Cyberscope

Audit Report

Trendify

December 2024

Network	BASE
Address	0xC515857199b3fb8C2A0f363E82954DB4a658850B
Audited by	© cyberscope

Analysis

● Critical ● Medium ● Minor / Informative ● Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Unresolved
●	OTUT	Transfers User's Tokens	Passed
●	ELFM	Exceeds Fees Limit	Passed
●	MT	Mints Tokens	Unresolved
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Unresolved

Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	OCTD	Transfers Contract's Tokens	Unresolved
●	DDP	Decimal Division Precision	Unresolved
●	IDI	Immutable Declaration Improvement	Unresolved
●	PLPI	Potential Liquidity Provision Inadequacy	Unresolved
●	RRA	Redundant Repeated Approvals	Unresolved
●	RSW	Redundant Storage Writes	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L13	Divide before Multiply Operation	Unresolved
●	L19	Stable Compiler Version	Unresolved

Table of Contents

Risk Classification

The criticality of findings in Cyberscope's smart contract audits is determined by evaluating multiple variables. The two primary variables are:

1. **Likelihood of Exploitation**: This considers how easily an attack can be executed, including the economic feasibility for an attacker.
2. **Impact of Exploitation**: This assesses the potential consequences of an attack, particularly in terms of the loss of funds or disruption to the contract's functionality.

Based on these variables, findings are categorized into the following severity levels:

1. **Critical**: Indicates a vulnerability that is both highly likely to be exploited and can result in significant fund loss or severe disruption. Immediate action is required to address these issues.
2. **Medium**: Refers to vulnerabilities that are either less likely to be exploited or would have a moderate impact if exploited. These issues should be addressed in due course to ensure overall contract security.
3. **Minor**: Involves vulnerabilities that are unlikely to be exploited and would have a minor impact. These findings should still be considered for resolution to maintain best practices in security.
4. **Informative**: Points out potential improvements or informational notes that do not pose an immediate risk. Addressing these can enhance the overall quality and robustness of the contract.

Severity	Likelihood / Impact of Exploitation
● Critical	Highly Likely / High Impact
● Medium	Less Likely / High Impact or Highly Likely/ Lower Impact
● Minor / Informative	Unlikely / Low to no Impact

Review

Contract Name	Delta
Compiler Version	v0.8.25+commit.b61c2a91
Optimization	200 runs
Explorer	https://basescan.org/address/0xc515857199b3fb8c2a0f363e82954db4a658850b
Address	0xc515857199b3fb8c2a0f363e82954db4a658850b
Network	BASE
Symbol	Delta
Decimals	18
Total Supply	10,000,000
Badge Eligibility	Must Fix Criticals

Audit Updates

Initial Audit	25 Dec 2024

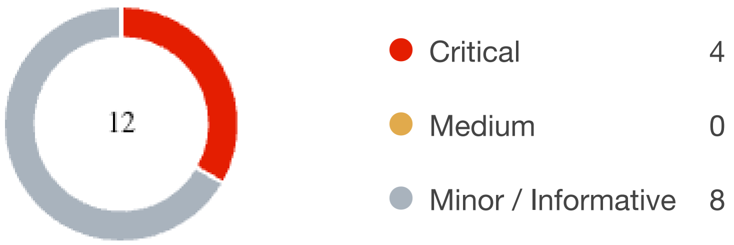
Source Files

Filename	SHA256
draft-IERC6093.sol	4aea87243e6de38804bf8737bf86f750443d3b5e63dd0fd0b7ad92f77cdbd3e3
Strings.sol	bb29970de17b591d2a1061a3458ce3c76618507c495aa1ecf2b3916c5cdcc2a0
StorageSlot.sol	b4a5fb7ab93bfeda06509eafbd5f71fde0e0de84b6d9129553bd535a42166c15

