

SKELETON ECOSYSTEM

SMART CONTRACT AUDIT



BlueFi Network (BlueFi) BEP20

0xba1dcf564Ac9bd4495E9DE8825626234eeBcC



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	8
Detected Vulnerability Description	12
Contract Flow Graph	15
Contract Interaction Graph	16
Inheritance Graph	17
Contract Descriptions	18
Audit Scope	25

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	BlueFi
Ticker/Symbol	BlueFi
Blockchain	Binance Smart Chain BEP20
Contract Address	0xba1dcf564Ac9bd4495E9DE8825626234eeBcCce5
Creator Address	0xC501a8f631edc3C5F0E7dcF21Da37B950c0e8C9D
Current Owner Address	0xC501a8f631edc3C5F0E7dcF21Da37B950c0e8C9D
Contract Explorer	https://bscscan.com/token/0xba1dcf564Ac9bd4495E9DE8825626234eeBcCce5#code
Compiler Version	v0.8.18+commit.87f61d96
License	Unlicense
Optimisation	No with 200 Runs
Total Supply	100,000,000 BlueFi
Decimals	9




Creation/Audit

Contract Deployed	18.04.2024
Audit Created	19.04.2024
Audit Update	V 1.0

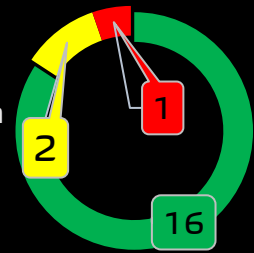
Verified Socials


Website	https://bluefi.org/
Telegram	https://t.me/bluefinetwork
Twitter (X)	https://twitter.com/bluefinetwork

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		0xC501a8f631edc3C5F0E7dcF21Da37B950c0e8C9D 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		Liquiduty Lock Status on 18.04.2024: 95.37% UNCX for 366 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		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 Fees function

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
545 function setBuyTaxes(uint256 newLiquidityTax!, uint256 newMarketingTax!, uint256 newDevelopmentTax!) external onlyOwner() {
546     buyLiquidityFee = newLiquidityTax!;
547     buyMarketingFee = newMarketingTax!;
548     buyDevelopmentFee = newDevelopmentTax!;
549
550     totalTaxIfBuying = buyLiquidityFee.add(buyMarketingFee).add(buyDevelopmentFee);
551 }
552
ftrace | funcSig
553 function setSellTaxes(uint256 newLiquidityTax!, uint256 newMarketingTax!, uint256 newDevelopmentTax!) external onlyOwner() {
554     sellLiquidityFee = newLiquidityTax!;
555     sellMarketingFee = newMarketingTax!;
556     sellDevelopmentFee = newDevelopmentTax!;
557
558     totalTaxIfSelling = sellLiquidityFee.add(sellMarketingFee).add(sellDevelopmentFee);
559 }
  
```

Whitelist function [Exclude from Fees and wallet limit]

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)

```

ftrace | funcSig
537 function setIsTxLimitExempt(address holder!, bool exempt!) external onlyOwner {
538     isTxLimitExempt[holder!] = exempt!;
539 }
540
ftrace | funcSig
541 function setIsExcludedFromFee(address account!, bool newValue!) public onlyOwner {
542     isExcludedFromFee[account!] = newValue!;
543 }
544
  
```

⚠️ Max Transaction and Holding Modify Function (Max 2% and no minimum)

If there is a function for this, the maximum trading amount or maximum position can be modified. Max transaction can be set to Zero which can cause honeypot.

```
568 |  
    | ftrace | funcSig  
569 | function setMaxTxAmount(uint256 maxTxAmount) external onlyOwner() {  
570 |     require(maxTxAmount <= (5 * 10**5 * 10**9), "Max wallet should be less or equal to 2% totalSupply");  
571 |     _maxTxAmount = maxTxAmount;  
572 | }  
573 |
```

```
582 | ftrace | funcSig  
582 | function setWalletLimit(uint256 newLimit) external onlyOwner {  
583 |     _walletMax = newLimit;  
584 | }  
585 |
```


Contract Security

Total Findings: 7

■ High 0

■ Medium 1

■ 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: (1)

- Incorrect Acces Control

Low severity issues: (4)

- Long number literals
- Low Level Calls
- Public Functions Should be Declared External
- Outdated Compiler Version

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

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	Low	Low
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	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 Acces Control [2 Item]

Item: 1	Location:	Line 520-523	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 BlueFi is importing an access control library @openzeppelin/contracts/access/Ownable.sol but the function approve is missing the modifier onlyOwner.</p>
Remedation	<ol style="list-style-type: none"> 1. 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.

```

520     ftrace | funcSig
521     function approve(address spender, uint256 amount) public override returns (bool) {
522         _approve(_msgSender(), spender, amount);
523         return true;
524     }
  
```

Item: 2	Location:	Line 637-640	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 BlueFi is importing an access control library @openzeppelin/contracts/access/Ownable.sol but the function transfer is missing the modifier onlyOwner.</p>
Remediation	<ol style="list-style-type: none"> 1. 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.

