



Cyberscope

Audit Report

ZyptoToken

May 2024

Network ETH

Address 0x7A65CB87F596Caf31a4932f074c59c0592bE77D7

Audited by © cyberscope

Analysis

● Critical ● Medium ● Minor / Informative ● Pass

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

Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	MTV	Misleading Tax Variable	Unresolved
●	MEE	Missing Events Emission	Unresolved
●	PLPI	Potential Liquidity Provision Inadequacy	Unresolved
●	PTRP	Potential Transfer Revert Propagation	Unresolved
●	PVC	Price Volatility Concern	Unresolved
●	RSW	Redundant Storage Writes	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L09	Dead Code Elimination	Unresolved
●	L16	Validate Variable Setters	Unresolved
●	L17	Usage of Solidity Assembly	Unresolved
●	L18	Multiple Pragma Directives	Unresolved
●	L19	Stable Compiler Version	Unresolved

Table of Contents

Analysis	1
Diagnostics	2
Table of Contents	3
Review	5
Audit Updates	5
Source Files	5
Findings Breakdown	6
MTV - Misleading Tax Variable	7
Description	7
Recommendation	7
MEE - Missing Events Emission	8
Description	8
Recommendation	8
PLPI - Potential Liquidity Provision Inadequacy	9
Description	9
Recommendation	10
PTRP - Potential Transfer Revert Propagation	11
Description	11
Recommendation	11
PVC - Price Volatility Concern	12
Description	12
Recommendation	12
RSW - Redundant Storage Writes	13
Description	13
Recommendation	13
L04 - Conformance to Solidity Naming Conventions	14
Description	14
Recommendation	15
L07 - Missing Events Arithmetic	16
Description	16
Recommendation	16
L09 - Dead Code Elimination	17
Description	17
Recommendation	17
L16 - Validate Variable Setters	18
Description	18
Recommendation	18
L17 - Usage of Solidity Assembly	19
Description	19

Recommendation	19
L18 - Multiple Pragma Directives	20
Description	20
Recommendation	20
L19 - Stable Compiler Version	21
Description	21
Recommendation	21
Functions Analysis	22
Inheritance Graph	29
Flow Graph	30
Summary	31
Disclaimer	32
About Cyberscope	33

Review

Contract Name	ZyptoToken
Compiler Version	v0.8.24+commit.e11b9ed9
Optimization	200 runs
Explorer	https://etherscan.io/address/0x7A65CB87F596Caf31a4932f074c59c0592bE77D7
Address	0x7A65CB87F596Caf31a4932f074c59c0592bE77D7
Network	ETH
Symbol	ZYPTO
Decimals	18
Total Supply	1,000,000,000

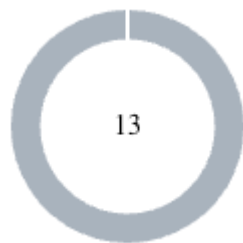
Audit Updates

Initial Audit	07 May 2024
---------------	-------------

Source Files

Filename	SHA256
ZyptoToken.sol	9e180a1e1551cf21f2bc7c639d849e3c17e8d768ae3aaf67e47fde544d69f7ac

Findings Breakdown



● Critical	0
● Medium	0
● Minor / Informative	13

Severity	Unresolved	Acknowledged	Resolved	Other
● Critical	0	0	0	0
● Medium	0	0	0	0
● Minor / Informative	13	0	0	0

MTV - Misleading Tax Variable

Criticality	Minor / Informative
Location	ZyptoToken.sol#L1510
Status	Unresolved

Description

The contract utilizes a specific tax variable, suggesting that this tax applies solely to the transactions. Contrarily, the implemented logic indiscriminately applies this tax across different transfer types. This misuse of the variable name leads to a misunderstanding of its application, which could result in unexpected tax implications for all transaction types, contrary to the intended exclusive focus based on the variable naming.

```
function getFeesForTrade(address from) internal view returns (uint256
liquidityFee, uint256 marketingFee, uint256 totalFee) {
    if (_isAutomatedMarketMakerPair[from]) {
        liquidityFee = _buyFeeLiquidity;
        marketingFee = _buyFeeMarketing;
        totalFee = _buyFeeTotal;
    } else {
        liquidityFee = _sellFeeLiquidity;
        marketingFee = _sellFeeMarketing;
        totalFee = _sellFeeTotal;
    }
}
```

Recommendation

To ensure the contract operates transparently and aligns with user expectations, it is essential to adjust the tax application logic to correspond with each specific type of transfer. The tax variable naming and functionality should accurately reflect the type of transaction it is intended for. If the tax rate is to be applied universally to all transfers, adopting a more general tax rate variable would be appropriate. This change will ensure that tax implications are clear and predictable, preventing any unintended consequences and enhancing the overall reliability of the contract's financial transactions.

