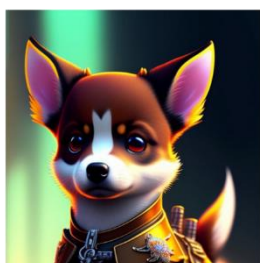


SKELETON ECOSYSTEM

SMART CONTRACT AUDIT



SHIBA4EVER
BEP 20

0x8fa900dbc3c3835801254abe50959181cd44c328



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	6
Detected Vulnerability Description	9
Contract Flow Chart	11
Inheritance Graph	12
Contract Descriptions	13
Audit Scope	17

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

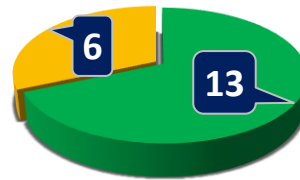
Contract Name	SHIBA4EVER
Ticker/Symbol	S4E
Blockchain	Binance Smart Chain Bep20
Contract Address	0x8fa900dbc3c3835801254abe50959181cd44c328
Creator Address	0xBbfabc2491396e403edaAA7f3CB669CFC4ad2530
Current Owner Address	Renounced
Contract Explorer	https://bscscan.com/token/0x8fa900dbc3c3835801254abe50959181cd44c328
Compiler Version	v0.8.7+commit.e28d00a7
License	MIT License
Optimisation	No with 200 Runs
Total Supply	100,000,000 S4E
Decimals	18

Creation/Audit

Contract Deployed	10 Aug 2023
Audit Created	21-Aug-23 19:00:00 UTC
Audit Update	V 0.1

Verified Socials

Website	https://shiba4ever.io
Telegram	https://t.me/Shiba4EvEr
X	https://Twitter.com/Shiba4_Ever



Contract Function Analysis



Pass



Attention Item































Risky Item

Pass

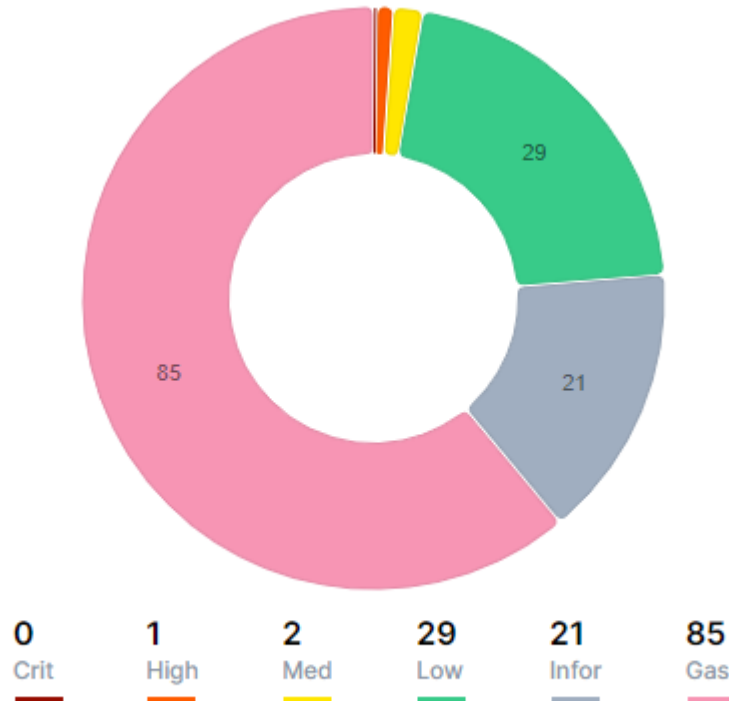
Attention

Risk

Contract Verified		The contract source code is uploaded to blockchain explorer and is open source, so everybody can read it.
Contract Ownership		The ownership of the contract was sent to dead address. With this the owner eliminates he's rights to modify the contract. The owner can not set any of the functions anymore.
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. Contract renounced so tax rate is fixed. 
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. Contract renounced so tax rate is fixed. 
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		Locked on 21.08.2023: 98% for 359 days on Mudra locker Note! Initial liquidity tokens scanned. For new LP Lockers allways re-check with skeleton scanner on telegram.
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.  If there is authorised hidden owner, or there is Retrieve Ownership Function, the trading disable function may 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). If contract is renounced this function can't be used.  If there is authorised hidden owner, or there is Retrieve Ownership Function, the set fees function may be used!  Renounced, this function can not be used.
Proxy Contract		The proxy contract means contract owner can modify the function of the token and possibly effect the price. The Owner is not the creator but the creator may have authorisation to change functions.
Mint Function		No mint function found. 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. If contract is renounced this function can't be used.

