SKELETON ECOSYSTEM

SMART CONTRACT AUDIT

Vive
$ViV
BEP20

0x953b8c871cd31527dbb67dc071c507e34aea3838

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	ViVe
Ticker/Simbol	ViV
Blockchain	Binance Smart Chain BEP20
Contract Address	0x953b8c871cd31527dbb67dc071c507e34aea3838
Creator Address	0x068181A722cEB830AB12149560912e879d9F9ff6
Current Owner Address	0x068181A722cEB830AB12149560912e879d9F9ff6
Contract Explorer	https://bscscan.com/token/0x953b8c871cd31527dbb67dc071c507e34aea3838#code
Compiler Version	v0.8.15+commit.e14f2714
License	MIT
Optimisation	Yes with 200 Runs
Total Supply	174,284,506.356682 **ViV**
Decimals	18

Creation/Audit

Contract Deployed	28.12.2023
Audit Created	06.01.2024
Audit Update	V 1.0

Verified Socials

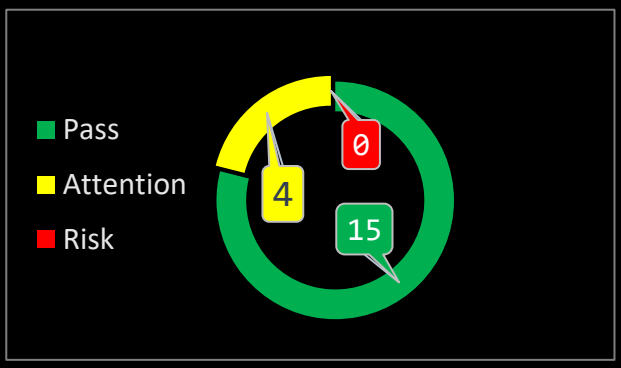
Website	https://vive.services/
Telegram	https://t.me/viveprotocol
Twitter (X)	https://twitter.com/viveprotocol

SKELETON ECOSYSTEM
SMART CONTRACT AUDIT REPORT

VIVE 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		0x068181A722cEB830AB12149560912e879d9F9ff6 Deployer
Buy Tax	10 %	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	10 %	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	✓	Liqudity Locker status 05.01.2024: Locked: 95% on Pinklock for 24 Days
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	⚠ max 26%	Fee Setting function found. (max 26% limit 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 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 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	 300s max	Trading Cooldown Function found. (max 300 sec) 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

The following functions are listed as functions may be triggered by the owner of the contract.

⚠ Whitelist (Set excluded)

[Remedation: Renounce ownership]

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)

```
1150    // exclude a wallet from fees
        ftrace | funcSig
1151    function setExcludeFees(address account, bool excluded) public onlyOwner {
1152        _isExcludedFromFees[account] = excluded;
1153        emit ExcludeFromFees(account, excluded);
1154    }
```

```
1192    // set max wallet, can not be lower than 0.05% of supply
        ftrace | funcSig
1193    function setmaxWallet(uint256 value) external onlyOwner {
1194        value = value * (10**18);
1195        require(value >= _totalSupply / 100000, "max wallet cannot be set to less than 0.05%");
1196        maxWallet = value;
1197    }
```

⚠ Max Transaction and Holding Modify function

[Remedation: Renounce ownership]

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

```
1192    // set max wallet, can not be lower than 0.05% of supply
        ftrace | funcSig
1193    function setmaxWallet(uint256 value) external onlyOwner {
1194        value = value * (10**18);
1195        require(value >= _totalSupply / 100000, "max wallet cannot be set to less than 0.05%");
1196        maxWallet = value;
1197    }
```

⚠ Set Trading Fees (Max 26% limit found)

[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 (honeypot risk).

```
ftrace | funcSig
function updateFees(
    uint256 deadBuy,
    uint256 deadSell,
    uint256 marketingBuy,
    uint256 marketingSell,
    uint256 liquidityBuy,
    uint256 liquiditySell,
    uint256 RewardsBuy,
    uint256 RewardsSell,
    uint256 devBuy,
    uint256 devSell
) public onlyOwner {
    buyDeadFees = deadBuy;
    buyMarketingFees = marketingBuy;
    buyLiquidityFee = liquidityBuy;
    buyRewardsFee = RewardsBuy;
    sellDeadFees = deadSell;
    sellMarketingFees = marketingSell;
    sellLiquidityFee = liquiditySell;
    sellRewardsFee = RewardsSell;
    buyDevFee = devBuy;
    sellDevFee = devSell;

    totalSellFees = sellRewardsFee
        .add(sellLiquidityFee)
        .add(sellMarketingFees)
        .add(sellDevFee);

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

    require(totalSellFees <= 26 && totalBuyFees <= 26, "total fees cannot exceed 26% sell or buy");
```

