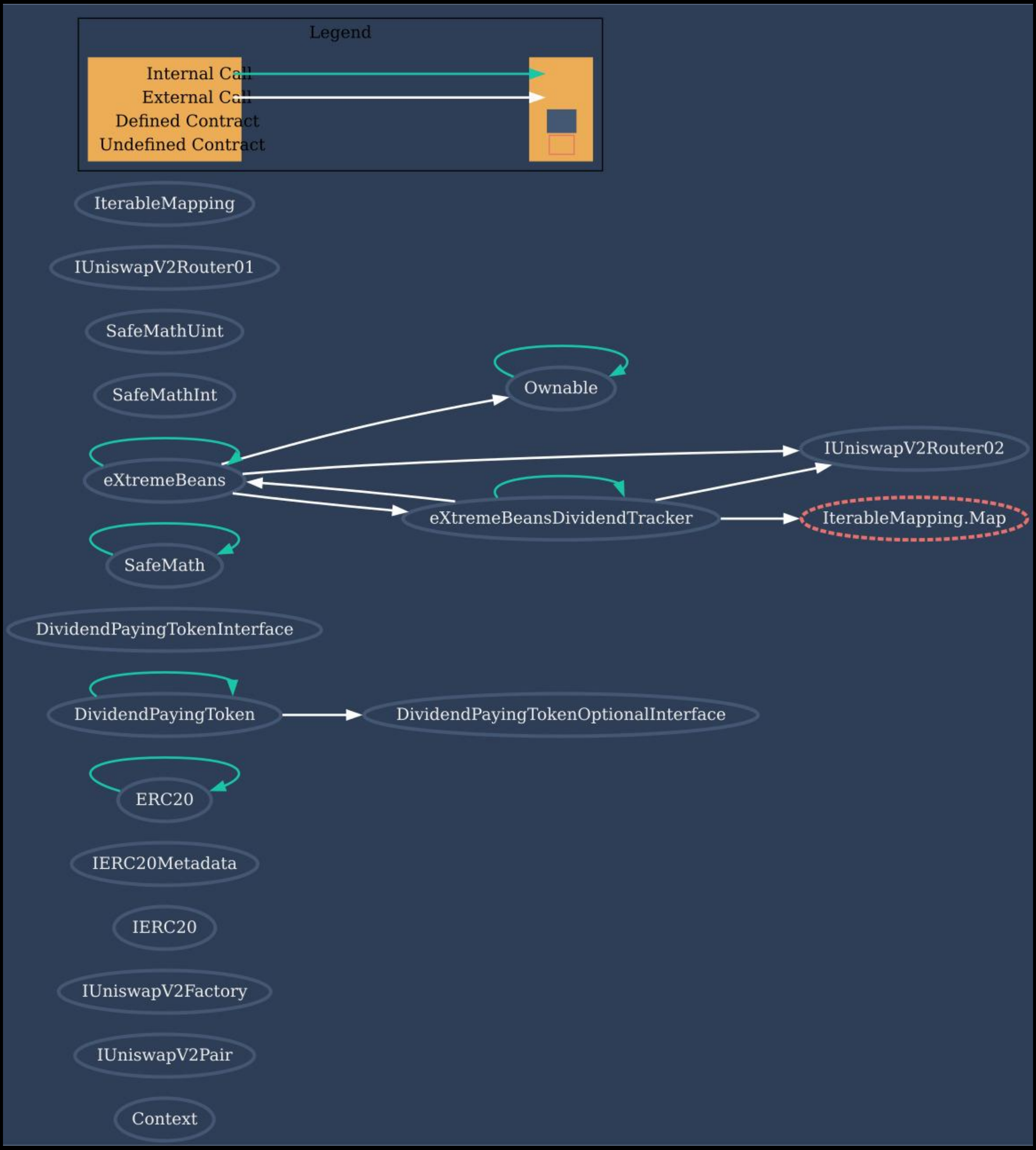


```
270     function transferFrom(
271         address sender,
272         address recipient,
273         uint256 amount
274     ) public virtual override returns (bool) {
275         _transfer(sender, recipient, amount);
276         _approve(
277             sender,
278             _msgSender(),
279             _allowances[sender][_msgSender()].sub(
280                 amount,
281                 "ERC20: transfer amount exceeds allowance"
282             )
283         );
284         return true;
285     }
286
```