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> <p> If contract is renounced this function still can be used as auto self Destruct</p>
Whitelist Function		<p>Whitelist 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> <p>If there is a whitelist, some addresses may not be able to trade normally (honeypot risk).  Renounced, this function can not be used.</p>
Hidden Owner Analysis		<p>No authorised hidden owner found.</p> <p>For contract with a hidden owner, developer can still manipulate the contract even if the ownership has been abandoned. Fake renounce.</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>Specific Tax Changing Functions found.</p> <p> Renounced, this function can not be used.</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>Trading Cooldown Function found.</p> <p> Renounced, this function can not be used.</p> <p>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> Renounced, this function can not be used.</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>Transaction Limiter Function Found.</p> <p> Renounced, this function can not be used.</p> <p>The number of overall token transactions may be limited (honeypot risk)</p>

Contract Safety and Weakness



● UNCHECKED ARRAY LENGTH	1
● RETURN VALUE OF LOW LEVEL CALLS	2
● EMPTY TRY/CATCH BLOCK	4
● USE OF FLOATING PRAGMA	1
● MISSING EVENTS	18
● OUTDATED COMPILER VERSION	1
● LONG NUMBER LITERALS	3
● FUNCTION RETURNS TYPE AND NO RE...	2
● MISSING PAYABLE IN CALL FUNCTION	1
● RETURN INSIDE LOOP	1
● MISSING STATE VARIABLE VISIBILITY	1
● MISSING INHERITANCE	2
● MISSING UNDERSCORE IN NAMING VA...	4
● BLOCK VALUES AS A PROXY FOR TIME	1
● REQUIRE WITH EMPTY MESSAGE	1
● HARD-CODED ADDRESS DETECTED	9
● UNUSED RECEIVE FALLBACK	1
● UNUSED NAMED RETURNS	2
● CONSTANT STATE VARIABLES	3


● UNNECESSARY CHECKED ARITHMETI...	2
● GAS OPTIMIZATION IN INCREMENTS	2
● FUNCTION SHOULD BE EXTERNAL	10
● FUNCTION SHOULD RETURN STRUCT	1
● CHEAPER INEQUALITIES IN REQUIRE()	6
● CHEAPER INEQUALITIES IN IF()	9
● GAS OPTIMIZATION FOR STATE VARIA...	1
● VARIABLES DECLARED BUT NEVER US...	1
● UNNECESSARY DEFAULT VALUE INITI...	1
● ARRAY LENGTH CACHING	2

Unchecked Array Lenght (1 item)

```

735     function includeInReward(address account) external onlyOwner {
736         require(!_excluded[account], "account is not excluded");
737         for (uint256 i = 0; i < _excluded.length; i++) {
738             if (_excluded[i] == account) {
739                 _excluded[i] = _excluded[_excluded.length - 1];
740                 _tOwned[account] = 0;
741                 _isExcluded[account] = false;
742                 _excluded.pop();
743                 break;

```

Function	Severity	Remediation
<p>UNCHECKED ARRAY LENGTH</p> <p>Ethereum is a very resource-constrained environment. Prices per computational step are orders of magnitude higher than with centralized providers. Moreover, Ethereum miners impose a limit on the total number of Gas consumed in a block. If array.length is large enough, the function exceeds the block gas limit, and transactions calling it will never be confirmed.</p> <pre>for (uint256 i = 0; i < array.length ; i++) { costlyFunc(); }</pre> <p>This becomes a security issue if an external actor influences array.length.</p> <p>E.g., if an array enumerates all registered addresses, an adversary can register many addresses, causing the problem described above.</p>	<p> Severity : High</p>	<p>Either explicitly or just due to normal operation, the number of iterations in a loop can grow beyond the block gas limit, which can cause the complete contract to be stalled at a certain point. Therefore, loops with a bigger or unknown number of steps should always be avoided.</p>


