

SKELETON ECOSYSTEM

SMART CONTRACT AUDIT



PlayBets Finance
PLAYBETS
BEP20

0x469EBfE5C256c6A9EA9811284b85bd332ECC3



Table of Contents

Table of Contents	1
Disclaimer	2
Overview	3
Creation/Audit Date	3
Verified Socials	3
Contract Functions Analysis	4
Contract Safety and Weakness	9
Detected Vulnerability Description	13
Contract Flow Graph	16
Contract Interaction Graph	17
Inheritance Graph	18
Contract Descriptions	19
Audit Scope	26

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% safety 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	PlayBest
Ticker/Symbol	PLAYBETS
Blockchain	Binance Smart Chain BEP20
Contract Address	0x469EBfE5C256c6A9EA9811284b85bd332ECC34eb
Creator Address	0x7ED2a37Ad3808C1281FE285D0D898B454cC949B1
Current Owner Address	0x7ED2a37Ad3808C1281FE285D0D898B454cC949B1
Contract Explorer	https://bscscan.com/address/0x469EBfE5C256c6A9EA9811284b85bd332ECC34eb#code
Compiler Version	v0.8.7+commit.e28d00a7
License	Unlicense
Optimisation	No with 200 Runs
Total Supply	1,000,000,000 PLAYBETS
Decimals	9




Creation/Audit

Contract Deployed	10.06.2024
Audit Created	11.06.2024
Audit Update	V 1.0

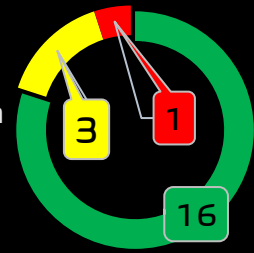
Verified Socials








Website	http://playbets.finance/
Telegram	https://t.me/playbetsfinance
Twitter (X)	https://x.com/PlaybetsFinance

Contract Function Analysis

 Pass
  Attention Item
  Risky Item

■ Pass
 ■ Attention
 ■ Risk



Contract Verified		The contract source code is uploaded to blockchain explorer and is open source, so everybody can read it.
Contract Ownership		0x7ED2a37Ad3808C1281FE285D0D898B454cC949B1 Deployer
Buy Tax	8 %	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	8 %	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
Liquidity Status		Liquidity status on 11.06.2024 Lp Tokens Locked 98.68% on Unicrypt locker for 367 days. BNB Smart Chain Transaction Hash (Txhash) Details BscScan
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 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		<p>No Balance Modifier function found.</p> <p>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.</p>
Blacklist Function		<p>No Blacklist Setting function found.</p> <p>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.</p>
Whitelist Function		<p>Whitelist Setting function found</p> <p>If there is a function for this Developer can set zero fee or no max wallet size for addresses (for example team wallets can trade without fee. Can cause farming)</p>
Hidden Owner Analysis		<p>No Hidden or multi owner with authorisation</p> <p>For contract with a hidden owner, developer can still manipulate the contract even if the ownership has been abandoned.</p>
Retrieve Ownership Function		<p>No Functions found which can retrieve ownership of the contract.</p> <p>If this function exists, it is possible for the project owner to regain ownership even after relinquishing it. Also known as fake renounce.</p>
Self Destruct Function		<p>No Self Destruct function found.</p> <p>If this function exists and is triggered, the contract will be destroyed, all functions will be unavailable, and all related assets will be erased.</p>
Specific Tax Changing Function		<p>No Specific Tax Changing Functions found.</p> <p>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!</p>
Trading Cooldown Function		<p>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.</p>
Max Transaction and Holding Modify Function		<p>Max Transaction and Holding Modify function found.</p> <p>If there is a function for this, the maximum trading amount or maximum position can be modified. Can cause honeypot</p>
Transaction Limiting Function		<p>No Transaction Limiter Function Found.</p> <p>The number of overall token transactions may be limited (honeypot risk)</p>