MEE - Missing Events Emission

Criticality	Minor / Informative
Location	ZyptoToken.sol#L1387,1423,1429
Status	Unresolved

Description

The contract performs actions and state mutations from external methods that do not result in the emission of events. Emitting events for significant actions is important as it allows external parties, such as wallets or dApps, to track and monitor the activity on the contract. Without these events, it may be difficult for external parties to accurately determine the current state of the contract.

```
_isOpenForTrading = true;  
_marketingWallet = payable(account);  
_isAutomatedMarketMakerPair[pair] = value;
```

Recommendation

It is recommended to include events in the code that are triggered each time a significant action is taking place within the contract. These events should include relevant details such as the user's address and the nature of the action taken. By doing so, the contract will be more transparent and easily auditable by external parties. It will also help prevent potential issues or disputes that may arise in the future.

PLPI - Potential Liquidity Provision Inadequacy

Criticality	Minor / Informative
Location	ZyptoToken.sol#L1453
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.

```
_uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(  
    tokenAmount,  
    0, // accept any amount of ETH  
    path,  
    to,  
    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.

PTRP - Potential Transfer Revert Propagation

Criticality	Minor / Informative
Location	ZyptoToken.sol#L1463
Status	Unresolved

Description

The contract sends funds to a `marketingWallet` as part of the transfer flow. This address can either be a wallet address or a contract. If the address belongs to a contract then it may revert from incoming payment. As a result, the error will propagate to the token's contract and revert the transfer.

```
function swapAndSendETHToMarketing(uint256 tokenAmount) private {  
    swapTokensForETH(tokenAmount, _marketingWallet);  
}
```

Recommendation

The contract should tolerate the potential revert from the underlying contracts when the interaction is part of the main transfer flow. This could be achieved by not allowing set contract addresses or by sending the funds in a non-revertable way.

PVC - Price Volatility Concern

Criticality	Minor / Informative
Location	ZyptoToken.sol#L1389,1432
Status	Unresolved

Description

The contract accumulates tokens from the taxes to swap them for ETH. The variable `swapTokensAtAmount` sets a threshold where the contract will trigger the swap functionality. Additionally, the contract can continuously accumulate tokens from the taxes to swap them for ETH if the `_swapAndLiquifyEnabled` variable is set to false. If the contract owner invokes the `swapAndLiquifyNow` function then the contract will trigger the swap functionality. If the variable is set to a big number, then the contract will swap a huge amount of tokens for ETH.

It is important to note that the price of the token representing it, can be highly volatile. This means that the value of a price volatility swap involving Ether could fluctuate significantly at the triggered point, potentially leading to significant price volatility for the parties involved.

```
function setSwapAmount(uint256 swapTokensAtAmount) public onlyOwner {
    _swapTokensAtAmount = swapTokensAtAmount;
}
function swapAndLiquifyNow() public {
    uint256 contractTokenBalance = balanceOf(address(this));
    if (contractTokenBalance > 0) {
        swapAndLiquify(contractTokenBalance);
    }
}
```

Recommendation

The contract could ensure that it will not sell more than a reasonable amount of tokens in a single transaction. A suggested implementation could check that the maximum amount should be less than a fixed percentage of the exchange reserves. Hence, the contract will guarantee that it cannot accumulate a huge amount of tokens in order to sell them.

RSW - Redundant Storage Writes

Criticality	Minor / Informative
Location	ZyptoToken.sol#L1393
Status	Unresolved

Description

The contract modifies the state of the following variables without checking if their current value is the same as the one given as an argument. As a result, the contract performs redundant storage writes, when the provided parameter matches the current state of the variables, leading to unnecessary gas consumption and inefficiencies in contract execution.