⚠ Return Value of low level calls (2 Item)

```

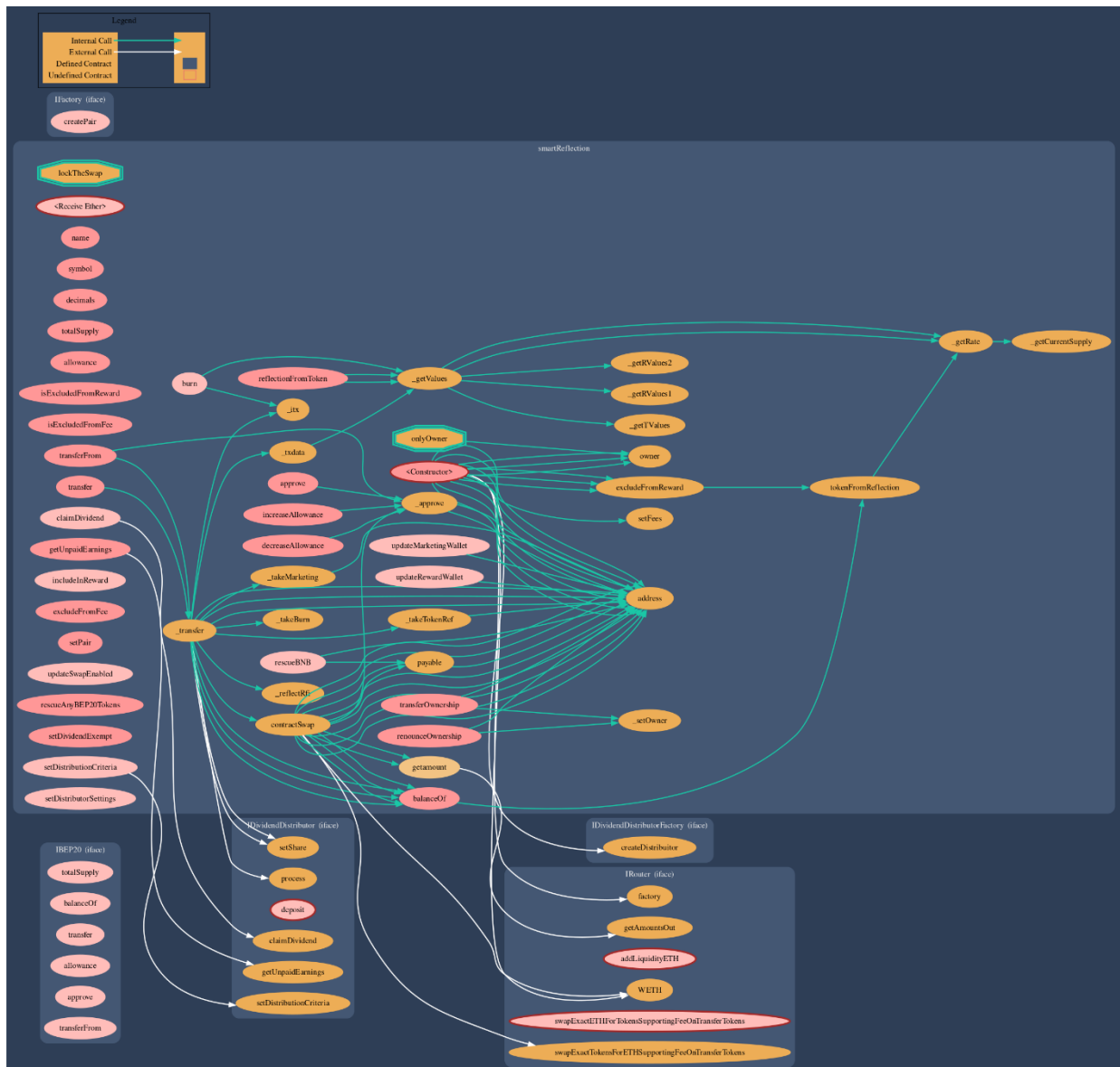
704         bool success;
705         if (amountBNBMarketing > 0) {
706             (success, ) = payable(marketingWallet).call{
707                 value: address(this).balance
708             }("");
709             // payable(marketingFeeReceiver).transfer(amountBNBMarketing);
710         }

789     function rescueBNB(uint256 weiAmount) external onlyOwner {
790         require(address(this).balance >= weiAmount, "insufficient BNB balance");
791         bool success;
792         (success, ) = payable(msg.sender).call{value: weiAmount}("");
793     }

```

Function	Severity	Remediation
<p>The functions do not check the return value of low-level calls. This can lock Ether in the contract if the call fails or may compromise the contract if the ownership is being changed.</p> <p>The following calls were detected without return value validations - call</p>	 Severity : medium	Ensure return value is checked using conditional statements for low-level calls. We should also ensure that we log failed calls using events.