⚠ Trading Cooldown Function (300 sec. limit found)

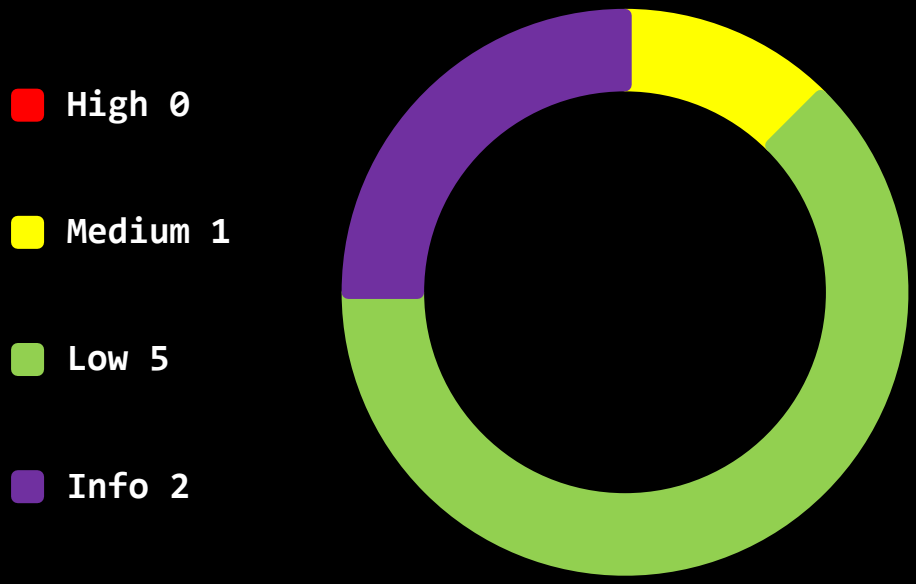
[Remedation: Renounce ownership]

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.

```
// set cooldown timer, can only be between 0 and 300 seconds (5 mins max)
ftrace | funcSig
function setcooldowntimer(uint256 value) external onlyOwner {
    require(value <= 300, "cooldown timer cannot exceed 5 minutes");
    cooldowntimer = value;
}
```

Contract Security

Total Findings: 8



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: (1)

- Usage of Tx. Origin

Low severity issues: (5)

- Missing Events
- Front Running attack approval
- Floating Pragma
- Long Number Literals
- Aprrove of Front Running Attack

Informational severity issues: (2)

- Public Functions Should be Declared External
- State Variables Should be Declared Constant

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	Low	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	Passed	Passed	Passed
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.

⚠ Usage of TX. Origin (6 Items)

Item: 1	Location:	Line 1399	Severity:	Medium
Item: 1	Location:	Line 1493	Severity:	Medium
Item: 1	Location:	Line 1495	Severity:	Medium
Item: 1	Location:	Line 1509	Severity:	Medium
Item: 1	Location:	Line 1510	Severity:	Medium
Item: 1	Location:	Line 1590	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.

```
1398            gas,
1399            tx.origin
```

```
1493        require(block.timestamp >= _holderLastTransferTimestamp[tx.origin] + cooldowntimer,
1494                "cooldown period active");
```

```
1495        _holderLastTransferTimestamp[tx.origin] = block.timestamp;
```

```
1509        require(_holderLastTransferBlock[tx.origin] != block.number,"Too many TX in block");
1510        _holderLastTransferBlock[tx.origin] = block.number;
```

```
1589            gas,
1590            tx.origin
```

⚠ Approve of front running attack (2 Items)