Details of Risk - Attention Items



Set Fee

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).

```
806         ftrace | funcSig
807         function setBuyTaxes(
808             uint256 newLiquidityTax!,
809             uint256 newMarketingTax!,
810             uint256 newTeamTax!
811         ) external onlyOwner {
812             _buyLiquidityFee = newLiquidityTax!;
813             _buyMarketingFee = newMarketingTax!;
814             _buyTeamFee = newTeamTax!;
815             _totalTaxIfBuying = _buyLiquidityFee.add(_buyMarketingFee).add(
816                 _buyTeamFee
817             );
818         }
819
820         ftrace | funcSig
821         function setSellTaxes(
822             uint256 newLiquidityTax!,
823             uint256 newMarketingTax!,
824             uint256 newTeamTax!
825         ) external onlyOwner {
826             _sellLiquidityFee = newLiquidityTax!;
827             _sellMarketingFee = newMarketingTax!;
828             _sellTeamFee = newTeamTax!;
829             _totalTaxIfSelling = _sellLiquidityFee.add(_sellMarketingFee).add(
830                 _sellTeamFee
831             );
832         }
833     }
```

⚠ Whitelist

If there is a function for this Developer can set zero fee or no max wallet size for addresses (for example team wallets can trade without fee. Can cause farming)

```

856 | ftrace | funcSig
857 | function setIsWalletLimitExempt(address holder!, bool exempt!)
858 |     external
859 |     onlyOwner
860 | {
861 |     _isWalletLimitExempt[holder!] = exempt!;
862 | }
  
```

```

792 | ftrace | funcSig
793 | function setIsTxLimitExempt(address holder!, bool exempt!)
794 |     external
795 |     onlyOwner
796 | {
797 |     _isTxLimitExempt[holder!] = exempt!;
798 | }
799 | ftrace | funcSig
800 | function setIsExcludedFromFee(address account!, bool newValue!)
801 |     public
802 |     onlyOwner
803 | {
804 |     _isExcludedFromFee[account!] = newValue!;
805 | }
  
```

⚠ 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

```

847 | ftrace | funcSig
848 | function setMaxTxAmount(uint256 maxTxAmount!) external onlyOwner {
849 |     _maxTxAmount = maxTxAmount!;
850 | }
851 |
  
```

```

863 | ftrace | funcSig
864 | function setWalletLimit(uint256 newLimit!) external onlyOwner {
865 |     _walletMax = newLimit!;
866 | }
  
```


⚠ Blocking trading by changing the router Version

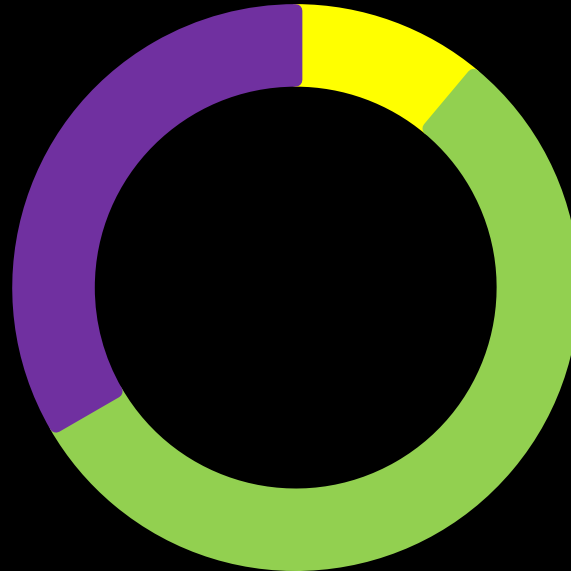
The router version can be manually changed, which can cause trading problems.

```
fttrace | funcSig
898 function changeRouterVersion(address newRouterAddress!)
899     public
900     onlyOwner
901     returns (address newPairAddress)
902 {
903     IUniswapV2Router02 _uniswapV2Router = IUniswapV2Router02(
904         newRouterAddress!
905     );
906
907     newPairAddress = IUniswapV2Factory(_uniswapV2Router.factory()).getPair(
908         address(this),
909         _uniswapV2Router.WETH()
910     );
911
912     if (newPairAddress == address(0)) //Create If Doesnt exist
913     {
914         newPairAddress = IUniswapV2Factory(_uniswapV2Router.factory())
915             .createPair(address(this), _uniswapV2Router.WETH());
916     }
917
918     uniswapPair = newPairAddress; //Set new pair address
919     uniswapV2Router = _uniswapV2Router; //Set new router address
920
921     isWalletLimitExempt[address(uniswapPair)] = true;
922     isMarketPair[address(uniswapPair)] = true;
923 }
924
925 //to recieve ETH from uniswapV2Router when swaping
fttrace
receive() external payable {}
```