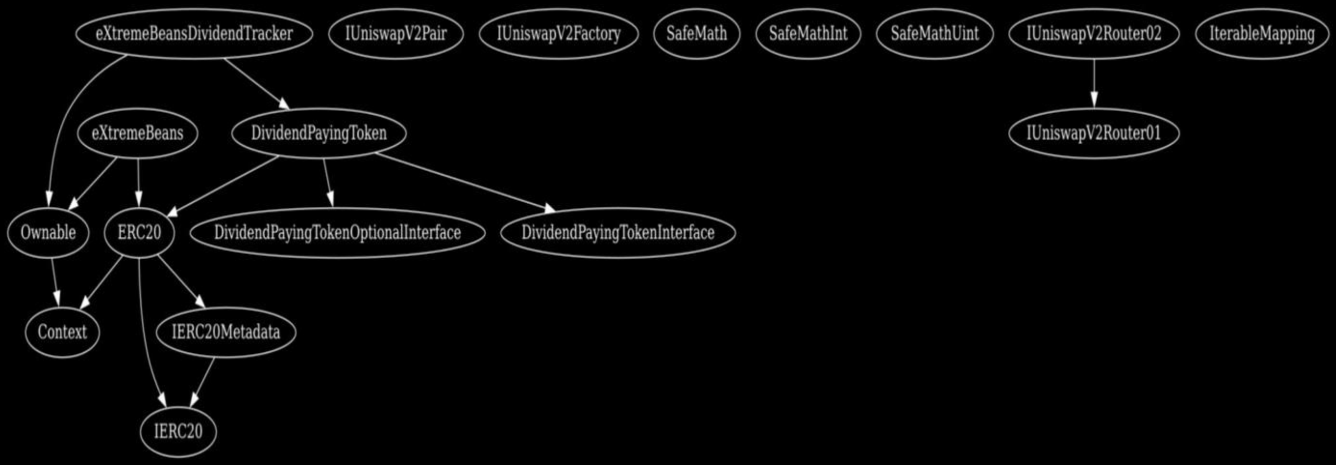
Contract Flow Graph



Contract Interaction Graph



Inheritance Graph



Contract Functions

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
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❗

L	MINIMUM_LIQUIDITY	External ❗		NO❗
L	factory	External ❗		NO❗
L	token0	External ❗		NO❗
L	token1	External ❗		NO❗
L	getReserves	External ❗		NO❗
L	price0CumulativeLast	External ❗		NO❗
L	price1CumulativeLast	External ❗		NO❗
L	kLast	External ❗		NO❗
L	mint	External ❗	🛑	NO❗
L	burn	External ❗	🛑	NO❗
L	swap	External ❗	🛑	NO❗
L	skim	External ❗	🛑	NO❗
L	sync	External ❗	🛑	NO❗
L	initialize	External ❗	🛑	NO❗
IUniswapV2Factory	Interface			
L	feeTo	External ❗		NO❗
L	feeToSetter	External ❗		NO❗
L	getPair	External ❗		NO❗
L	allPairs	External ❗		NO❗
L	allPairsLength	External ❗		NO❗
L	createPair	External ❗	🛑	NO❗

└	setFeeTo	External ❗	🛑	NO❗
└	setFeeToSetter	External ❗	🛑	NO❗
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		
└		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 🔒	🛑	
DividendPayingTokenOptionalInterface	Interface			
└	withdrawableDividendOf	External ❗️		NO❗️
└	withdrawnDividendOf	External ❗️		NO❗️
└	accumulativeDividendOf	External ❗️		NO❗️
DividendPayingTokenInterface	Interface			
└	dividendOf	External ❗️		NO❗️
└	distributeDividends	External ❗️	💵	NO❗️

L	withdrawDividend	External ❗	🛑	NO❗
SafeMath	Library			
L	add	Internal 🔒		
L	sub	Internal 🔒		
L	sub	Internal 🔒		
L	mul	Internal 🔒		
L	div	Internal 🔒		
L	div	Internal 🔒		
L	mod	Internal 🔒		
L	mod	Internal 🔒		
Ownable	Implementation	Context		
L		Public ❗	🛑	NO❗
L	owner	Public ❗		NO❗
L	renounceOwnership	Public ❗	🛑	onlyOwner
L	transferOwnership	Public ❗	🛑	onlyOwner
SafeMathInt	Library			
L	mul	Internal 🔒		
L	div	Internal 🔒		
L	sub	Internal 🔒		
L	add	Internal 🔒		
L	abs	Internal 🔒		

