Audit Report

ANTCOIN

July 2022

Type	BEP20
Networ	BSC
Addres	0xc974eefb38cb5e814f8bde916187
Audited by	© SecureZilla

Contents

Contract Review

Contract Name	Antcoin
Compiler Version	v0.8.4+commit.c7e474f2
Optimization	200 runs
Licence	MIT
Explorer	https://bscscan.com/token/0xc974eefb38cb5e814f8bde916187395292aa6b27
Symbol	Ants
Decimals	9
Total Supply	1,000,000,000
Domain	antcoin.app

Source Files

Filename	**SHA256**
contract.sol	67d5b7803f50c7a07cc6770468bb46735e03dba95df7c7c517552dd39fe9b8ab

Audit Updates

Initial Audit	8th July 2022
Corrected	

Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 +	A vulnerability that creates a risk that the contract may be damaged.	Instant action to lessen risk level.
High	7 +	A vulnerability that disturbs the anticipated outcome when using contract.	Application of counteractive actions asap.
Medium	4 +	A vulnerability that could affect the contract in a specific scenario.	Execution of educative actions in a certain period.
Low	2 +	A vulnerability that does not have a major effect.	Operation of certain helpful actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational appeal but is not effecting any of the code.	a comment that does not control a level of risk.

Contract Analysis

Critical High Medium Low

Severity	Code	Description
	ST	Contract Owner is not able to stop or pause transactions
	OCTT	Contract Owner is not able to transfer tokens from specific address
	OTUT	Owner Transfer User's Tokens
	IFRL	Contract Owner is not able to increase fees more than a reasonable percent (25%)
	ULTO	Contract Owner is not able to increase the amount of liquidity taken by dev wallet more than a reasonable percent
	MNT	Contract Owner is not able to mint new tokens
	BTSW	Contract Owner is not able to burn tokens from specific wallet
	BWS	Contract Owner is not able to blacklist wallets from selling

ST - Stop Transactions

Criticality	critical
Location	contract.sol#L585,612

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 setting the `_liquidityFee` to a high value. As a result, the amount will not be sufficient and the transaction will revert.

```
uint256 marketingBnb =
transferredBalance.div(_liquidityFee).mul(marketingDivisor);
uint256 liquidityBnb =
transferredBalance.div(_liquidityFee).mul(liquidityDivisor);
transferToAddressETH(marketingAddress, marketingBnb);
transferToAddressETH(liquidityAddress, liquidityBnb);
```

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` to zero.

```
if(from != owner() && to != owner()) {
    require(amount <= _maxTxAmount, "Transfer amount exceeds the maxTxAmount.");
}
```

Recommendation

The contract could embody a check for not allowing setting the _maxTxAmount 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 `swapTokens` method should not be called if the `_liquidityFee` is zero.

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.

IFRL – Increase Fee than a Reasonable Limit

Criticality	critical
Location	contract.sol#L828,832

Description

The contract owner has the authority to increase over the allowed limit of 25%. The owner may take advantage of it by calling the `setTaxFeePercent` function with a high percentage value.

```
function setTaxFeePercent(uint256 taxFee) external onlyOwner() {
    _taxFee = taxFee;
}
```

Recommendation

The contract could embody a check for the maximum acceptable value.

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.

ULTO - Unlimited Liquidity to Owner

Criticality	low
Location	contract.sol#L887

Description

The contract owner has the authority to transfer funds without limit to the team wallet. These funds have been accumulated from fees collected from the contract. The owner may take advantage of it by calling the `doManualSwapTokens` and `recoverBalance` methods.

```
function recoverBalance(uint256 amount) public onlyOwner {
    payable(owner()).transfer(amount);
}


function doManualSwapTokens(uint256 tokensAmount) public
{
    swapTokens(tokensAmount);
}
```

Recommendation

The contract could embody a check for the maximum amount of funds that can be swapped. Since a huge amount may volatile the token's price.