Contract Security

Total Findings: 9

- High 0
- Medium 1
- Low 5
- Info 3



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)

- Incorrect Access Control

Low severity issues: (5)

- Missing Events
- Long number literals
- Outdated compiler Version
- Unchecked Array Length
- Floating Pragma

Informational severity issues: (3)

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

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	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	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	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.

Incorrect Access Control (2 Item)

Item: 1	Location:	Line 764-771	Severity:	 Medium
---------	-----------	--------------	-----------	--

Function	<p>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.</p> <p>The contract PlayBets is importing an access control library @openzeppelin/contracts/access/Ownable.sol but the function approve is missing the modifier onlyOwner.</p>
Remediation	<ol style="list-style-type: none"> 1. Consider adding access control modifiers to the function to Ensure that initialization functions can only be called once and only by authorized entities. 2. Implement least-privilege roles using libraries like OpenZeppelin's Access Control. 3. Add proper access control modifiers to sensitive functions, such as onlyOwner or custom roles.

```

764 | ftrace | funcSig
765 | function approve(address spender, uint256 amount)
766 |     public
767 |     override
768 |     returns (bool)
769 | {
770 |     _approve(_msgSender(), spender, amount);
771 |     return true;
772 | }
  
```

Item: 2	Location:	Line 928-935	Severity: ■ Medium
---------	-----------	--------------	---

Function	<p>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.</p> <p>The contract PlayBets is importing an access control library @openzeppelin/contracts/access/Ownable.sol but the function transfer is missing the modifier onlyOwner.</p>
Remedation	<ol style="list-style-type: none"> 1. Consider adding access control modifiers to the function to Ensure that initialization functions can only be called once and only by authorized entities. 2. Implement least-privilege roles using libraries like OpenZeppelin's Access Control. 3. Add proper access control modifiers to sensitive functions, such as onlyOwner or custom roles.

```

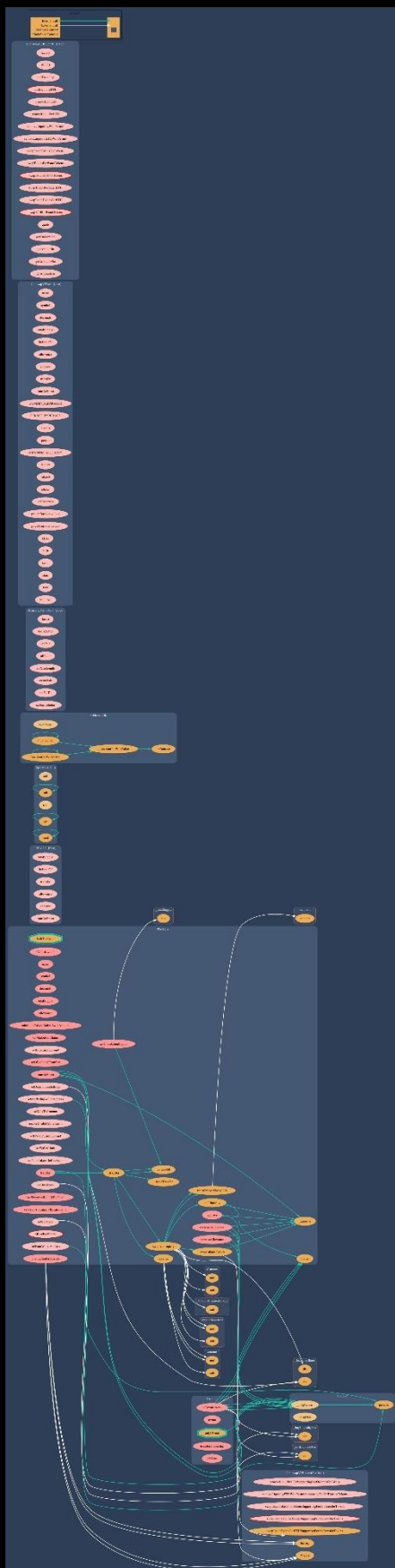
ftrace | funcSig
928 | function transfer(address recipient!, uint256 amount!)
929 |     public
930 |     override
931 |     returns (bool)
932 | {
933 |     _transfer(_msgSender(), recipient!, amount!);
934 |     return true;
935 | }
936 |
  
