Cyberscope

Audit Report

Delta Ace

March 2022

Type	BEP20
Network	BSC
Address	0x73F9F594b2F94dfe176a6c014F95a32e40bD0d77
Audited by	© cyberscope

Table of Contents

Contract Review

Contract Name	deltaace
Compiler Version	v0.8.4+commit.c7e474f2
Optimization	200 runs
Licence	Unlicense
Explorer	https://bscscan.com/token/0x73F9F594b2F94dfe176a6c014F95a32e40bD0d77
Symbol	ACED
Decimals	18
Total Supply	10,000,000,000
Domain	deltaace.org

Source Files

Filename	**SHA256**
contract.sol	b848e05228f8b98f217464f8ec1a7c90592f61c760087633e0280ecc3d274785

Audit Updates

Initial Audit	26th March 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#L1326,995,1027

Description

The contract owner has the authority to stop transactions for all users excluding the owner. He can do that in multiple ways. He can increase the `sellFeeIncreaseFactor` to a very high percent and convert the contract into a **HONEYPOT**

```
    if(automatedMarketMakerPairs[to]) {
       fees = fees.div(100).mul(sellFeeIncreaseFactor);
    }
```

He can also convert it into a honeypot and prevent users from selling by setting the `maxSellTransactionAmount` to zero.

```
  else if (
              tradingIsEnabled &&
             automatedMarketMakerPairs[to] &&
             !excludedAccount
         ) {
             require(amount <= maxSellTransactionAmount, "Sell transfer amount
exceeds the maxSellTransactionAmount.");
```

The owner may take advantage of it by setting the `maxWalletBalance` to zero.

```
   if(!automatedMarketMakerPairs[to] && tradingIsEnabled && !excludedAccount){
             require(balanceOf(to).add(amount) <= maxWalletBalance, 'Wallet
balance is exceeding maxWalletBalance');
         }
```

Recommendation

The contract could embody a check for not allowing setting the maxSellTransactionAmount and maxWalletBalance 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.

He could also not allow the sellFeeIncreaseFactor to be higher than a reasonable amount.

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.

ELFM - Exceed Limit Fees Manipulation

Criticality	critical
Location	contract.sol#L1007,1015,1023

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 `setMarketingFeeSell` function with a high percentage value.

```
    function setMarketingFeeSell(uint256 fee) external onlyOwner {
          _marketingFeeSell = fee * (10**18);
    }
```

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.

Contract Diagnostics

● Critical ● Medium ● Minor

Severity	Code	Description
●	L01	Public Function could be Declared External
●	L02	State Variables could be Declared Constant
●	L04	Conformance to Solidity Naming Conventions
●	L05	Unused State Variable
●	L07	Missing Events Arithmetic
●	L09	Dead Code Elimination
●	L11	Unnecessary Boolean equality
●	L12	Using Variables before Declaration
●	L13	Divide before Multiply Operation
●	L14	Uninitialized Variables in Local Scope
●	L15	Local Scope Variable Shadowing

L01 - Public Function could be Declared External

Criticality	minor
Location	contract.sol#L37,42,86,90,94,106,111,115,120,126 and 13 more

Description

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

```
getIsExcludedFromFees
setAutomatedMarketMakerPair
process
getAccountAtIndex
size
getKeyAtIndex
getIndexOfKey
get
withdrawnDividendOf
...
```

Recommendation

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

L02 - State Variables could be Declared Constant

Criticality	minor
Location	contract.sol#L215,887,892,855,883,858

Description

Constant state variables should be declared constant to save gas.

```
tradingIsEnabled
previousBuybackFee
deadAddress
_busdDividendRewardsFeeSell
_busdDividendRewardsFeeBuy
lastAmount
```

Recommendation

Add the constant attribute to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor
Location	contract.sol#L281,285,289,294,212,222,369,370,387,407 and 27 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.

```
_buybackFeeSell
_marketingFeeSell
_LpFeeSell
_busdDividendRewardsFeeSell
_buybackFeeBuy
_LpFeeBuy
_marketingFeeBuy
_busdDividendRewardsFeeBuy
_dividendAddress
...
```

Recommendation

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

L05 - Unused State Variable

Criticality	minor
Location	contract.sol#L215

Description

There are segments that contain unused state variables.

```
lastAmount
```

Recommendation

Remove unused state variables.