SignedMath.sol	768c28e3a33c3312e57ae8a1caaec2893bc89ac6e386621de018f85e9a2d6e99
ShortStrings.sol	ddd52921d2996abf2e3d9c1c4f6d00194a3e3b278a164948f995862371444a55
SafeERC20Remastered.sol	f12be6e1cb3808cd7c48764f6684435b97a62a4e5ab500c528bd2e8fc9af0be3
Pausable.sol	741d96df6cf0a26bb7f2cf0ad930e30bef0963dec6aa0e0d78d87938a1af7936
Ownable2Step.sol	90f1f1cdd07ce4b90e987065e82899fdaa6ef967d1996915143c6e39818e160c
Ownable.sol	5557ddc8af9d76ecc291d9fe0ad6175e09483a990726ee41697ca2a854fb3af4
Nonces.sol	9b4cbb85d1f5053c744e83302538eb643a713ffd14bc37665b224f1c66529339
Mintable.sol	e42325e08fa3aebd9cf5d6a8ae90a6a51833ecf1333e3ff30d54f44802600453
MessageHashUtils.sol	3c0764f49ebb9cca770c701a949c47f3f90271c4a2096a0bdd1e28e5dff0fa8c
Math.sol	a6ee779fc42e6bf01b5e6a963065706e882b016affbedfd8be19a71ea48e6e15
Initializable.sol	d8bdbc02610707a06c0f9ab4574061df7acd8efcd398ca7d372808c4f4a0f391
IUniswapV2Router02.sol	a2900701961cb0b6152fc073856b972564f7c798797a4a044e83d2ab8f0e8d38
IUniswapV2Router01.sol	0439ffe0fd4a5e1f4e22d71ddbda76d63d61679947d158cba4ee0a1da60cf663
IUniswapV2Pair.sol	29c75e69ce173ff8b498584700fef76bc81498c1d98120e2877a1439f0c31b5a

IUniswapV2Factory.sol	51d056199e3f5e41cb1a9f11ce581aa3e190cc982db5771ffeef8d8d1f962a0d
IERC5267.sol	efd1ebd1e04b6ef9c3b8781a097588f83da954323f438d54a71dc06508e6c7b8
IERC20Permit.sol	912509e0e9bf74e0f8a8c92d031b5b26d2d35c6d4abf3f56251be1ea9ca946bf
IERC20Metadata.sol	c54116cde6fb2353a298201a9fd85375daa7a9d34bb33433e52d72d0c833ab3d
IERC20.sol	6f2faae462e286e24e091d7718575179644dc60e79936ef0c92e2d1ab3ca3cee
ERC20Permit.sol	3b62857c4bea11705f55d332efc764f6dbf2c88bb84006a7403a091932384b3d
ERC20Burnable.sol	dcd71cbfb559a195cd0ed395d69e4719349e08da9729555548db39fa57d1701d
ERC20.sol	b9a23053ca7916e0d304b5cc2b049803d02c9f1f79e365b8ccf8cf0498a0d37f
EIP712.sol	9b51c185def17d1c22947f06d480769a52a3bbf89d26ed1d1a79d34df80a11ff
ECDSA.sol	37828cb50b47bcc51c7b770bde15d5885d871ef1e67028057a0b788c3568726e
Delta.sol	38bfd1c4c1352d944b8bdfeb0d7c673bbc6957c24ea83cfb51795dc7779859f8
Context.sol	847fda5460fee70f56f4200f59b82ae622bb03c79c77e67af010e31b7e2cc5b6
Address.sol	b3710b1712637eb8c0df81912da3450da6ff67b0b3ed18146b033ed15b1aa3b9

Findings Breakdown



Severity		Unresolved	Acknowledged	Resolved	Other
●	Critical	4	0	0	0
●	Medium	0	0	0	0
●	Minor / Informative	8	0	0	0

ST - Stops Transactions

Criticality	Critical
Location	Delta.sol#L120,461
Status	Unresolved

Description

The transactions are initially disabled for all users excluding the authorized addresses. The owner can enable the transactions for all users. Once the transactions are enable the owner will not be able to disable them again.

```
if (
    (AMMPairs[from] && !isExcludedFromTradingRestriction[to]) ||
    ( AMMPairs[to] && !isExcludedFromTradingRestriction[from])
)
{
   if (!tradingEnabled) revert TradingNotEnabled();
}
```

The contract owner has the authority to stop the transactions for all users. The owner may take advantage of it by calling the `pause` method.

```
function pause() public onlyOwner {
   _pause();
}
```

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.

Temporary Solutions:

These measurements do not decrease the severity of the finding

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-signature wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

Permanent Solution:

- Renouncing the ownership, which will eliminate the threats but it is non-reversible.

MT - Mints Tokens

Criticality	Critical
Location	Mintable.sol#L18
Status	Unresolved

Description

The contract owner has the authority to mint tokens. The owner may take advantage of it by calling the `mint` function. As a result, the contract tokens will be highly inflated.

```
function mint(address to, uint256 amount) public onlyOwner {
    if (totalSupply() + amount > maxSupply) revert
MintCannotExceedMaxSupply();

    _mint(to, amount);
}
```

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.