```

Outdated Compiler Version.

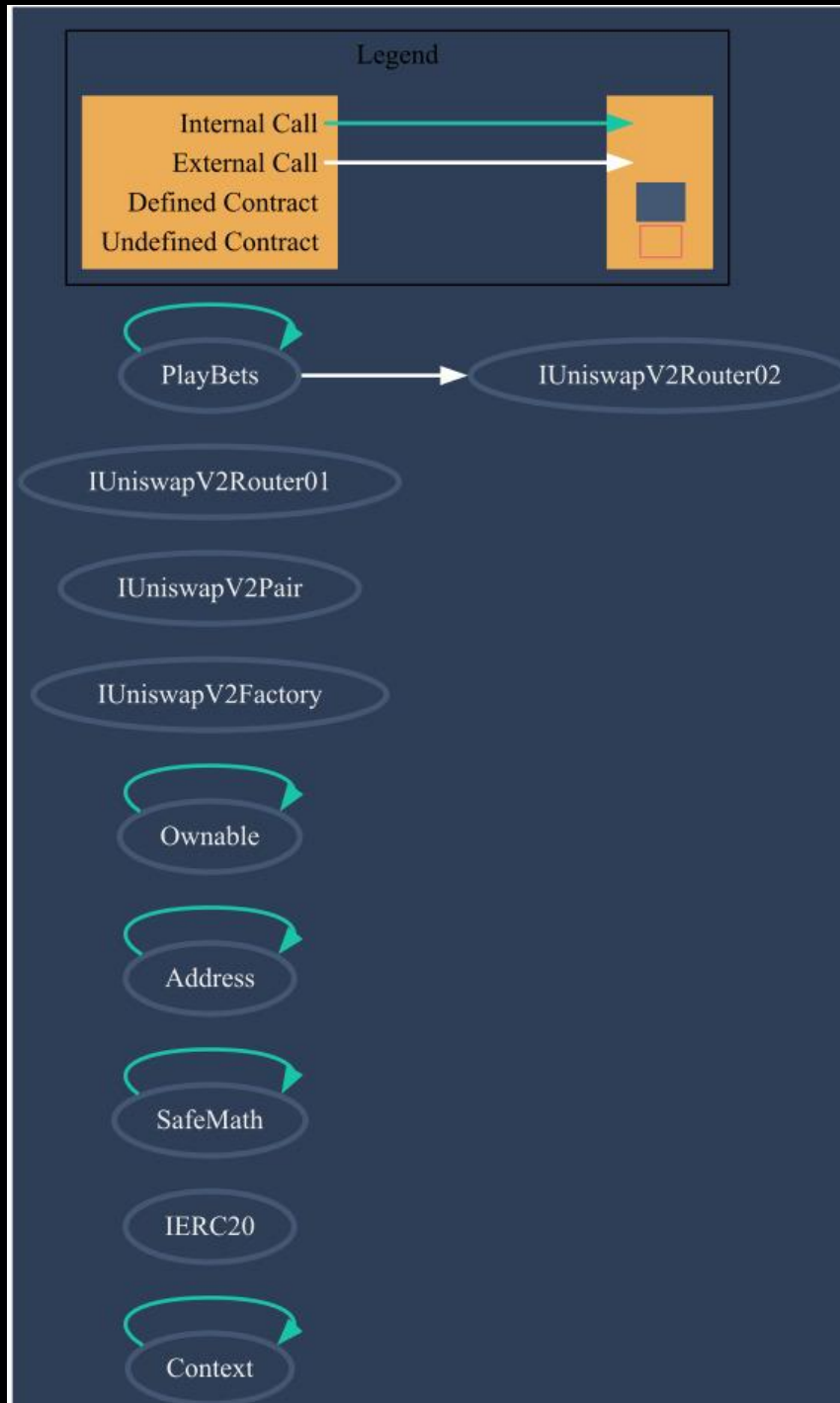
Item: 1	Location:	Line 14	Severity:	 Low
---------	-----------	---------	-----------	---