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 Low

Severity	Code	Description
	S01	Public Function could be Declared External
	S02	State Variables could be Declared Constant
	S04	Conformance to Solidity Naming Conventions
	S07	Missing Events Arithmetic
	S09	Dead Code Elimination
	S13	Divide before Multiply Operation

S01 - Public Function could be Declared External

Criticality	low
Location	contract.sol#L157,162,168,172,176,183,455,459,463,467,476,481,485,490,496,501,506,510,514,519,529,546,816,820,824,887,892

Description

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

```
doManualSwapTokens
recoverBalance
includeInFee
excludeFromFee
isExcludedFromFee
excludeFromReward
reflectionFromToken
deliver
minimumTokensBeforeSwapAmount
...
```

Recommendation

Use the external attribute for functions never called from the contract.

S02 - State Variables could be Declared Constant

Criticality	low
Location	contract.sol#L410,408,409,404,424

Description

Constant state variables should be declared constant to save gas.

```
liquidityDivisor
_tTotal
_symbol
_name
_decimals
```

Recommendation

Add the constant attribute to state variables that never change.

S04 - Conformance to Solidity Naming Conventions

Criticality	low
Location	contract.sol#L219,220,229,249,787,793,844,850,855,859,413,416,419,426

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.

```
_maxTxAmount
_burnFee
_liquidityFee
_taxFee
_enabled
_newaddress
_marketingAddress
_minimumTokensBeforeSwap
_amount
...
```

Recommendation

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

S07 - Missing Events Arithmetic

Criticality	low
Location	contract.sol#L828,832,836,840,844

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.

```
minimumTokensBeforeSwap = _minimumTokensBeforeSwap
marketingDivisor = divisor
_maxTxAmount = maxTxAmount
_liquidityFee = liquidityFee
_taxFee = taxFee
```

Recommendation

Emit an event for critical parameter changes.

S09 - Dead Code Elimination

Criticality	low
Location	contract.sol#L115,98,102,106,110,78,89,656,639

Description

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

```
swapETHForTokens
addLiquidity
sendValue
isContract
functionCallWithValue
functionCall
_functionCallWithValue
```

Recommendation

Remove unused functions.

S13 - Divide before Multiply Operation

Criticality	low
Location	contract.sol#L607,689

Description

Performing divisions before multiplications may cause lose of prediction.

```
tBurnAmount = tAmount.div(100).mul(_burnFee)
liquidityBnb = transferredBalance.div(_liquidityFee).mul(liquidityDivisor)
marketingBnb = transferredBalance.div(_liquidityFee).mul(marketingDivisor)
```

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>	Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	getUnlockTime	Public		-
	getTime	Public		-
	lock	Public	✓	onlyOwner
	unlock	Public	✓	-
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		-

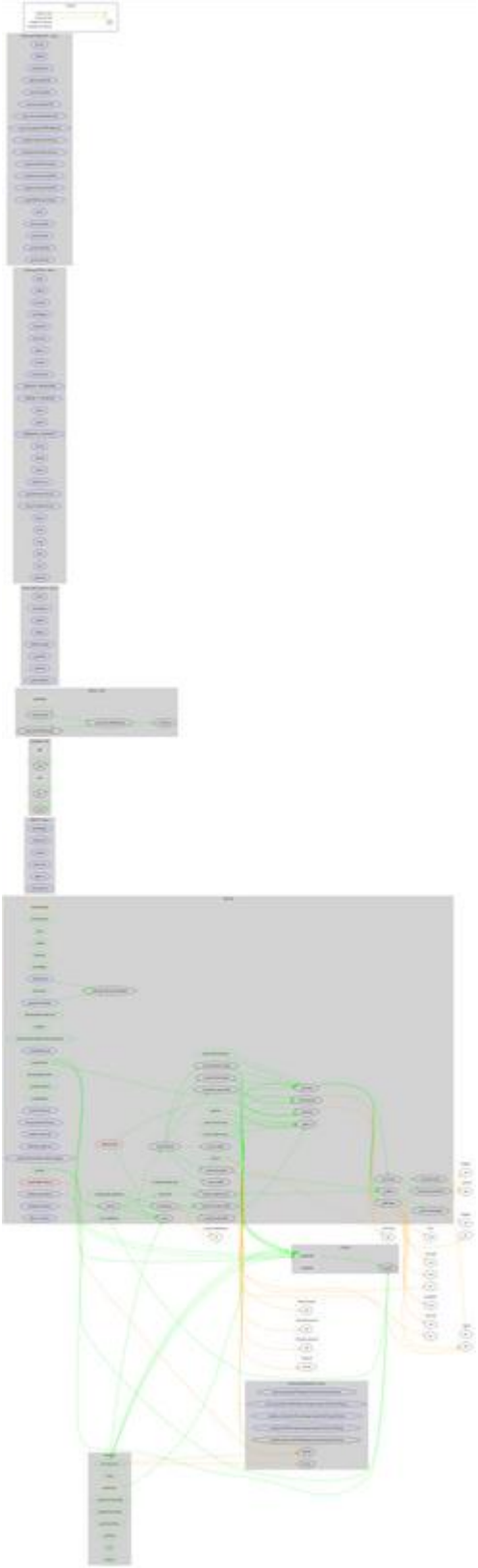
	nonces	External		-
	permit	External	✓	-
	MINIMUM_LIQUIDITY	External		-
	factory	External		-
	token0	External		-
	token1	External		-
	getReserves	External		-
	price0CumulativeLast	External		-
	price1CumulativeLast	External		-
	kLast	External		-
	burn	External	✓	-
	swap	External	✓	-
	skim	External	✓	-
	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	✓	-
Antcoin	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	✓	-
	isExcludedFromReward	Public		-
	totalFees	Public		-
	minimumTokensBeforeSwapAmount	Public		-
	deliver	Public	✓	-
	reflectionFromToken	Public		-
	tokenFromReflection	Public		-
	excludeFromReward	Public	✓	onlyOwner
	includeInReward	External	✓	onlyOwner