L07 - Missing Events Arithmetic

Criticality	minor
Location	contract.sol#L995,999,1003,1007,1011,1015,1019,1023,1027,1044 and 1 more

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.

```
sellFeeIncreaseFactor = _multiplier
swapTokensAtAmount = _swapAmount * (10 ** 18)
maxSellTransactionAmount = _maxTxn * (10 ** 18)
_LpFeeSell = LpFeeSell * (10 ** 18)
_buybackFeeSell = fee * (10 ** 18)
_marketingFeeSell = fee * (10 ** 18)
_LpFeeBuy = LpFeeBuy * (10 ** 18)
_buybackFeeBuy = fee * (10 ** 18)
_marketingFeeBuy = fee * (10 ** 18)
...
```

Recommendation

Emit an event for critical parameter changes.

L09 - Dead Code Elimination

Criticality	minor
Location	contract.sol#L299,175,1436

Description

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

```
transferToWallet
_setupDecimals
_transfer
```

Recommendation

Remove unused functions.

L11 - Unnecessary Boolean equality

Criticality	minor
Location	contract.sol#L1061,1073,1084

Description

The comparison to boolean constants is redundant. Boolean constants can be used directly and do not need to be compared to true or false.

```
_enabled == false
```

Recommendation

Remove the equality to the boolean constant.

L12 - Using Variables before Declaration

Criticality	minor
Location	contract.sol#L1349

Description

The contract is using a variable before the declaration. This is usually happening either if it has not been declared yet or the variable has been declared in a different scope.

```
iterations
claims
lastProcessedIndex
```

Recommendation

The variables should be declared before any usage of them.

L13 - Divide before Multiply Operation

Criticality	minor
Location	contract.sol#L1242

Description

Performing divisions before multiplications may cause lose of prediction.

```
fees = fees.div(100).mul(sellFeeIncreaseFactor)
```

Recommendation

The multiplications should be prior to the divisions.

L14 - Uninitialized Variables in Local Scope

Criticality	minor
Location	contract.sol#L1349

Description

The are variables that are defined in the local scope and are not initialized.

```
iterations
lastProcessedIndex
claims
```

Recommendation

All the local scoped variables should be initialized.

L15 - Local Scope Variable Shadowing

Criticality	minor
Location	contract.sol#L231,281,285,289,294

Description

The are variables that are defined in the local scope containing the same name from an upper scope.

```
_owner
_symbol
_name
```

Recommendation

The local variables should have different names from the upper scoped variables.

Contract Functions

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Ownable	Implementation	Context		
	<Constructor>	Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
ERC20	Implementation	Context, IERC20		
	<Constructor>	Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-

	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	_transfer	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_approve	Internal	✓	
	_setupDecimals	Internal	✓	
	_beforeTokenTransfer	Internal	✓	
IDividendPayingToken	Interface			
	dividendOf	External		-
	withdrawDividend	External	✓	-
IDividendPayingTokenOptional	Interface			
	withdrawableDividendOf	External		-
	withdrawnDividendOf	External		-
	accumulativeDividendOf	External		-
DividendPayingToken	Implementation	ERC20, IDividendPayingToken, IDividendPayingTokenOptional, Ownable		
	<Constructor>	Public	✓	ERC20
	setAuth	External	✓	onlyOwner
	distributeDividends	Public	✓	onlyOwner
	withdrawDividend	Public	✓	-
	setDividendTokenAddress	External	✓	onlyOwner
	_withdrawDividendOfUser	Internal	✓	
	dividendOf	Public		-
	withdrawableDividendOf	Public		-
	withdrawnDividendOf	Public		-

	accumulativeDividendOf	Public		-
	_transfer	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_setBalance	Internal	✓	
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		
	removeLiquidityETHSupportingFeeOn	External	✓	-

