Audit Report

ShibaSafe

February 2022

Type	Github
SHA256	8d7fa92e9f9418915729bb509b280c66d11b49f02639f6bcb609afb9eb6c3d32
Audited by	© coinscope

Table of Contents

Table of Contents 1
Contract Review 3
Audit Updates 3
Contract Analysis 4
ST - Stop Transactions 5
Description 5
Recommendation 6
ULTW - Unlimited Liquidity to Team Wallet 7
Description 7
Recommendation 7
Fixed Swap Address 9
Description 9
Recommendation 9
BC - Blacklisted Contracts 10
Description 10
Recommendation 10
Contract Diagnostics 11
MAL - Misused Algorithmic Logic 12
Description 12
Recommendation 12
DSM - Data Structure Misuse 13
Description 13
Recommendation 13
L01 - Public Function could be Declared External 14
Description 14
Recommendation 14

L02 - State Variables could be Declared Constant 15
Description 15
Recommendation 15
L05 - Unused State Variable 16
Description 16
Recommendation 16
L04 - Conformance to Solidity Naming Conventions 17
Description 17
Recommendation 17
L09 - Dead Code Elimination 18
Description 18
Recommendation 18
L07 - Missing Events Arithmetic 19
Description 19
Recommendation 19
L13 - Divide before Multiply Operation 20
Description 20
Recommendation 20
Contract Functions 21
Contract Flow 27
Domain Info 28
Summary 29
Disclaimer 30
About Coinscope 31

Contract Review

Github	https://github.com/Elevate-Software/ShibaSafe
Commit	499a806dc70b3560533d054935add41c0cccfa87
SHA256	8d7fa92e9f9418915729bb509b280c66d11b49f02639f6bcb609afb9eb6c3d32
Domain	shibasafe.com

Audit Updates

Initial Audit	16th February 2022
Corrected	

Contract Analysis

Critical Medium Minor Pass

Severity	Code	Description
●	ST	Contract Owner is not able to stop or pause transactions
●	OCTD	Contract Owner is not able to transfer tokens from specific address
●	OTUT	Owner Transfer User's Tokens
●	ELFM	Contract Owner is not able to increase fees more than a reasonable percent (25%)
●	ULTW	Contract Owner is not able to increase the amount of liquidity taken by dev wallet more than a reasonable percent
●	MT	Contract Owner is not able to mint new tokens
●	BT	Contract Owner is not able to burn tokens from specific wallet
●	BC	Contract Owner is not able to blacklist wallets from selling

ST - Stop Transactions

Criticality	critical
Location	contract.sol#L1021,1025,1060

Description

The contract owner has the authority to stop the sales for all users excluding the owner. The owner may take advantage of it by following the steps:

- _setStakingFee(1)
- _setFutureFee(10)
- _setUseFee(10)
- _setMarketingFee(10)

At this state the variables will be:

- _stakingFee = 1
- _totalFee = 31

```
if(msg.sender != address(uniswapV2Router)){ // sell
    uint256 takeStakingReward = amount.sub(amount.mul(_stakingFee/_totalFee));
    _stakingWalletAddress.transfer(takeStakingReward);
    amount -= takeStakingReward;
}
```

The following calculation will be yielded:

```
takeStakingReward = amount - amount * (1/32)
takeStakingReward = amount - amount * (1/32)
takeStakingReward = amount * (31/32)
```

The amount will decreased to

```
amount = amount - amount * (31/32)
amount = amount * (1/32)
```

This amount is as low as stopping the sale transactions.

The contract owner has the authority to stop transactions for all users excluding the owner. The owner may take advantage of it by setting the `_maxTxAmount` or `_maxWalletSize` to zero.

```
if(sender != owner() && recipient != owner() && sender != address(this) &&
recipient != address(this)) {
    require(amount <= _maxTxAmount, "Transfer amount exceeds the maxTxAmount.");
```

```
require(tokenBalanceRecipient + amount <= _maxWalletSize, "Recipient exceeds max
wallet size.");
```

Recommendation

The contract should not allow huge fees in the amount.

The contract could embody a check for not allowing setting the *_maxTxAmount* or *_maxWalletSize* less than a reasonable amount. A suggested implementation could check that the maximum amount should be more than a fixed percentage of the total supply.

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. That risk can be prevented by temporarily locking the contract or renouncing ownership.

ULTW - Unlimited Liquidity to Team Wallet

Criticality	medium
Location	contract.sol#L1041,1106

Description

The contract owner has the authority to transfer a lot of funds to the team wallet. These funds have been accumulated from the swap & liquify feature. All the accumulated amount from the team fees is converted to BNB and sended to the team's wallets.