	_approve	Private	✓	
	_transfer	Private	✓	
	swapTokens	Private	✓	lockTheSwap
	swapTokensForEth	Private	✓	
	swapETHForTokens	Private	✓	
	addLiquidity	Private	✓	
	_tokenTransfer	Private	✓	
	_transferStandard	Private	✓	
	_transferToExcluded	Private	✓	
	_transferFromExcluded	Private	✓	
	_transferBothExcluded	Private	✓	
	_reflectFee	Private	✓	
	_getValues	Private		
	_getTValues	Private		
	_getRValues	Private		
	_getRate	Private		
	_getCurrentSupply	Private		
	_takeLiquidity	Private	✓	
	calculateTaxFee	Private		
	calculateLiquidityFee	Private		
	removeAllFee	Private	✓	
	restoreAllFee	Private	✓	
	isExcludedFromFee	Public		-
	excludeFromFee	Public	✓	onlyOwner
	includeInFee	Public	✓	onlyOwner
	setTaxFeePercent	External	✓	onlyOwner
	setLiquidityFeePercent	External	✓	onlyOwner
	setMaxTxAmount	External	✓	onlyOwner
	setMarketingDivisor	External	✓	onlyOwner
	setNumTokensSellToAddToLiquidity	External	✓	onlyOwner
	setMarketingAddress	External	✓	onlyOwner
	setliquidityAddress	External	✓	onlyOwner
	setSwapAndLiquifyEnabled	Public	✓	onlyOwner
	prepareForPreSale	External	✓	onlyOwner
	afterPreSale	External	✓	onlyOwner

	transferToAddressETH	Public	✓	-
	recoverBalance	Public	✓	onlyOwner
	doManualSwapTokens	Public	✓	-
	<Receive Ether>	External	Payable	-

Contract Flow



Secure Zilla

Domain Info

Antcoin Token Audit

Domain Name	antcoin.app
Registry Domain ID	494D0CA2C-APP
Creation Date	2022-06-25T20:09:16Z
Updated Date	2022-06-30T20:09:16Z
Registry Expiry Date	2023-06-25T20:09:16Z
Registrar WHOIS Server	whois.nic.google
Registrar URL	None
Registrar	NameSilo, LLC
Registrar IANA ID	1479

The domain has been created in 12 months before the creation of the audit.

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 stopping transactions, manipulating fees and transferring funds to the team's wallet. The contract can be converted into a honeypot and prevent users from selling if the owner abuses the admin functions. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats.

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.

SecureZilla 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 SecureZilla 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 SecureZilla 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 SecureZilla team disclaims any liability for the resulting losses.

About SecureZilla

SecureZilla is aiming to make crypto discoverable and efficient globally. They provide all the essential tools to assist users draw their own conclusions.

The SecureZilla team

https://www.SecureZilla.io