L	toUint256Safe	Internal 🔒		
SafeMathUint	Library			
L	toInt256Safe	Internal 🔒		
IUniswapV2Router01	Interface			
L	factory	External ❗		NO❗
L	WETH	External ❗		NO❗
L	addLiquidity	External ❗	🛑	NO❗
L	addLiquidityETH	External ❗	💵	NO❗
L	removeLiquidity	External ❗	🛑	NO❗
L	removeLiquidityETH	External ❗	🛑	NO❗
L	removeLiquidityWithPermit	External ❗	🛑	NO❗
L	removeLiquidityETHWithPermit	External ❗	🛑	NO❗
L	swapExactTokensForTokens	External ❗	🛑	NO❗
L	swapTokensForExactTokens	External ❗	🛑	NO❗
L	swapExactETHForTokens	External ❗	💵	NO❗
L	swapTokensForExactETH	External ❗	🛑	NO❗
L	swapExactTokensForETH	External ❗	🛑	NO❗
L	swapETHForExactTokens	External ❗	💵	NO❗

L	quote	External ❗		NO❗
L	getAmountOut	External ❗		NO❗
L	getAmountIn	External ❗		NO❗
L	getAmountsOut	External ❗		NO❗
L	getAmountsIn	External ❗		NO❗
IUniswapV2Router02	Interface	IUniswapV2Router01		
L	removeLiquidityETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
L	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
L	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
L	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵	NO❗
L	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
DividendPayingToken	Implementation	ERC20, DividendPayingTokenInterface, DividendPayingTokenOptionalInterface		

L		Public ❗️	🛑	ERC20
L		External ❗️	💵	NO❗️
L	distributeDividends	Public ❗️	💵	NO❗️
L	withdrawDividend	Public ❗️	🛑	NO❗️
L	_withdrawDividendOfUser	Internal 🔒	🛑	
L	dividendOf	Public ❗️		NO❗️
L	withdrawableDividendOf	Public ❗️		NO❗️
L	withdrawnDividendOf	Public ❗️		NO❗️
L	accumulativeDividendOf	Public ❗️		NO❗️
L	_transfer	Internal 🔒	🛑	
L	_mint	Internal 🔒	🛑	
L	_burn	Internal 🔒	🛑	
L	_setBalance	Internal 🔒	🛑	
eXtremeBeans	Implementation	ERC20, Ownable		
L		Public ❗️	🛑	ERC20
L	decimals	Public ❗️		NO❗️
L		External ❗️	💵	NO❗️
L	updateStakingAmounts	Public ❗️	🛑	onlyOwner
L	enableTrading	External ❗️	🛑	onlyOwner

L	setPresaleWallet	External ❗	🛑	onlyOwner
L	setExcludeFees	Public ❗	🛑	onlyOwner
L	setExcludeDividends	Public ❗	🛑	onlyOwner
L	setIncludeDividends	Public ❗	🛑	onlyOwner
L	setCanTransferBefore	External ❗	🛑	onlyOwner
L	setLimitsInEffect	External ❗	🛑	onlyOwner
L	setGasPriceLimit	External ❗	🛑	onlyOwner
L	setcooldowntimer	External ❗	🛑	onlyOwner
L	setmaxWallet	External ❗	🛑	onlyOwner
L	enableStaking	Public ❗	🛑	onlyOwner
L	stake	Public ❗	🛑	NO❗
L	setSwapTriggerAmount	Public ❗	🛑	onlyOwner
L	enableSwapAndLiquify	Public ❗	🛑	onlyOwner
L	setAutomatedMarketMakerPair	Public ❗	🛑	onlyOwner
L	setAllowCustomTokens	Public ❗	🛑	onlyOwner
L	setAllowAutoReinvest	Public ❗	🛑	onlyOwner
L	_setAutomatedMarketMakerPair	Private 🔐	🛑	