Contract Flow Graph

















Inheritance Graph















Contract Descriptions

Contract	Type	Bases		
		Visibility	Mutability	Modifiers
IBEP20	Interface			
	totalSupply	External !		NO !
	balanceOf	External !		NO !
	transfer	External !		NO !
	allowance	External !		NO !
	approve	External !		NO !
	transferFrom	External !		NO !
IRouter	Interface			
	factory	External !		NO !
	WETH	External !		NO !
	addLiquidityETH	External !		NO !
	swapExactTokensForETHSupportingFeeOnTransferTokens	External !		NO !
	swapExactETHForTokensSupportingFeeOnTransferTokens	External !		NO !
	getAmountsOut	External !		NO !

IFactory	Interface			
	createPair	External !		NO !
IDividendDistributor	Interface			
	setDistributionCriteria	External !		NO !
	setShare	External !		NO !
	deposit	External !		NO !
	process	External !		NO !
	claimDividend	External !		NO !
	getUnpaidEarnings	External !		NO !
IDividendDistributorFactory	Interface			
	createDistributor	External !		NO !
smartReflection	Implementation			
		Public !		NO !
		External !		NO !
	owner	Public !		NO !
	name	Public !		NO !
	symbol	Public !		NO !
	decimals	Public !		NO !
	totalSupply	Public !		NO !
	balanceOf	Public !		NO !
	allowance	Public !		NO !
	approve	Public !		NO !
	increaseAllowance	Public !		NO !
	decreaseAllowance	Public !		NO !
	_approve	Private 		

	isExcludedFromReward	Public !		NO!
	isExcludedFromFee	Public !		NO!
	reflectionFromToken	Public !		NO!
	tokenFromReflection	Public !		NO!
	_reflectRfi	Private 🔒	🛑	
	_takeBurn	Private 🔒	🛑	
	_takeMarketing	Private 🔒	🛑	
	_takeTokenRef	Private 🔒	🛑	
	_getValues	Private 🔒		
	_getTValues	Private 🔒		
	_getRValues1	Private 🔒		
	_getRValues2	Private 🔒		
	_getRate	Public !		NO!
	_getCurrentSupply	Public !		NO!
	transfer	Public !	🛑	NO!
	transferFrom	Public !	🛑	NO!
	_transfer	Private 🔒	🛑	
	burn	External !	🛑	NO!
	_itx	Internal 🔒	🛑	
	_txdata	Private 🔒		
	getamount	Internal 🔒		
	contractSwap	Private 🔒	🛑	lockTheSwap
	claimDividend	External !	🛑	NO!
	getUnpaidEarnings	Public !		NO!
	excludeFromReward	Public !	🛑	onlyOwner
	includeInReward	External !	🛑	onlyOwner
	setFees	Public !	🛑	onlyOwner
	excludeFromFee	Public !	🛑	onlyOwner
	setPair	Public !	🛑	onlyOwner

	updateMarketingWallet	External !		onlyOwner
	updateRewardWallet	External !		onlyOwner
	updateSwapEnabled	External !		onlyOwner
	rescueBNB	External !		onlyOwner
	rescueAnyBEP20Tokens	Public !		onlyOwner
	setDividendExempt	Public !		onlyOwner
	setDistributionCriteria	External !		onlyOwner
	setDistributorSettings	External !		onlyOwner
	_setOwner	Private 		
	renounceOwnership	Public !		onlyOwner
	transferOwnership	Public !		onlyOwner



Function
can modify
state



Function
is payable

Source:

File Name SHA-1 Hash

c:\Solidity\shiba.sol 92ea91fccaba25ce1ffb3f79f819ffcf27e39bcd

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>