```
_swapAndLiquifyEnabled = enabled;
```

Recommendation

The team is advised to implement additional checks within to prevent redundant storage writes when the provided argument matches the current state of the variables. By incorporating statements to compare the new values with the existing values before proceeding with any state modification, the contract can avoid unnecessary storage operations, thereby optimizing gas usage.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	ZyptoToken.sol#L812,1107,1124,1282,1292,1317,1318,1319,1323,1324,1325,1326,1327,1328,1329,1330,1331,1332,1333,1337,1338
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.

```
function DOMAIN_SEPARATOR() external view returns (bytes32);
IERC20 _token
function WETH() external pure returns (address);
uint256 _amount
uint256 _basisPoints
...
```

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/v0.8.17/style-guide.html#naming-convention>.

L07 - Missing Events Arithmetic

Criticality	Minor / Informative
Location	ZyptoToken.sol#L1390,1397,1403
Status	Unresolved

Description

Events are a way to record and log information about changes or actions that occur within a contract. They are often used to notify external parties or clients about events that have occurred within the contract, such as the transfer of tokens or the completion of a task.

It's important to carefully design and implement the events in a contract, and to ensure that all required events are included. It's also a good idea to test the contract to ensure that all events are being properly triggered and logged.

```
_swapTokensAtAmount = swapTokensAtAmount;  
_buyFeeLiquidity = liquidityFee;  
_sellFeeLiquidity = liquidityFee;
```

Recommendation

By including all required events in the contract and thoroughly testing the contract's functionality, the contract ensures that it performs as intended and does not have any missing events that could cause issues with its arithmetic.

L09 - Dead Code Elimination

Criticality	Minor / Informative
Location	ZyptoToken.sol#L32,645,911,920,951,1018,1026,1035,1050,1084,1292
Status	Unresolved

Description

In Solidity, dead code is code that is written in the contract, but is never executed or reached during normal contract execution. Dead code can occur for a variety of reasons, such as:

- Conditional statements that are always false.
- Functions that are never called.
- Unreachable code (e.g., code that follows a return statement).

Dead code can make a contract more difficult to understand and maintain, and can also increase the size of the contract and the cost of deploying and interacting with it.

```
function _contextSuffixLength() internal view virtual returns (uint256) {
    return 0;
}

function _burn(address account, uint256 value) internal {
    if (account == address(0)) {
        revert ERC20InvalidSender(address(0));
    }
    _update(account, address(0), value);
}
...
```

Recommendation

To avoid creating dead code, it's important to carefully consider the logic and flow of the contract and to remove any code that is not needed or that is never executed. This can help improve the clarity and efficiency of the contract.

L16 - Validate Variable Setters

Criticality	Minor / Informative
Location	ZyptoToken.sol#L1347,1423
Status	Unresolved

Description

The contract performs operations on variables that have been configured on user-supplied input. These variables are missing of proper check for the case where a value is zero. This can lead to problems when the contract is executed, as certain actions may not be properly handled when the value is zero.

```
_marketingWallet = payable(account);
```

Recommendation

By adding the proper check, the contract will not allow the variables to be configured with zero value. This will ensure that the contract can handle all possible input values and avoid unexpected behavior or errors. Hence, it can help to prevent the contract from being exploited or operating unexpectedly.

L17 - Usage of Solidity Assembly

Criticality	Minor / Informative
Location	ZyptoToken.sol#L967
Status	Unresolved

Description

Using assembly can be useful for optimizing code, but it can also be error-prone. It's important to carefully test and debug assembly code to ensure that it is correct and does not contain any errors.

Some common types of errors that can occur when using assembly in Solidity include Syntax, Type, Out-of-bounds, Stack, and Revert.

```
assembly {  
    let returndata_size := mload(returndata)  
    revert(add(32, returndata), returndata_size)  
}
```

Recommendation

It is recommended to use assembly sparingly and only when necessary, as it can be difficult to read and understand compared to Solidity code.

L18 - Multiple Pragma Directives

Criticality	Minor / Informative
Location	ZyptoToken.sol#L6,1098,1120,1278
Status	Unresolved

Description

If the contract includes multiple conflicting pragma directives, it may produce unexpected errors. To avoid this, it's important to include the correct pragma directive at the top of the contract and to ensure that it is the only pragma directive included in the contract.