Item: 1	Location:	Line 264-272	Severity:	Low

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-run attack

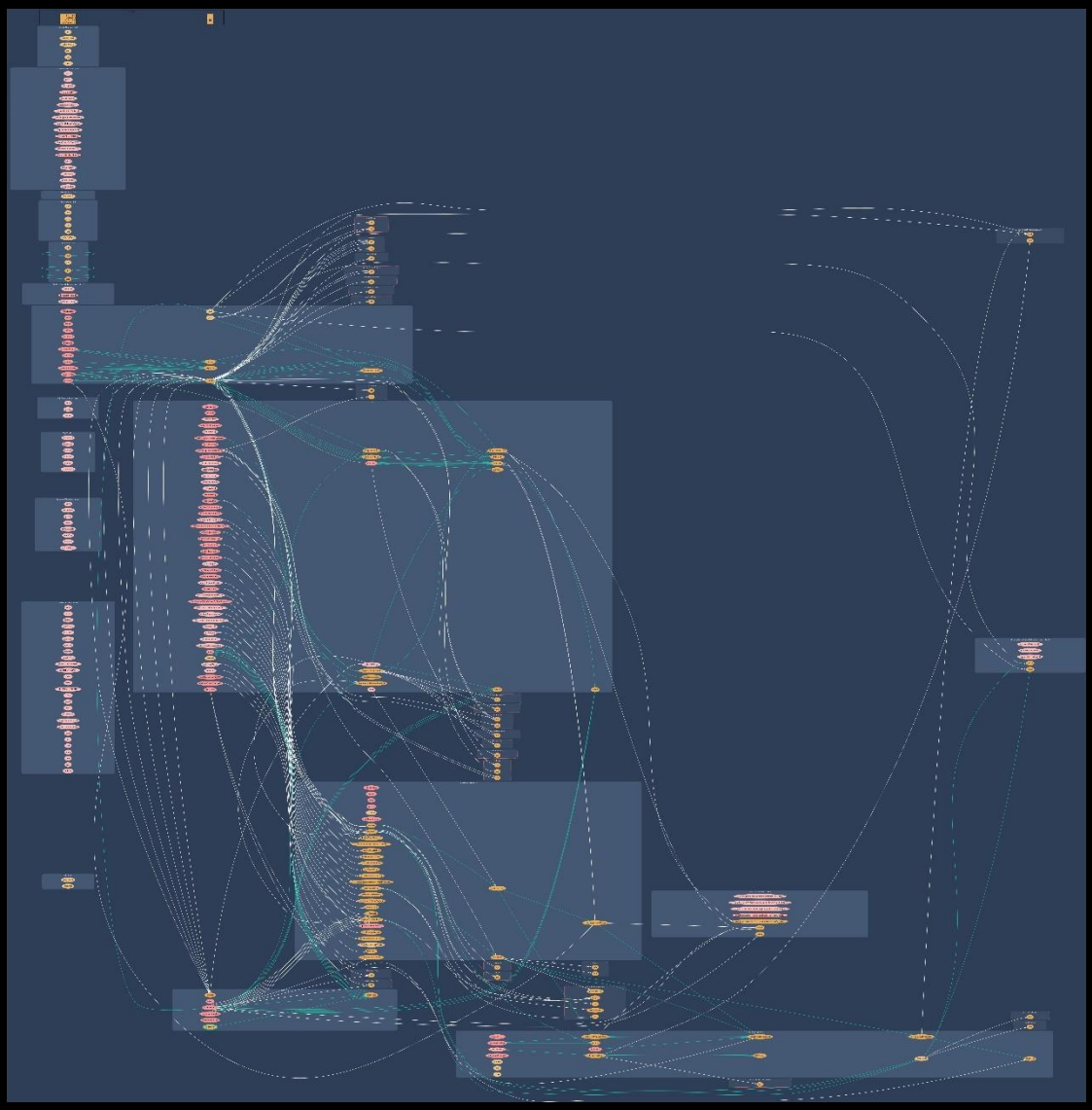
```
ftrace | funcSig
264  function approve(address spender, uint256 amount)
265      public
266      virtual
267      override
268      returns (bool)
269  {
270      _approve(_msgSender(), spender, amount);
271      return true;
272  }
273
```