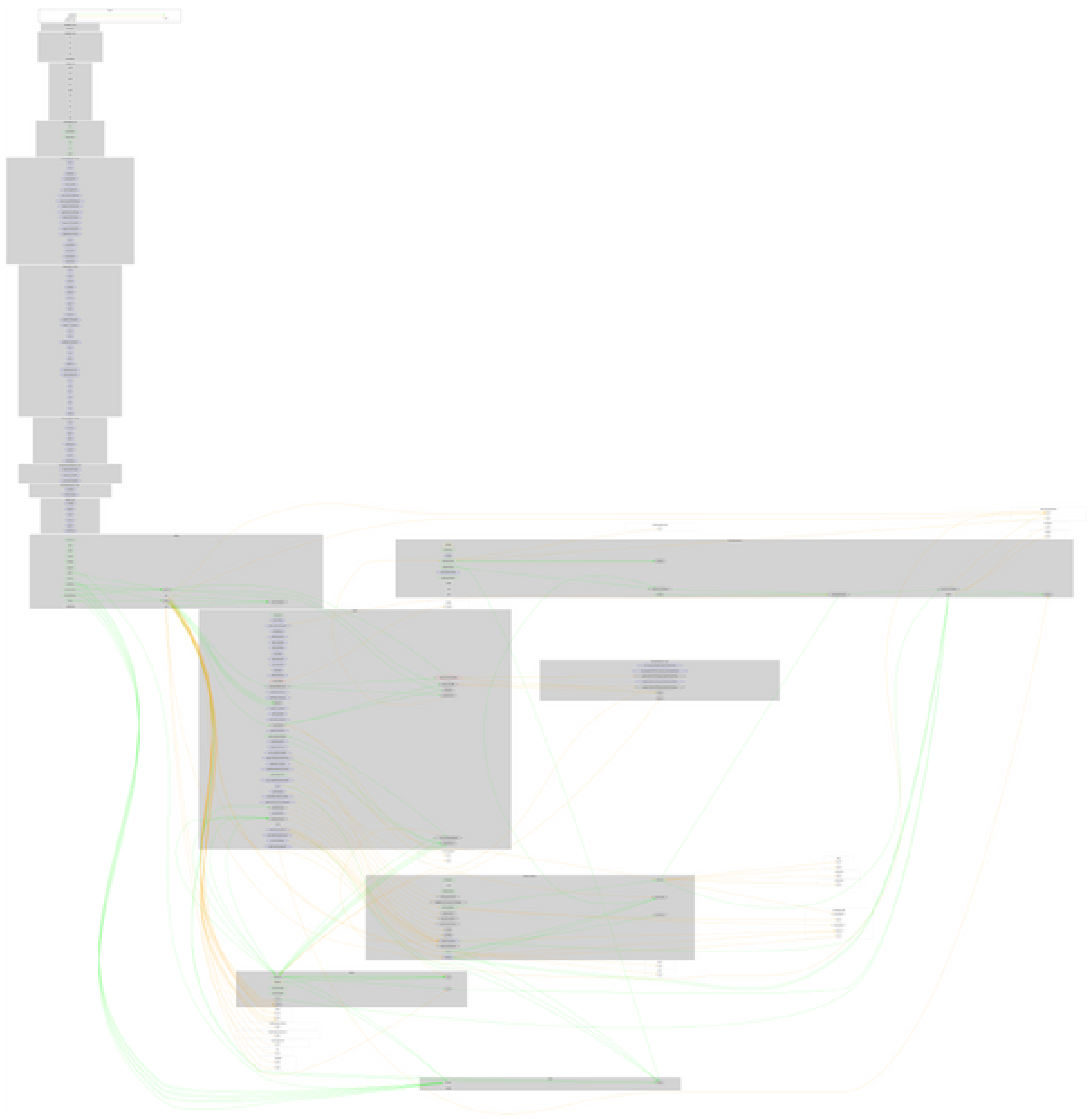
	TransferTokens			
	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External	✓	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
IterableMapping	Library			
	get	Public		-
	getIndexOfKey	Public		-
	getKeyAtIndex	Public		-
	size	Public		-
	set	Public	✓	-
	remove	Public	✓	-
SafeMath	Library			
	tryAdd	Internal		
	trySub	Internal		
	tryMul	Internal		
	tryDiv	Internal		
	tryMod	Internal		
	add	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	mod	Internal		
	sub	Internal		
	div	Internal		
	mod	Internal		
SafeMathInt	Library			
	mul	Internal		
	div	Internal		