```
pragma solidity 0.8.24;  
pragma solidity ^0.8.23;
```

Recommendation

It is important to include only one pragma directive at the top of the contract and to ensure that it accurately reflects the version of Solidity that the contract is written in.

By including all required compiler options and flags in a single pragma directive, the potential conflicts could be avoided and ensure that the contract can be compiled correctly.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	ZyptoToken.sol#L1098,1120,1278
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 ^0.8.23;
```

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
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
	_contextSuffixLength	Internal		
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	_checkOwner	Internal		
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
IERC20Errors	Interface			
IERC721Errors	Interface			
IERC1155Errors	Interface			

IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IERC20Metadata	Interface	IERC20		
	name	External		-
	symbol	External		-
	decimals	External		-
ERC20	Implementation	Context, IERC20, IERC20Meta data, IERC20Error s		
		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	✓	
IERC20Permit	Interface			
	permit	External	✓	-
	nonces	External		-
	DOMAIN_SEPARATOR	External		-
Address	Library			
	sendValue	Internal	✓	
	functionCall	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionStaticCall	Internal		
	functionDelegateCall	Internal	✓	

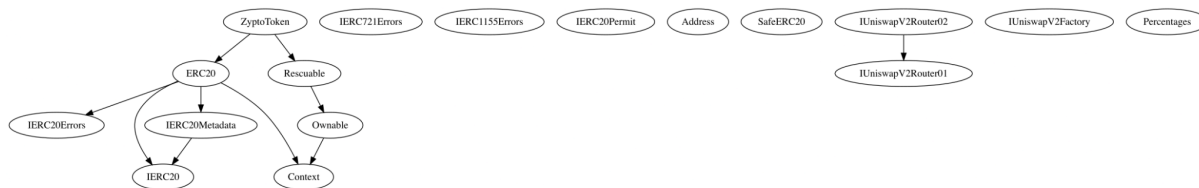
	verifyCallResultFromTarget	Internal		
	verifyCallResult	Internal		
	_revert	Private		
SafeERC20	Library			
	safeTransfer	Internal	✓	
	safeTransferFrom	Internal	✓	
	safeIncreaseAllowance	Internal	✓	
	safeDecreaseAllowance	Internal	✓	
	forceApprove	Internal	✓	
	_callOptionalReturn	Private	✓	
	_callOptionalReturnBool	Private	✓	
Rescuable	Implementation	Ownable		
	rescueERC20	External	✓	onlyOwner
	rescueETH	External	✓	onlyOwner
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	✓	-