Temporary Solutions:

These measurements do not decrease the severity of the finding

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-signature wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

Permanent Solution:

- Renouncing the ownership, which will eliminate the threats but it is non-reversible.

BC - Blacklists Addresses

Criticality	Critical
Location	Delta.sol#L138
Status	Unresolved

Description

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

```
function blacklist(address account, bool isBlacklisted) external onlyOwner
{
    blacklisted[account] = isBlacklisted;
    emit BlacklistUpdated(account, isBlacklisted);
}
```

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.

Temporary Solutions:

These measurements do not decrease the severity of the finding

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-signature wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

Permanent Solution:

- Renouncing the ownership, which will eliminate the threats but it is non-reversible.

OCTD - Transfers Contract's Tokens

Criticality	Critical
Location	Delta.sol#L128
Status	Unresolved

Description

The contract owner has the authority to claim all the balance of the contract. The owner may take advantage of it by calling the `recoverToken` function.

The `_taxwalletPending` and `_liquidityPending` variables accumulate tokens that are meant to be swapped. If the owner calls the `recoverToken` method, then the contract will transfer all of its tokens to the owner's address. The contract's balance will become zero, but the contract does not reset the `_taxwalletPending` and `_liquidityPending` variables back to zero. As a result, the next transaction that triggers the swap functionality will revert since these variables will have an amount greater than the contract's balance actual amount.

```
function recoverToken(uint256 amount) external onlyOwner {
   uint256 maxRecoverable = balanceOf(address(this)) - getAllPending();if
(amount > maxRecoverable) revert InvalidAmountToRecover(amount,
maxRecoverable);
   _transfer(address(this), msg.sender, amount);
}
```

Additionally, the contract mistakenly transfers the fee amount to the `taxwalletAddress` instead of the contract's address. Hence, transactions triggering the swap functionality will revert since the contract's balance will be zero.

```
if (fees > 0) {
   super._update(from, address(taxwalletAddress), fees);
}
```

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.

Temporary Solutions:

These measurements do not decrease the severity of the finding

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-signature wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

Permanent Solution:

- Renouncing the ownership, which will eliminate the threats but it is non-reversible.

DDP - Decimal Division Precision

Criticality	Minor / Informative
Location	Delta.sol#L367,370,375
Status	Unresolved

Description

Division of decimal (fixed point) numbers can result in rounding errors due to the way that division is implemented in Solidity. Thus, it may produce issues with precise calculations with decimal numbers.

Solidity represents decimal numbers as integers, with the decimal point implied by the number of decimal places specified in the type (e.g. decimal with 18 decimal places). When a division is performed with decimal numbers, the result is also represented as an integer, with the decimal point implied by the number of decimal places in the type. This can lead to rounding errors, as the result may not be able to be accurately represented as an integer with the specified number of decimal places.

Hence, the splitted shares will not have the exact precision and some funds may not be calculated as expected.

```
_taxwalletPending += fees * taxwalletFees[txType] / totalFees[txType];
autoBurnPortion = fees * autoBurnFees[txType] / totalFees[txType];
_liquidityPending += fees * liquidityFees[txType] / totalFees[txType];
```

Recommendation

The team is advised to take into consideration the rounding results that are produced from the solidity calculations. The contract could calculate the subtraction of the divided funds in the last calculation in order to avoid the division rounding issue.

IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	Mintable.sol#L15
Status	Unresolved

Description

The contract declares state variables that their value is initialized once in the constructor and are not modified afterwards. The `immutable` is a special declaration for this kind of state variables that saves gas when it is defined.

```
maxSupply
```

Recommendation

By declaring a variable as immutable, the Solidity compiler is able to make certain optimizations. This can reduce the amount of storage and computation required by the contract, and make it more gas-efficient.