L	updateGasForProcessing	Public ▯	🛑	onlyOwner
L	transferAdmin	Public ▯	🛑	onlyOwner
L	updateTransferFee	Public ▯	🛑	onlyOwner
L	updateFees	Public ▯	🛑	onlyOwner
L	getStakingInfo	External ▯		NO▯
L	getTotalDividendsDistributed	External ▯		NO▯
L	isExcludedFromFees	Public ▯		NO▯
L	withdrawableDividendOf	Public ▯		NO▯
L	dividendTokenBalanceOf	Public ▯		NO▯
L	getAccountDividendsInfo	External ▯		NO▯
L	getAccountDividendsInfoAtIndex	External ▯		NO▯
L	processDividendTracker	External ▯	🛑	NO▯
L	claim	External ▯	🛑	NO▯
L	getLastProcessedIndex	External ▯		NO▯
L	getNumberOfDividendTokenHolders	External ▯		NO▯
L	setAutoClaim	External ▯	🛑	NO▯
L	setReinvest	External ▯	🛑	NO▯

L	setDividendsPaused	External ❗	🛑	onlyOwner
L	isExcludedFromAutoClaim	External ❗		NO❗
L	isReinvest	External ❗		NO❗
L	_transfer	Internal 🔒	🛑	
L	getStakingBalance	Private 🔐		
L	swapAndLiquify	Private 🔐	🛑	
L	swapTokensForEth	Private 🔐	🛑	
L	updatePayoutToken	Public ❗	🛑	onlyOwner
L	getPayoutToken	Public ❗		NO❗
L	setMinimumTokenBalanceForAutoDividends	Public ❗	🛑	onlyOwner
L	setMinimumTokenBalanceForDividends	Public ❗	🛑	onlyOwner
L	addLiquidity	Private 🔐	🛑	
L	forceSwapAndSendDividends	Public ❗	🛑	onlyOwner
L	swapAndSendDividends	Private 🔐	🛑	
L	multiSend	Public ❗	🛑	onlyOwner
L	airdropToWallets	External ❗	🛑	onlyOwner

eXtremeBeans DividendTracker	Implementation	DividendPaying Token, Ownable		
L		Public ▯	🛑	DividendPaying Token
L	decimals	Public ▯		NO▯
L	name	Public ▯		NO▯
L	symbol	Public ▯		NO▯
L	_transfer	Internal 🔒		
L	withdrawDividend	Public ▯		NO▯
L	isExcludedFromAutoClaim	External ▯		onlyOwner
L	isReinvest	External ▯		onlyOwner
L	setAllowCustomTokens	External ▯	🛑	onlyOwner
L	setAllowAutoReinvest	External ▯	🛑	onlyOwner
L	excludeFromDividends	External ▯	🛑	onlyOwner
L	includeFromDividends	External ▯	🛑	onlyOwner
L	setAutoClaim	External ▯	🛑	onlyOwner
L	setReinvest	External ▯	🛑	onlyOwner
L	setMinimumTokenBalanceForAutoDividends	External ▯	🛑	onlyOwner
L	setMinimumTokenBalanceForDividends	External ▯	🛑	onlyOwner

L	setDividendsPaused	External ❗	🛑	onlyOwner
L	getLastProcessedIndex	External ❗		NO❗
L	getNumberOfTokenHolders	External ❗		NO❗
L	getAccount	Public ❗		NO❗
L	getAccountAtIndex	Public ❗		NO❗
L	setBalance	External ❗	🛑	onlyOwner
L	process	Public ❗	🛑	NO❗
L	processAccount	Public ❗	🛑	onlyOwner
L	updateUniswapV2Router	Public ❗	🛑	onlyOwner
L	updatePayoutToken	Public ❗	🛑	onlyOwner
L	getPayoutToken	Public ❗		NO❗
L	_reinvestDividendOfUser	Private 🔐	🛑	
L	_withdrawDividendOfUser	Internal 🔒	🛑	
IterableMapping	Library			
L	get	Internal 🔒		
L	getIndexOfKey	Internal 🔒		
L	getKeyAtIndex	Internal 🔒		
L	size	Internal 🔒		
L	set	Internal 🔒	🛑	

L	remove	Internal 🔒	🛑	

🛑	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