Additionally, the contract owner can manually trigger the swap & liquidy feature by calling the *manualSwap* and *manualSend* sequentially.

As a result, when the swap & liquify is triggered, there will be a huge sale in the exchange. There is a feature to balance this sale, like auto generated liquidity pool, buyback etc.

```
if (!inSwap && swapEnabled && overMinTokenBalance && sender != uniswapV2Pair) {
    // Swap tokens for ETH and send to resepctive wallets
    swapTokensForEth(contractTokenBalance); // swaps native token for ETH

    uint256 contractETHBalance = address(this).balance;
    if(contractETHBalance > 0) {
        sendETHToTeam(address(this).balance); // send ETH to respective wallets
    }
}
```

```
function manualSwap() external onlyOwner() {
    uint256 contractBalance = balanceOf(address(this));
    swapTokensForEth(contractBalance);
}

function manualSend() external onlyOwner() {
    uint256 contractETHBalance = address(this).balance;
    sendETHToTeam(contractETHBalance);
}
```

Recommendation

The contract should not allow massive token sales.

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. That risk can be prevented by temporarily locking the contract or renouncing ownership.

Fixed Swap Address

Description

The swap address is assigned once in the constructor and it can not be changed. The decentralized swaps sometimes create a new swap version or abandon the current. A contract that cannot change the swap address may not be able to catch-up the upgrade.

```
IUniswapV2Router02 _uniswapV2Router =
IUniswapV2Router02(0x10ED43C718714eb63d5aA57B78B54704E256024E);
// Create a uniswap pair for this new token
uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory())
    .createPair(address(this), _uniswapV2Router.WETH());

// set the rest of the contract variables
uniswapV2Router = _uniswapV2Router;
```

Recommendation

It could be better to allow the swap address mutation in case of future swap updates.

BC - Blacklisted Contracts

Criticality	medium
Location	contract.sol#L1017

Description

The contract owner has the authority to stop contracts from transactions. The owner may take advantage of it by calling the `addBotToBlacklist` function.

```
require(!_isBlackListedBot[sender], "You are blacklisted");
require(!_isBlackListedBot[msg.sender], "You are blacklisted");
require(!_isBlackListedBot[tx.origin], "You are blacklisted");
```

Recommendation

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. That risk can be prevented by temporarily locking the contract or renouncing ownership.

Contract Diagnostics

Critical Medium Minor

Severity	Code	Description
●	MAL	Misused Algorithmic Logic
●	DSM	Data Structure Misuse
●	L01	Public Function could be Declared External
●	L02	State Variables could be Declared Constant
●	L05	Unused State Variable
●	L04	Conformance to Solidity Naming Conventions
●	L09	Dead Code Elimination
●	L07	Missing Events Arithmetic
●	L13	Divide before Multiply Operation

MAL - Misused Algorithmic Logic

Criticality	medium
Location	contract.sol#L1199

Description

The algorithmic flow does not follow the required business logic.

The contract assumes that the transactions where the sender is not the swap address are sales. There are cases like pure transfers from wallet to wallet that this assumption does not work.

```
if(msg.sender == address(uniswapV2Router)){ // buy
    uint256 tFee = tAmount.mul(buyTaxFee).div(100);
    uint256 tTeam = tAmount.mul(buyTeamFee).div(100);
    uint256 tTransferAmount = tAmount.sub(tFee).sub(tTeam);
    return (tTransferAmount, tFee, tTeam);
}else if(msg.sender != address(uniswapV2Router)){ // sell
    uint256 tFee = tAmount.mul(taxFee).div(100);
    uint256 tTeam = tAmount.mul(teamFee).div(100);
    uint256 tTransferAmount = tAmount.sub(tFee).sub(tTeam);
    return (tTransferAmount, tFee, tTeam);
}
```

Recommendation

The algorithm should be reshaped so it will match to the business logic.

DSM - Data Structure Misuse

Criticality	minor
Location	contract.sol#L907,914

Description

The contract uses the `_isBlackListedBot` and the `_blackListedBots` in order to store the blacklisted addresses. The `_blackListedBots` is not used anywhere in the contract. Hence, it merely produces unnecessary gas consumption.

```
function addBotToBlacklist (address account) external onlyOwner() {
    require(account != 0x10ED43C718714eb63d5aA57B78B54704E256024E, 'We cannot
blacklist UniSwap router');
    require (!_isBlackListedBot[account], 'Account is already blacklisted');
    _isBlackListedBot[account] = true;
    _blackListedBots.push(account);
}

function removeBotFromBlacklist(address account) external onlyOwner() {
    require (_isBlackListedBot[account], 'Account is not blacklisted');
    for (uint256 i = 0; i < _blackListedBots.length; i++) {
            if (_blackListedBots[i] == account) {
                _blackListedBots[i] = _blackListedBots[_blackListedBots.length -
1];
                _isBlackListedBot[account] = false;
                _blackListedBots.pop();
                break;
            }
    }
}
```

Recommendation

The `_blackListedBots` could be eliminated from the contract since it is not used.