Item: 2	Location:	Line 274-289	Severity:	Low

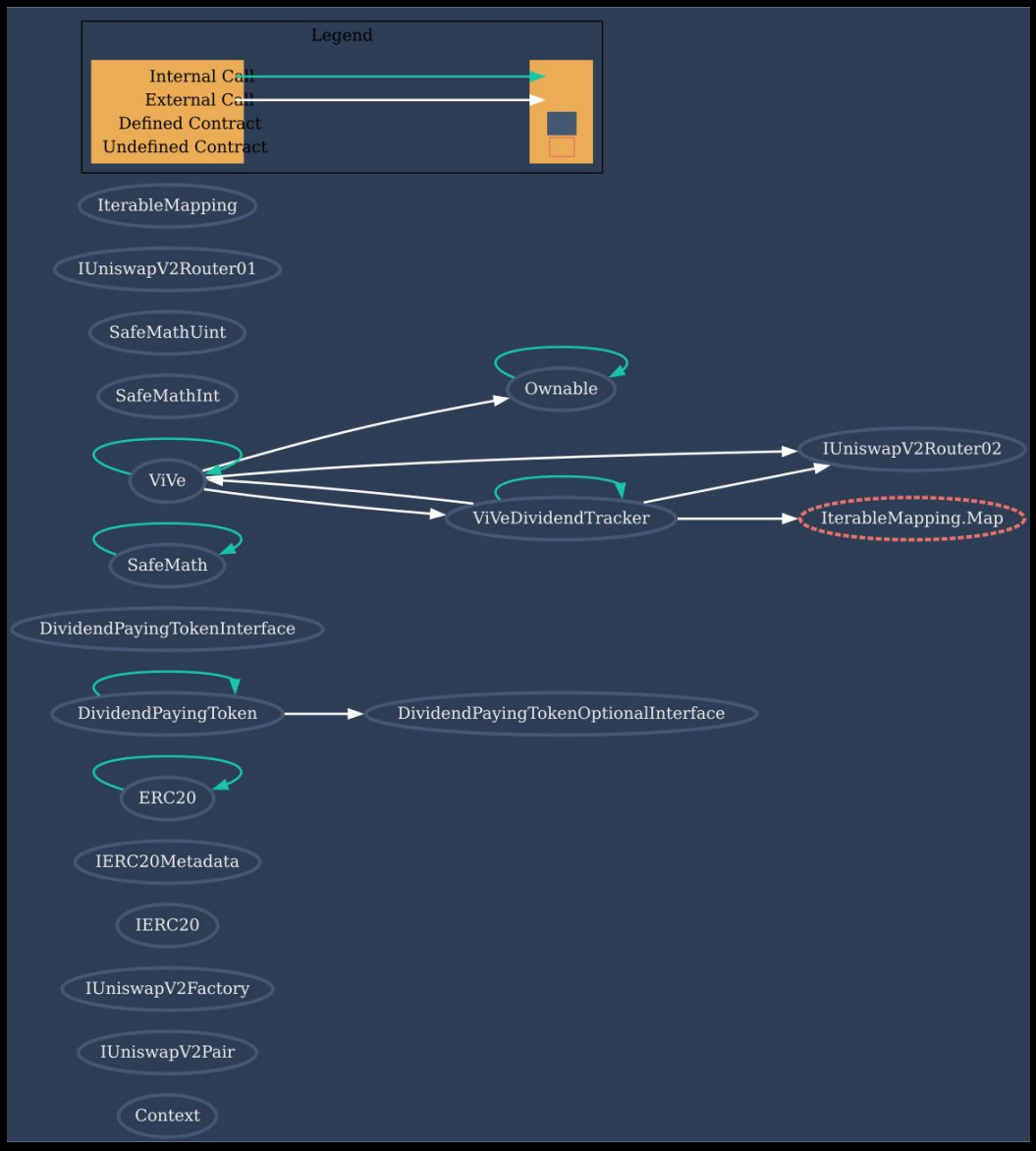
Function	The transferFrom() method overrides current allowance regardless of whether the spender already used it or not, so there is no way to increase or decrease allowance by a certain value atomically unless the token owner is a smart contract, not an account. This can be abused by a token receiver 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	3. 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. 4. Use transaction taxes to prevent against front-run attack

```
ftrace | funcSig
274     function transferFrom(
275         address sender,
276         address recipient,
277         uint256 amount
278     ) public virtual override returns (bool) {
279         _transfer(sender, recipient, amount);
280         _approve(
281             sender,
282             _msgSender(),
283             _allowances[sender][_msgSender()].sub(
284                 amount,
285                 "ERC20: transfer amount exceeds allowance"
286             )
287         );
288         return true;
289     }
290
```