PLPI - Potential Liquidity Provision Inadequacy

Criticality	Minor / Informative
Location	Delta.sol#L143
Status	Unresolved

Description

The contract operates under the assumption that liquidity is consistently provided to the pair between the contract's token and the native currency. However, there is a possibility that liquidity is provided to a different pair. This inadequacy in liquidity provision in the main pair could expose the contract to risks. Specifically, during eligible transactions, where the contract attempts to swap tokens with the main pair, a failure may occur if liquidity has been added to a pair other than the primary one. Consequently, transactions triggering the swap functionality will result in a revert.

```
function _swapTokensForCoin(uint256 tokenAmount) private {
    address[] memory path = new address[](2);
    path[0] = address(this);
    path[1] = routerV2.WETH();
    _approve(address(this), address(routerV2), tokenAmount);
    routerV2.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount, 0,
path, address(this), block.timestamp);
}
```

Recommendation

The team is advised to implement a runtime mechanism to check if the pair has adequate liquidity provisions. This feature allows the contract to omit token swaps if the pair does not have adequate liquidity provisions, significantly minimizing the risk of potential failures.

Furthermore, the team could ensure the contract has the capability to switch its active pair in case liquidity is added to another pair.

Additionally, the contract could be designed to tolerate potential reverts from the swap functionality, especially when it is a part of the main transfer flow. This can be achieved by executing the contract's token swaps in a non-reversible manner, thereby ensuring a more resilient and predictable operation.

RRA - Redundant Repeated Approvals

Criticality	Minor / Informative
Location	Delta.sol#L147
Status	Unresolved

Description

The contract is designed to `approve` token transfers during the contract's operation by calling the _approve function before specific operations. This approach results in additional gas costs since the approval process is repeated for every operation execution, leading to inefficiencies and increased transaction expenses.

```
function _swapTokensForCoin(uint256 tokenAmount) private {
    address[] memory path = new address[](2);
    path[0] = address(this);
    path[1] = routerV2.WETH();
    _approve(address(this), address(routerV2), tokenAmount);
    routerV2.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount, 0,
path, address(this), block.timestamp);
}
```

Recommendation

Since the approved address is a trusted third-party source, it is recommended to optimize the contract by approving the maximum amount of tokens once in the initial set of the variable, rather than before each operation. This change will reduce the overall gas consumption and improve the efficiency of the contract.

RSW - Redundant Storage Writes

Criticality	Minor / Informative
Location	Delta.sol#L244,276,336
Status	Unresolved

Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

The contract modifies the state of certain variables without checking if their current state is the same as the provided argument. As a result, redundant storage writes are being performed by the contract.

```
isExcludedFromFees[account] = isExcluded;
isExcludedFromLimits[account] = isExcluded;
isExcludedFromTradingRestriction[account] = isExcluded;
```

Recommendation

The team is advised to take these segments into consideration and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	Delta.sol#L36,151,167,179,180,181,197,234,289,303,309,315,321
Status	Unresolved

Description

The Solidity style guide is a set of guidelines for writing clean and consistent Solidity code. Adhering to a style guide can help improve the readability and maintainability of the Solidity code, making it easier for others to understand and work with.

The followings are a few key points from the Solidity style guide:

1. Use camelCase for function and variable names, with the first letter in lowercase (e.g., myVariable, updateCounter).
2. Use PascalCase for contract, struct, and enum names, with the first letter in uppercase (e.g., MyContract, UserStruct, ErrorEnum).
3. Use uppercase for constant variables and enums (e.g., MAX_VALUE, ERROR_CODE).
4. Use indentation to improve readability and structure.
5. Use spaces between operators and after commas.
6. Use comments to explain the purpose and behavior of the code.
7. Keep lines short (around 120 characters) to improve readability.

```
mapping (address => bool) public AMMPairs
uint16 _swapThresholdRatio
address _newAddress
uint16 _buyFee
uint16 _sellFee
uint16 _transferFee
uint256 _maxWalletAmount
uint256 _maxBuyAmount
uint256 _maxSellAmount
uint256 _maxTransferAmount
uint256 _tradeCooldownTime
```

Recommendation

By following the Solidity naming convention guidelines, the codebase increased the readability, maintainability, and makes it easier to work with.
Find more information on the Solidity documentation
https://docs.soliditylang.org/en/stable/style-guide.html#naming-conventions.