```

637  function transfer(address recipient!, uint256 amount!) public override returns (bool) {
638      _transfer(_msgSender(), recipient!, amount!);
639      return true;
640  }
641

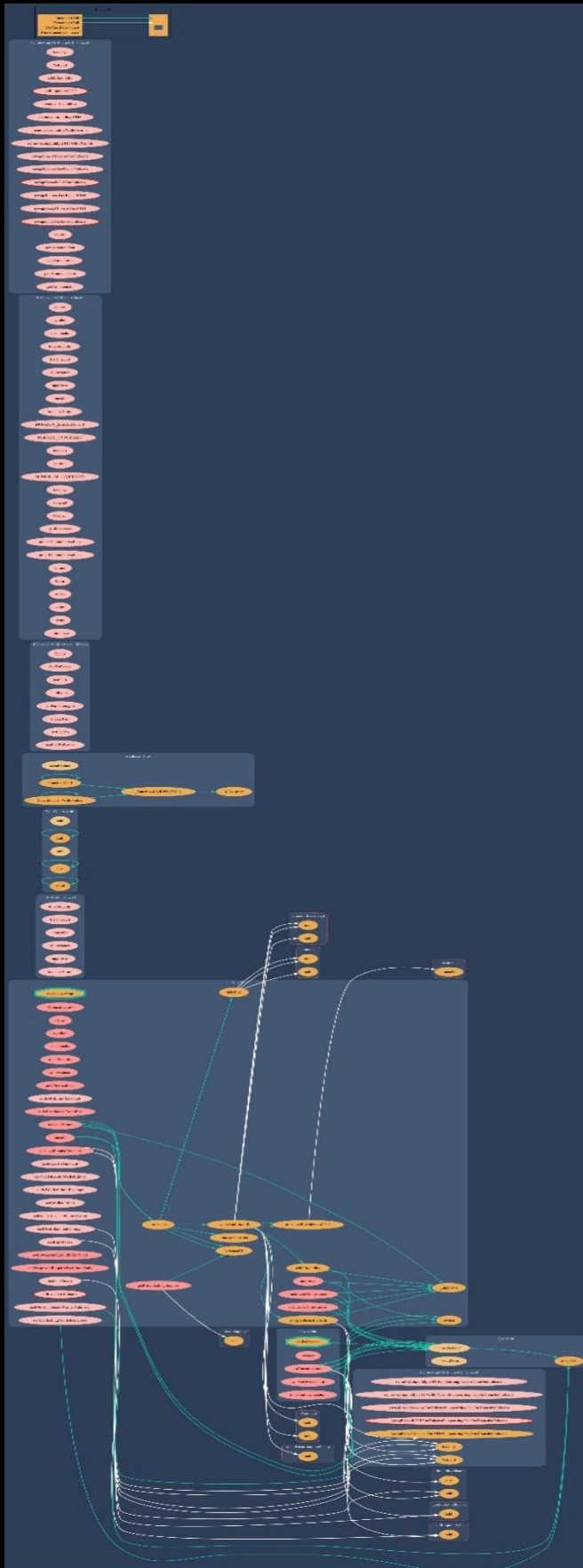
```

⚠ Outdated Compiler Version [1 Item]

Item: 1	Location:	Line 13	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: /bluefi.sol - ^0.8.18
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 v0.8.24, which patches most solidity vulnerabilities..

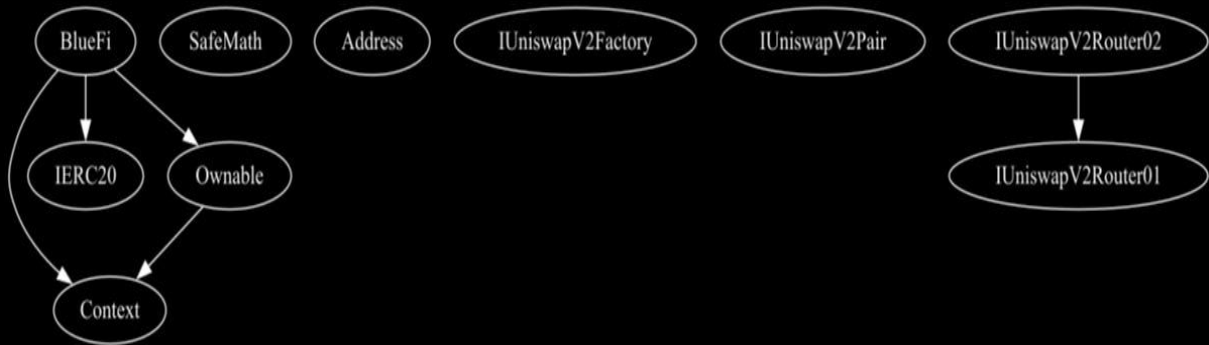
Contract Flow Graph




























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	waiveOwnership	Public 🚫	🔒	onlyOwner
L	transferOwnership	Public 🚫	🔒	onlyOwner
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_SEPARATOR	External ¶		NO ¶
L	PERMIT_TYPEHASH	External ¶		NO ¶
L	nonces	External ¶		NO ¶
L	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	burn	External 		NO 
L	swap	External 		NO 
L	skim	External 		NO 
L	sync	External 		NO 
L	initialize	External 		NO 
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 ¶
BlueFi	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	approve	Public 		NO 
L	_approve	Private 		
L	addMarketPair	Public 		onlyOwner
L	setIsTxLimitExempt	External 		onlyOwner
L	setIsExcludedFromFee	Public 		onlyOwner
L	setBuyTaxes	External 		onlyOwner
L	setSellTaxes	External 		onlyOwner
L	setDistributionsSettings	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	setDevelopmentWalletAddress	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>