L01 - Public Function could be Declared External

Criticality	minor
Location	contract.sol#L420,429,818,822,826,830,839,844,848,853 and 22 more

Description

Public functions that are never called by the contract should be declared external to save gas.

```
_getETHBalance
_getMaxTxAmount
_getFutureFee
_getStakingFee
_getBuyUseFee
_getUseFee
_getBuyFutureFee
_getBuyMarketingFee
_getMarketingFee
...
```

Recommendation

Use the external attribute for functions never called from the contract

L02 - State Variables could be Declared Constant

Criticality	minor
Location	contract.sol#L386

Description

Constant state variables should be declared constant to save gas.

```
_previousOwner
```

Recommendation

Add the constant attribute to state variables that never change.

L05 - Unused State Variable

Criticality	minor
Location	contract.sol#L386

Description

There are segments that contain unused state variables.

```
_previousOwner
```

Recommendation

Remove unused state variables.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor
Location	contract.sol#L491,492,521,540,786,787,1238,1242,1247,1251 and 33 more

Description

Solidity defines a naming convention that should be followed. Rule exceptions:

- Allow constant variable name/symbol/decimals to be lowercase.
- Allow _ at the beginning of the mixed_case match for private variables and unused parameters.

```
_numOfTokensToExchangeForTeam
_futureFeeWalletAddress
_stakingWalletAddress
_useCaseWalletAddress
_marketingWalletAddress
_decimals
_symbol
_name
_tTotal
...
```

Recommendation

Follow the Solidity naming convention.
https://docs.soliditylang.org/en/v0.4.25/style-guide.html#naming-conventions

L09 - Dead Code Elimination

Criticality	minor
Location	contract.sol#L360,320,330,345,355,267,294,14,227,243

Description

Functions that are not used in the contract, and make the code's size bigger.

```
mod
_msgData
sendValue
isContract
functionCallWithValue
functionCall
_functionCallWithValue
...
```

Recommendation

Remove unused functions.

L07 - Missing Events Arithmetic

Criticality	minor
Location	contract.sol#L881,1346,1354,1361,1368,1376,1384,1391,1414

Description

Detected missing events for critical arithmetic parameters. There are functions that have no event emitted, so it is difficult to track off-chain changes.

```
_maxTxAmount = maxTxAmount
_futureFee = futureFee
_stakingFee = stakingFee
_buyUseFee = buyUseFee
_useFee = useFee
_buyFutureFee = buyFutureFee
_buyMarketingFee = buyMarketingFee
_marketingFee = marketingFee
_tFeeTotal = _tFeeTotal.add(tAmount)
```

Recommendation

Emit an event for critical parameter changes.

L13 - Divide before Multiply Operation

Criticality	minor
Location	contract.sol#L1013,1087

Description

Performing divisions before multiplications may cause lose of prediction.

```
_futureFeeWalletAddress.transfer(amount.mul(_futureFee / _totalFee))
_futureFeeWalletAddress.transfer(amount.mul(_buyFutureFee / _buyTotalFee))
_useCaseWalletAddress.transfer(amount.mul(_useFee / _totalFee))
_useCaseWalletAddress.transfer(amount.mul(_buyUseFee / _buyTotalFee))
_marketingWalletAddress.transfer(amount.mul(_marketingFee / _totalFee))
_marketingWalletAddress.transfer(amount.mul(_buyMarketingFee / _buyTotalFee))
takeStakingReward = amount.sub(amount.mul(_stakingFee / _totalFee))
```

Recommendation

The multiplications should be prior to the divisions.