L13 - Divide before Multiply Operation

Criticality	Minor / Informative
Location	Delta.sol#L364,367,370,375
Status	Unresolved

Description

It is important to be aware of the order of operations when performing arithmetic calculations. This is especially important when working with large numbers, as the order of operations can affect the final result of the calculation. Performing divisions before multiplications may cause loss of prediction.

```
fees = amount * totalFees[txType] / 10000
autoBurnPortion = fees * autoBurnFees[txType] / totalFees[txType]
```

Recommendation

To avoid this issue, it is recommended to carefully consider the order of operations when performing arithmetic calculations in Solidity. It's generally a good idea to use parentheses to specify the order of operations. The basic rule is that the multiplications should be prior to the divisions.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	Mintable.sol#L3
Status	Unresolved

Description

The `^` symbol indicates that any version of Solidity that is compatible with the specified version (i.e., any version that is a higher minor or patch version) can be used to compile the contract. The version lock is a mechanism that allows the author to specify a minimum version of the Solidity compiler that must be used to compile the contract code. This is useful because it ensures that the contract will be compiled using a version of the compiler that is known to be compatible with the code.

```
pragma solidity ^0.8.19;
```

Recommendation

The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest version that provides all the required functionality for the codebase. As a result, the project will be compiled in a well-tested LTS (Long Term Support) environment.

Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
IERC20Errors	Interface			
IERC721Errors	Interface			
IERC1155Errors	Interface			
Strings	Library			
	toString	Internal		
	toStringSigned	Internal		
	toHexString	Internal		
	toHexString	Internal		
	toHexString	Internal		
	equal	Internal		
StorageSlot	Library			
	getAddressSlot	Internal		
	getBooleanSlot	Internal		
	getBytes32Slot	Internal		
	getUint256Slot	Internal		
	getStringSlot	Internal		

	getStringSlot	Internal		
	getBytesSlot	Internal		
	getBytesSlot	Internal		
SignedMath	Library			
	max	Internal		
	min	Internal		
	average	Internal		
	abs	Internal		
ShortStrings	Library			
	toShortString	Internal		
	toString	Internal		
	byteLength	Internal		
	toShortStringWithFallback	Internal	✓	
	toStringWithFallback	Internal		
	byteLengthWithFallback	Internal		
SafeERC20Remastered	Library			
	safeTransfer	Internal	✓	
	safeTransfer_noRevert	Internal	✓	
	safeTransferFrom	Internal	✓	
	safeIncreaseAllowance	Internal	✓	
	forceApprove	Internal	✓	

	_callOptionalReturn	Private	✓	
	_callOptionalReturnBool	Private	✓	
Pausable	Implementation	Context		
		Public	✓	-
	paused	Public		-
	_requireNotPaused	Internal		
	_requirePaused	Internal		
	_pause	Internal	✓	whenNotPaused
	_unpause	Internal	✓	whenPaused
Ownable2Step	Implementation	Ownable		
	pendingOwner	Public		-
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
	acceptOwnership	Public	✓	-
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	_checkOwner	Internal		
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	

Nonces	Implementation			
	nonces	Public		-
	_useNonce	Internal	✓	
	_useCheckedNonce	Internal	✓	
Mintable	Implementation	ERC20, Ownable2Step		
		Public	✓	-
	mint	Public	✓	onlyOwner
MessageHashUtils	Library			
	toEthSignedMessageHash	Internal		
	toEthSignedMessageHash	Internal		
	toDataWithIntendedValidatorHash	Internal		
	toTypedDataHash	Internal		
Math	Library			
	tryAdd	Internal		
	trySub	Internal		
	tryMul	Internal		
	tryDiv	Internal		
	tryMod	Internal		
	max	Internal		
	min	Internal		

	average	Internal		
	ceilDiv	Internal		
	mulDiv	Internal		
	mulDiv	Internal		
	sqrt	Internal		
	sqrt	Internal		
	log2	Internal		
	log2	Internal		
	log10	Internal		
	log10	Internal		
	log256	Internal		
	log256	Internal		
	unsignedRoundsUp	Internal		
Initializable	Implementation			
IUniswapV2Router02	Interface	IUniswapV2Router01		
	removeLiquidityETHSupportingFeeOnTransferTokens	External	✓	-
	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External	✓	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	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		-
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	✓	-

IUniswapV2Factory	Interface			
	feeTo	External		-
	feeToSetter	External		-
	getPair	External		-
	allPairs	External		-
	allPairsLength	External		-
	createPair	External	✓	-
	setFeeTo	External	✓	-
	setFeeToSetter	External	✓	-
IERC5267	Interface			
	eip712Domain	External		-
IERC20Permit	Interface			
	permit	External	✓	-
	nonces	External		-
	DOMAIN_SEPARATOR	External		-
IERC20Metadata	Interface	IERC20		
	name	External		-
	symbol	External		-
	decimals	External		-

IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
ERC20Permit	Implementation	ERC20, IERC20Permit, EIP712, Nonces		
		Public	✓	EIP712
	permit	Public	✓	-
	nonces	Public		-
	DOMAIN_SEPARATOR	External		-
ERC20Burnable	Implementation	Context, ERC20		
	burn	Public	✓	-
	burnFrom	Public	✓	-
ERC20	Implementation	Context, IERC20, IERC20Metadata, IERC20Errors		
		Public	✓	-
	name	Public		-

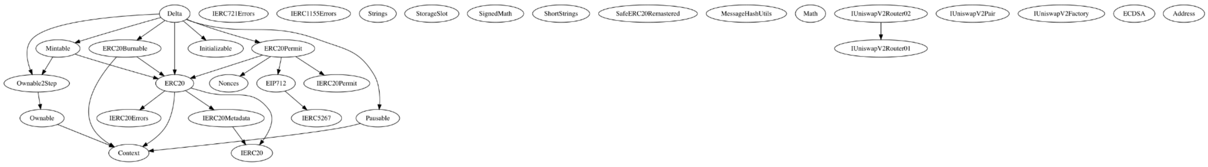
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	_transfer	Internal	✓	
	_update	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_approve	Internal	✓	
	_approve	Internal	✓	
	_spendAllowance	Internal	✓	
EIP712	Implementation	IERC5267		
		Public	✓	-
	_domainSeparatorV4	Internal		
	_buildDomainSeparator	Private		
	_hashTypedDataV4	Internal		
	eip712Domain	Public		-
	_EIP712Name	Internal		
	_EIP712Version	Internal		

ECDSA	Library			
	tryRecover	Internal		
	recover	Internal		
	tryRecover	Internal		
	recover	Internal		
	tryRecover	Internal		
	recover	Internal		
	_throwError	Private		
Delta	Implementation	ERC20, Ownable2Step, ERC20Burnable, ERC20Permit, Mintable, Pausable, Initializable		
		Public	✓	ERC20 Ownable Mintable ERC20Permit
	decimals	Public		-
	pause	Public	✓	onlyOwner
	unpause	Public	✓	onlyOwner
	recoverToken	External	✓	onlyOwner
	recoverForeignERC20	External	✓	onlyOwner
	blacklist	External	✓	onlyOwner
	_swapTokensForCoin	Private	✓	
	updateSwapThreshold	Public	✓	onlyOwner
	getSwapThresholdAmount	Public		-
	getAllPending	Public		-