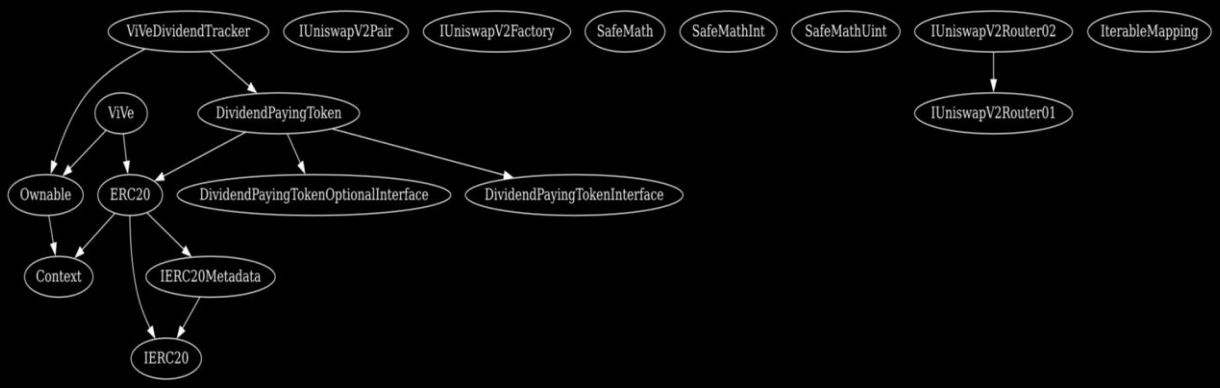
Contract Flow Graph



Contract Interaction Graph



Inheritance Graph



Contract Functions

Contract	Type	Bases		
L	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
L	_msgSender	Internal 🔒		
L	_msgData	Internal 🔒		
IUniswapV2Pair	Interface			
L	name	External ❗		NO❗
L	symbol	External ❗		NO❗
L	decimals	External ❗		NO❗
L	totalSupply	External ❗		NO❗
L	balanceOf	External ❗		NO❗
L	allowance	External ❗		NO❗
L	approve	External ❗	🛑	NO❗
L	transfer	External ❗	🛑	NO❗
L	transferFrom	External ❗	🛑	NO❗
L	DOMAIN_SEPARATOR	External ❗		NO❗
L	PERMIT_TYPEHASH	External ❗		NO❗
L	nonces	External ❗		NO❗
L	permit	External ❗	🛑	NO❗

Contract	Type	Bases		
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❗

Contract	Type	Bases		
L	createPair	External ❗	🔘	NO❗
L	setFeeTo	External ❗	🔘	NO❗
L	setFeeToSetter	External ❗	🔘	NO❗
IERC20	Interface			
L	totalSupply	External ❗		NO❗
L	balanceOf	External ❗		NO❗
L	transfer	External ❗	🔘	NO❗
L	allowance	External ❗		NO❗
L	approve	External ❗	🔘	NO❗
L	transferFrom	External ❗	🔘	NO❗
IERC20Metadata	Interface	IERC20		
L	name	External ❗		NO❗
L	symbol	External ❗		NO❗
L	decimals	External ❗		NO❗
ERC20	Implementation	Context, IERC20, IERC20Metadata		
L		Public ❗	🔘	NO❗
L	name	Public ❗		NO❗
L	symbol	Public ❗		NO❗
L	decimals	Public ❗		NO❗
L	totalSupply	Public ❗		NO❗

Contract	Type	Bases		
L	balanceOf	Public ❗		NO❗
L	transfer	Public ❗	🛑	NO❗
L	allowance	Public ❗		NO❗
L	approve	Public ❗	🛑	NO❗
L	transferFrom	Public ❗	🛑	NO❗
L	increaseAllowance	Public ❗	🛑	NO❗
L	decreaseAllowance	Public ❗	🛑	NO❗
L	_transfer	Internal 🔒	🛑	
L	_mint	Internal 🔒	🛑	
L	_burn	Internal 🔒	🛑	
L	_approve	Internal 🔒	🛑	
L	_beforeTokenTransfer	Internal 🔒	🛑	
DividendPayingTokenOptionalInterface	Interface			
L	withdrawableDividendOf	External ❗		NO❗
L	withdrawnDividendOf	External ❗		NO❗
L	accumulativeDividendOf	External ❗		NO❗
DividendPayingTokenInterface	Interface			

Contract	Type	Bases		
L	dividendOf	External ❗		NO❗
L	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 🔒		

Contract	Type	Bases		
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❗

Contract	Type	Bases		
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	swapExactETHForTokensSuppo	External ❗	💸	NO❗

Contract	Type	Bases		
	rtingFeeOnTransferTokens			
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 🔒	🛑	

Contract	Type	Bases		
L	_setBalance	Internal 🔒	🛑	
ViVe	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

Contract	Type	Bases		
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❗

Contract	Type	Bases		
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❗

Contract	Type	Bases		
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
ViVeDividendTracker	Implementation	DividendPayingToken, Ownable		
L		Public ❗	🛑	DividendPayingToken
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

Contract	Type	Bases		
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

Contract	Type	Bases		
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