Contract Functions

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		
	mod	Internal		
	mod	Internal		
Address	Library			
	isContract	Internal		
	sendValue	Internal	✓	
	functionCall	Internal	✓	
	functionCall	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionCallWithValue	Internal	✓	

	_functionCallWithValue	Private	✓	
Ownable	Implementation	Context		
	<Constructor>	Internal	✓	
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
IUniswapV2Factory	Interface			
	feeTo	External		-
	feeToSetter	External		-
	getPair	External		-
	allPairs	External		-
	allPairsLength	External		-
	createPair	External	✓	-
	setFeeTo	External	✓	-
	setFeeToSetter	External	✓	-
IUniswapV2Pair	Interface			
	name	External		-
	symbol	External		-
	decimals	External		-
	totalSupply	External		-
	balanceOf	External		-
	allowance	External		-
	approve	External	✓	-
	transfer	External	✓	-
	transferFrom	External	✓	-
	DOMAIN_SEPARATOR	External		-
	PERMIT_TYPEHASH	External		-
	nonces	External		-
	permit	External	✓	-
	MINIMUM_LIQUIDITY	External		-
	factory	External		-

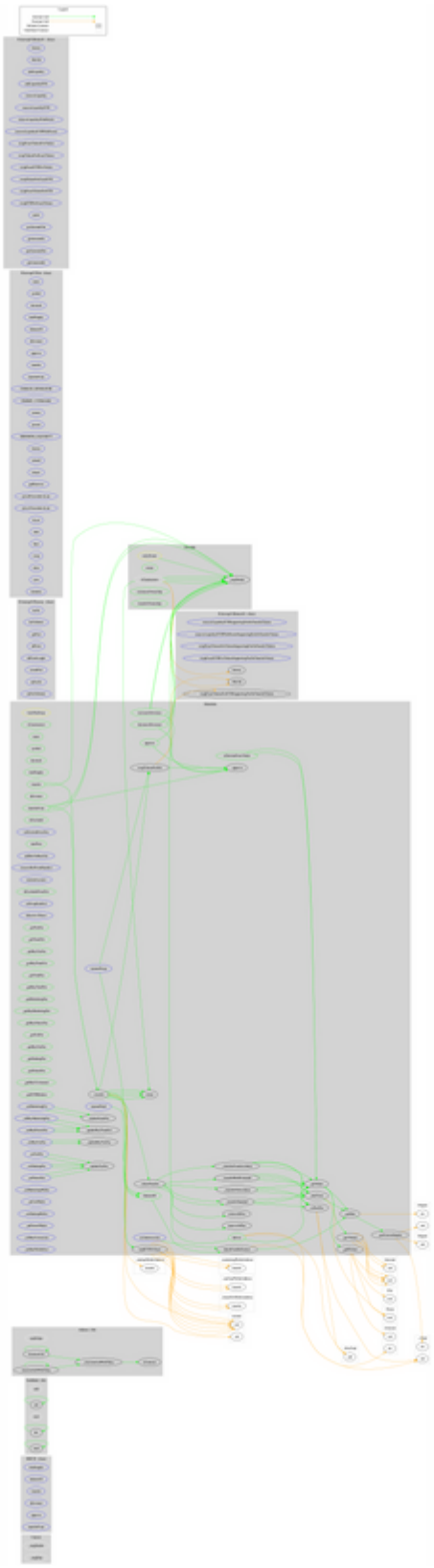
	token0	External		-
	token1	External		-
	getReserves	External		-
	price0CumulativeLast	External		-
	price1CumulativeLast	External		-
	kLast	External		-
	mint	External	✓	-
	burn	External	✓	-
	swap	External	✓	-
	skim	External	✓	-
	sync	External	✓	-
	initialize	External	✓	-
IUniswapV2Router01	Interface			
	factory	External		-
	WETH	External		-
	addLiquidity	External	✓	-
	addLiquidityETH	External	Payable	-
	removeLiquidity	External	✓	-
	removeLiquidityETH	External	✓	-
	removeLiquidityWithPermit	External	✓	-
	removeLiquidityETHWithPermit	External	✓	-
	swapExactTokensForTokens	External	✓	-
	swapTokensForExactTokens	External	✓	-
	swapExactETHForTokens	External	Payable	-
	swapTokensForExactETH	External	✓	-
	swapExactTokensForETH	External	✓	-
	swapETHForExactTokens	External	Payable	-
	quote	External		-
	getAmountOut	External		-
	getAmountIn	External		-
	getAmountsOut	External		-
	getAmountsIn	External		-