Function	Using an outdated compiler version can be problematic especially if there are publicly disclosed bugs and issues that affect the current compiler version. The following outdated versions were detected: /playbets.sol - ^0.8.7
Remedation	It is recommended to use a recent version of the Solidity compiler that should not be the most recent version, and it should not be an outdated version as well. Using very old versions of Solidity prevents the benefits of bug fixes and newer security checks. Consider using the solidity version v0.8.25, which patches most solidity vulnerabilities.

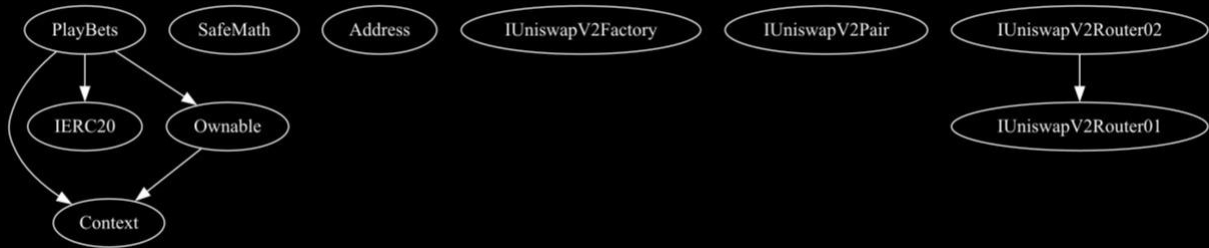
Contract Flow Graph





























Contract Interaction Graph















































Inheritance Graph























Contract Functions

Contract	Type	Bases		
L	Function Name	Visibility	Mutability	Modifiers
Context	Implementation			
L	_msgSender	Internal 		
L	_msgData	Internal 		
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 
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 		
Address	Library			
L	isContract	Internal 		













L	sendValue	Internal 		
L	functionCall	Internal 		
L	functionCall	Internal 		
L	functionCallWithValue	Internal 		
L	functionCallWithValue	Internal 		
L	_functionCallWithValue	Private 		
Ownable	Implementation	Context		
L		Public 		NO 
L	owner	Public 		NO 
L	transferOwnership	Public 		onlyOwner
L	getTime	Public 		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 
L	setFeeTo	External 		NO 
L	setFeeToSetter	External 		NO 
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_SEPARAT OR	External ¶		NO ¶
L	PERMIT_TYPEHAS H	External ¶		NO ¶
L	nonces	External ¶		NO ¶
L	permit	External ¶	●	NO ¶
L	MINIMUM_LIQUIDI TY	External ¶		NO ¶
L	factory	External ¶		NO ¶
L	token0	External ¶		NO ¶
L	token1	External ¶		NO ¶
L	getReserves	External ¶		NO ¶
L	price0Cumulative Last	External ¶		NO ¶
L	price1Cumulative Last	External ¶		NO ¶
L	kLast	External ¶		NO ¶
L	burn	External ¶	●	NO ¶
L	swap	External ¶	●	NO ¶
L	skim	External ¶	●	NO ¶
L	sync	External ¶	●	NO ¶
L	initialize	External ¶	●	NO ¶