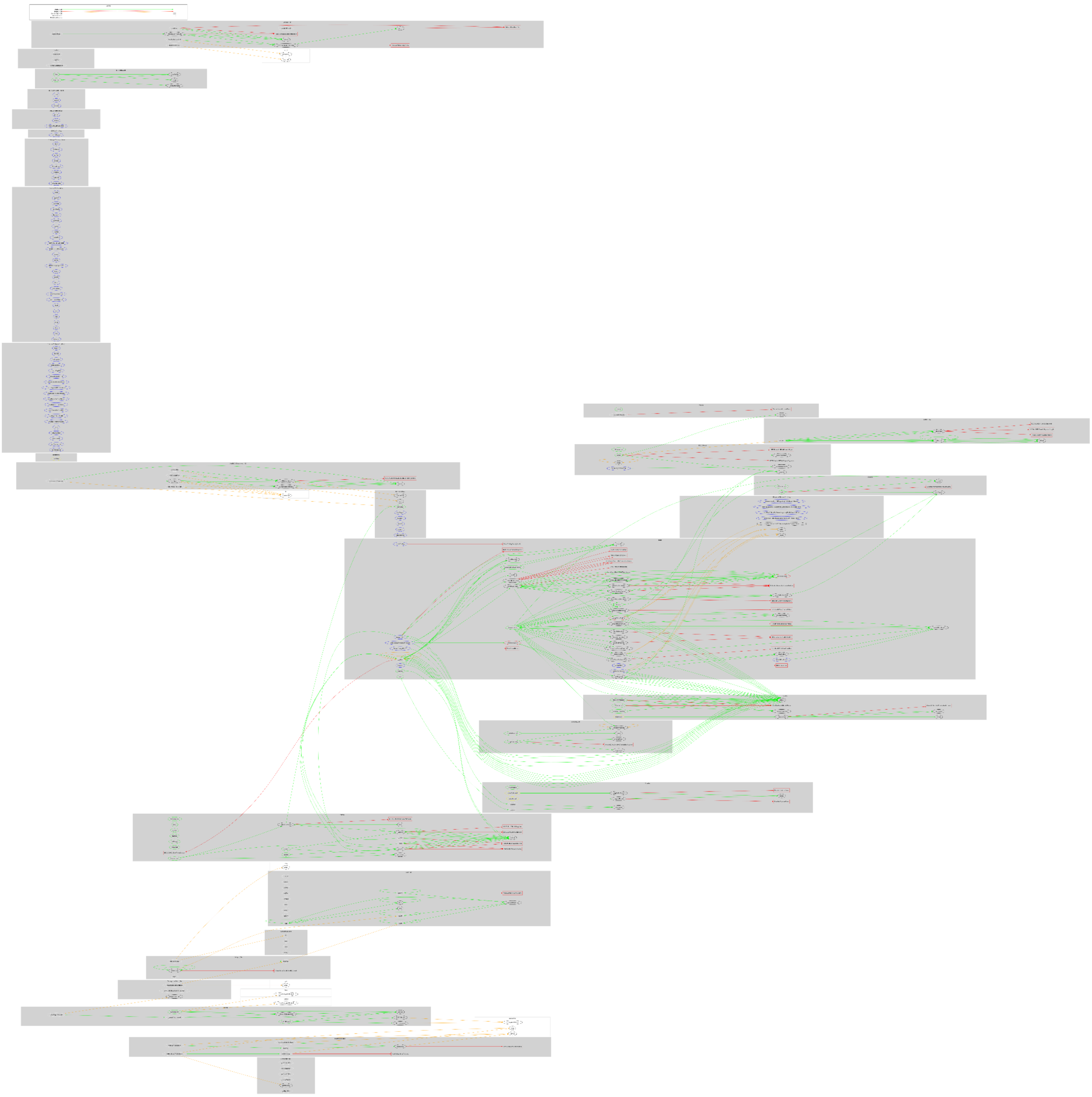
	taxwalletAddressSetup	Public	✓	onlyOwner
	taxwalletFeesSetup	Public	✓	onlyOwner
	autoBurnFeesSetup	Public	✓	onlyOwner
	_swapAndLiquify	Private	✓	
	_addLiquidity	Private	✓	
	addLiquidityFromLeftoverTokens	External	✓	-
	liquidityFeesSetup	Public	✓	onlyOwner
	excludeFromFees	Public	✓	onlyOwner
	_updateRouterV2	Private	✓	
	setAMMPair	External	✓	onlyOwner
	_setAMMPair	Private	✓	
	excludeFromLimits	External	✓	onlyOwner
	_excludeFromLimits	Internal	✓	
	setPairForTaxation	External	✓	onlyOwner
	updateMaxWalletAmount	Public	✓	onlyOwner
	_maxWalletSafeLimit	Private		
	_maxTxSafeLimit	Private		
	updateMaxBuyAmount	Public	✓	onlyOwner
	updateMaxSellAmount	Public	✓	onlyOwner
	updateMaxTransferAmount	Public	✓	onlyOwner
	updateTradeCooldownTime	Public	✓	onlyOwner
	enableTrading	External	✓	onlyOwner
	excludeFromTradingRestriction	Public	✓	onlyOwner
	_update	Internal	✓	

	_beforeTokenUpdate	Internal		whenNotPaused
	_afterTokenUpdate	Internal	✓	
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
	_contextSuffixLength	Internal		
Address	Library			
	sendValue	Internal	✓	
	functionCall	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionStaticCall	Internal		
	functionDelegateCall	Internal	✓	
	verifyCallResultFromTarget	Internal		
	verifyCallResult	Internal		
	_revert	Private		

Inheritance Graph



Flow Graph



Summary

Trendify Token contract implements a token mechanism. This audit investigates security issues, business logic concerns, and potential improvements. There are some functions that can be abused by the owner like stop transactions, mint tokens, and massively blacklist addresses. If the contract owner abuses the mint functionality, then the contract will be highly inflated. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats. There is also a limit of max 25% fees.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope's prior written consent. This report is not nor should be considered an "endorsement" or "disapproval" of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security Cyberscope's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.

The Cyberscope team

cyberscope.io