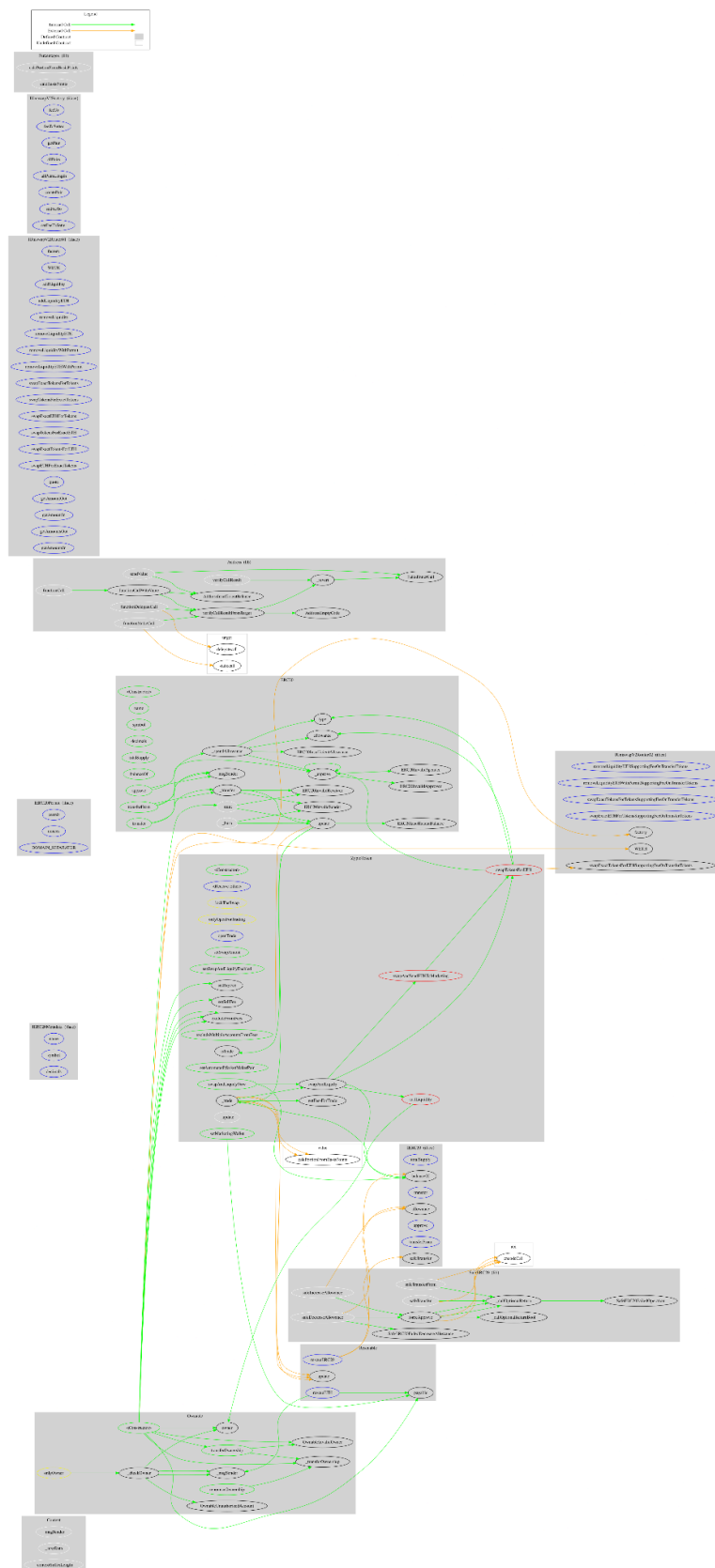
IUniswapV2Factory	Interface			
	feeTo	External		-
	feeToSetter	External		-
	getPair	External		-
	allPairs	External		-
	allPairsLength	External		-
	createPair	External	✓	-
	setFeeTo	External	✓	-
	setFeeToSetter	External	✓	-
Percentages	Library			
	calcPortionFromBasisPoints	Internal		
	calcBasisPoints	Internal		
ZyptoToken	Implementation	ERC20, Rescuable		
		Public	✓	ERC20 Ownable
		External	Payable	-
	openTrade	External	✓	onlyOwner
	setSwapAmount	Public	✓	onlyOwner
	setSwapAndLiquifyEnabled	Public	✓	onlyOwner
	setBuyFee	Public	✓	onlyOwner
	setSellFee	Public	✓	onlyOwner
	excludeFromFees	Public	✓	onlyOwner

	excludeMultipleAccountsFromFees	Public	✓	onlyOwner
	setMarketingWallet	Public	✓	onlyOwner
	setAutomatedMarketMakerPair	Public	✓	onlyOwner
	swapAndLiquifyNow	Public	✓	-
	swapTokensForETH	Private	✓	
	swapAndSendETHToMarketing	Private	✓	
	addLiquidity	Private	✓	
	swapAndLiquify	Private	✓	lockTheSwap
	getFeesForTrade	Internal		
	isTrade	Internal		
	_update	Internal	✓	
	_trade	Internal	✓	onlyOpenForTrading

Inheritance Graph



Flow Graph



Summary

ZyptoToken contract implements a token mechanism. This audit investigates security issues, business logic concerns, and potential improvements. ZyptoToken is an interesting project that has a friendly and growing community. The Smart Contract analysis reported no compiler errors or critical issues. The contract Owner can access some admin functions that can not be used in a malicious way to disturb the users' transactions. There is also a limit of max 4% buy and sell 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

<https://www.cyberscope.io>