IUniswapV2Router 01	Interface			
L	factory	External ¶		NO ¶
L	WETH	External ¶		NO ¶
L	addLiquidity	External ¶		NO ¶
L	addLiquidityETH	External ¶		NO ¶
L	removeLiquidity	External ¶		NO ¶
L	removeLiquidityE TH	External ¶		NO ¶
L	removeLiquidityW ithPermit	External ¶		NO ¶
L	removeLiquidityE THWithPermit	External ¶		NO ¶
L	swapExactTokens ForTokens	External ¶		NO ¶
L	swapTokensForEx actTokens	External ¶		NO ¶
L	swapExactETHFor Tokens	External ¶		NO ¶
L	swapTokensForEx actETH	External ¶		NO ¶
L	swapExactTokens ForETH	External ¶		NO ¶
L	swapETHForExact Tokens	External ¶		NO ¶
L	quote	External ¶		NO ¶
L	getAmountOut	External ¶		NO ¶
L	getAmountIn	External ¶		NO ¶
L	getAmountsOut	External ¶		NO ¶
L	getAmountsIn	External ¶		NO ¶
IUniswapV2Router 02	Interface	IUniswapV2Router 01		

L	removeLiquidityETHSupportingFeeOnTransferTokens	External 		NO 
L	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External 		NO 
L	swapExactTokensForTokensSupportingFeeOnTransferTokens	External 		NO 
L	swapExactETHForTokensSupportingFeeOnTransferTokens	External 		NO 
L	swapExactTokensForETHSupportingFeeOnTransferTokens	External 		NO 
PlayBets	Implementation	Context, IERC20, Ownable		
L		Public 		NO 
L	name	Public 		NO 
L	symbol	Public 		NO 
L	decimals	Public 		NO 
L	totalSupply	Public 		NO 
L	balanceOf	Public 		NO 
L	allowance	Public 		NO 
L	increaseAllowance	Public 		NO 
L	decreaseAllowance	Public 		NO 
L	minimumTokensBeforeSwapAmount	Public 		NO 
L	approve	Public 		NO 
L	_approve	Private 		

L	setMarketPairStatus	Public 		onlyOwner
L	setIsTxLimitExempt	External 		onlyOwner
L	setIsExcludedFromFee	Public 		onlyOwner
L	setBuyTaxes	External 		onlyOwner
L	setSellTaxes	External 		onlyOwner
L	setDistributionSettings	External 		onlyOwner
L	setMaxTxAmount	External 		onlyOwner
L	enableDisableWalletLimit	External 		onlyOwner
L	setIsWalletLimitExempt	External 		onlyOwner
L	setWalletLimit	External 		onlyOwner
L	setNumTokensBeforeSwap	External 		onlyOwner
L	setMarketingWalletAddress	External 		onlyOwner
L	setTeamWalletAddress	External 		onlyOwner
L	setSwapAndLiquifyEnabled	Public 		onlyOwner
L	setSwapAndLiquifyByLimitOnly	Public 		onlyOwner
L	getCirculatingSupply	Public 		NO 
L	transferToAddressETH	Private 		
L	changeRouterVersion	Public 		onlyOwner
L		External 		NO 
L	transfer	Public 		NO 
L	transferFrom	Public 		NO 

L	_transfer	Private 		
L	_basicTransfer	Internal 		
L	swapAndLiquify	Private 		lockTheSwap
L	swapTokensForEth	Private 		
L	addLiquidity	Private 		
L	takeFee	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 vulnerabilities in the code. Findings getting reported and improvements getting suggested.

Automatic and Manual Review

We are using automated tools to scan functions and weaknesses 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>