	sub	Internal		
	add	Internal		
	toUint256Safe	Internal		
SafeMathUint	Library			
	toInt256Safe	Internal		
BUSDDividend Tracker	Implementation	DividendPay ingToken		
	<Constructor>	Public	✓	DividendPayin gToken
	_transfer	Internal		
	withdrawDividend	Public		-
	setDividendTokenAddress	External	✓	onlyOwner
	updateMinimumTokenBalanceForDivi dends	External	✓	onlyOwner
	excludeFromDividends	External	✓	onlyOwner
	updateClaimWait	External	✓	onlyOwner
	getLastProcessedIndex	External		-
	getNumberOfTokenHolders	External		-
	getAccount	Public		-
	getAccountAtIndex	Public		-
	canAutoClaim	Private		
	setBalance	External	✓	onlyOwner
	process	Public	✓	-
	processAccount	Public	✓	onlyOwner
deltaace	Implementation	ERC20, Ownable		
	<Constructor>	Public	✓	ERC20
	<Receive Ether>	External	Payable	-
	prepareForPartherOrExchangeListing	External	✓	onlyOwner
	setWalletBalance	External	✓	onlyOwner
	setMaxBuyTransaction	External	✓	onlyOwner
	setMarketingFeeBuy	External	✓	onlyOwner
	setBuybackFeeBuy	External	✓	onlyOwner
	setLpFeeBuy	External	✓	onlyOwner

	setMarketingFeeSell	External	✓	onlyOwner
	setBuybackFeeSell	External	✓	onlyOwner
	setLpFeeSell	External	✓	onlyOwner
	setMaxSellTransaction	External	✓	onlyOwner
	updateBusdDividendToken	External	✓	onlyOwner
	updateMarketingWallet	External	✓	onlyOwner
	setSwapTokensAtAmount	External	✓	onlyOwner
	setSellTransactionMultiplier	External	✓	onlyOwner
	setAuthOnDividends	Public	✓	onlyOwner
	setBusdDividendEnabled	External	✓	onlyOwner
	setMarketingEnabled	External	✓	onlyOwner
	setSwapAndLiquifyEnabled	External	✓	onlyOwner
	updatebusdDividendTracker	External	✓	onlyOwner
	updateUniswapV2Router	External	✓	onlyOwner
	excludeFromFees	Public	✓	onlyOwner
	excludeFromDividend	Public	✓	onlyOwner
	setAutomatedMarketMakerPair	Public	✓	onlyOwner
	_setAutomatedMarketMakerPair	Private	✓	onlyOwner
	updateGasForProcessing	External	✓	onlyOwner
	updateMinimumBalanceForDividends	External	✓	onlyOwner
	updateClaimWait	External	✓	onlyOwner
	getBusdClaimWait	External		-
	getTotalBusdDividendsDistributed	External		-
	getIsExcludedFromFees	Public		-
	withdrawableBusdDividendOf	External		-
	busdDividendTokenBalanceOf	External		-
	getAccountBusdDividendsInfo	External		-
	getAccountBusdDividendsInfoAtIndex	External		-
	processDividendTracker	External	✓	onlyOwner
	claim	External	✓	-
	getLastBusdDividendProcessedIndex	External		-
	getNumberOfBusdDividendTokenHolders	External		-
	_transfer	Internal	✓	
	swapAndLiquify	Private	✓	
	addLiquidity	Private	✓	

	swapTokensForBNB	Private	✓	
	swapTokensForDividendToken	Private	✓	
	swapAndSendBusdDividends	Private	✓	
	transferToWallet	Private	✓	
	transferDividends	Private	✓	

Contract Flow



Domain Info

Domain Name	deltaace.org
Registry Domain ID	D402200000019145028-LROR
Creation Date	2022-02-21T16:41:21Z
Updated Date	2022-03-15T21:14:44Z
Registry Expiry Date	2023-02-21T16:41:21Z
Registrar WHOIS Server	whois.godaddy.com
Registrar URL	http://www.whois.godaddy.com
Registrar	GoDaddy.com, LLC
Registrar IANA ID	146

The domain has been created about 1 month before the creation of the audit. It will expire in 11 months.

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

Summary

Delta Ace Token is an interesting Project that has a friendly and growing community. There are some functions that can be abused by the owner, like manipulating fees up to 100% and stopping transactions for everyone except the user. The contract can be converted into a honeypot and prevent users from Selling if the admin functions are abused. 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.

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

About Cyberscope

Coinscope audit and K.Y.C. service has been rebranded to Cyberscope.

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.

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

The Cyberscope team

https://www.cyberscope.io