IUniswapV2Router02	Interface	IUniswapV2Router01		
	removeLiquidityETHSupportingFeeOnTransferTokens	External	✓	-
	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External	✓	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
ShibaSafe	Implementation	Context, IERC20, Ownable		
	<Constructor>	Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	isExcluded	Public		-
	setExcludeFromFee	External	✓	onlyOwner
	totalFees	Public		-
	deliver	Public	✓	-
	reflectionFromToken	Public		-
	tokenFromReflection	Public		-
	addBotToBlacklist	External	✓	onlyOwner
	removeBotFromBlacklist	External	✓	onlyOwner
	excludeAccount	External	✓	onlyOwner
	includeAccount	External	✓	onlyOwner

	removeAllFee	Private	✓	
	restoreAllFee	Private	✓	
	isExcludedFromFee	Public		-
	_approve	Private	✓	
	_transfer	Private	✓	
	swapTokensForEth	Private	✓	lockTheSwap
	sendETHToTeam	Private	✓	
	manualSwap	External	✓	onlyOwner
	manualSend	External	✓	onlyOwner
	setSwapEnabled	External	✓	onlyOwner
	_tokenTransfer	Private	✓	
	_transferStandard	Private	✓	
	_transferToExcluded	Private	✓	
	_transferFromExcluded	Private	✓	
	_transferBothExcluded	Private	✓	
	_takeTeam	Private	✓	
	_reflectFee	Private	✓	
	<Receive Ether>	External	Payable	-
	_getValues	Private		
	_getTValues	Private		
	_getRValues	Private		
	_getRate	Private		
	_getCurrentSupply	Private		
	_getTaxFee	Public		-
	_getTeamFee	Public		-
	_getBuyTaxFee	Public		-
	_getBuyTeamFee	Public		-
	_getTotalFee	Public		-
	_getBuyTotalFee	Public		-
	_getMarketingFee	Public		-
	_getBuyMarketingFee	Public		-
	_getBuyFutureFee	Public		-
	_getUseFee	Public		-
	_getBuyUseFee	Public		-
	_getStakingFee	Public		-

	_getFutureFee	Public		-
	_getMaxTxAmount	Public		-
	_getETHBalance	Public		-
	_updateTeamFee	Private	✓	
	_updateBuyTeamFee	Private	✓	
	_updateTaxFee	Private	✓	
	_updateBuyTaxFee	Private	✓	
	_setMarketingFee	External	✓	onlyOwner
	_setBuyMarketingFee	External	✓	onlyOwner
	_setBuyFutureFee	External	✓	onlyOwner
	_setUseFee	External	✓	onlyOwner
	_setBuyUseFee	External	✓	onlyOwner
	_setStakingFee	External	✓	onlyOwner
	_setFutureFee	External	✓	onlyOwner
	_setMarketingWallet	External	✓	onlyOwner
	_setUseWallet	External	✓	onlyOwner
	_setStakingWallet	External	✓	onlyOwner
	_setFutureWallet	External	✓	onlyOwner
	_setMaxTxAmount	External	✓	onlyOwner
	_setMaxWalletSize	External	✓	onlyOwner

Contract Flow



Domain Info

Domain Name	shibasafe.com
Registry Domain ID	2659543673_DOMAIN_COM-VRSN
Creation Date	2021-12-04T21:03:54Z
Updated Date	2021-12-04T21:03:54Z
Registry Expiry Date	2022-12-04T21:03:54Z
Registrar WHOIS Server	whois.godaddy.com
Registrar URL	http://www.godaddy.com
Registrar	GoDaddy.com, LLC
Registrar IANA ID	146

The domain has been created 2 months before the creation of the audit. It will expire in 10 months.

There is no public billing information, the creator is protected by the privacy settings.

Summary

There are some functions that can be abused by the owner, like blacklisting addresses, transferring funds to the team's wallet and stopping transactions. If the contract configuration is abused from the contract owner, then the contract may operate similarly to a honeypot. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats.

In the scope of the audit we focus on security, performance optimizations and business logic recommendations.

Disclaimer

All the content provided in this document is for general information only and should not be used as financial advice or a reason to buy any investment.

Coinscope team provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document. Always Do your own research and protect yourselves from being scammed.

The Coinscope team has audited this project for general information and only expresses their opinion based on similar projects and checks from popular diagnostic tools. Under no circumstances did Coinscope receive a payment to manipulate those results or change the awarding badge that we will be adding in our website.

Always Do your own research and protect yourselves from scams. This document should not be presented as a reason to buy or not buy any particular token.

The Coinscope team disclaims any liability for the resulting losses.

About Coinscope

Coinscope is the leading early coin listing, voting and auditing authority firm. The audit process is analyzing and monitoring many aspects of the project. That way, it gives the community a good sense of security using an informative report and a generic score.

Coinscope is aiming to make crypto discoverable and efficient globally. It provides all the essential tools to assist users draw their own conclusions.

The Coinscope.co team

https://www.coinscope.co