Cyberscope

A TAC Security Company

Audit Report

Crypto One

October 2025

Repository https://github.com/Vsc-blockchain/crypto-one-token

Commit 2fca1eb274c671e76b3410d33c604771360c05d8

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
	PVC	Price Volatility Concern	Unresolved
	ROT	Redundant Ownership Transfer	Unresolved
	MEE	Missing Events Emission	Unresolved
	CCR	Contract Centralization Risk	Unresolved
	MC	Missing Check	Unresolved
	PMRM	Potential Mocked Router Manipulation	Unresolved
	IDI	Immutable Declaration Improvement	Unresolved
	L04	Conformance to Solidity Naming Conventions	Unresolved
	L07	Missing Events Arithmetic	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

Repository	https://github.com/Vsc-blockchain/crypto-one-token
Commit	2fca1eb274c671e76b3410d33c604771360c05d8
Badge Eligibility	Yes

Audit Updates

Initial Audit	30 Sep 2025 https://github.com/cyberscope-io/audits/blob/main/0e-one/v1/audit.pdf
Corrected Phase 2	04 Oct 2025 https://github.com/cyberscope-io/audits/blob/main/0e-one/v2/audit.pdf
Corrected Phase 3	06 Oct 2025

Source Files

Filename	SHA256
ERC20.sol	26471cb3b201619290c22693ae6c53e74a8826e01d27157947257aed6ec043b3
CryptoOne.sol	bf1bfc8323d40105c82d1441cb9d6b4a7462ae55f35497172016437e032c45ab

Findings Breakdown

Critical	0
Medium	0
Minor / Informative	9

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	0	0	0	0
Minor / Informative	9	0	0	0

PVC - Price Volatility Concern

Criticality	Minor / Informative
Location	CryptoOne.sol#L99
Status	Unresolved

Description

The contract accumulates tokens from the taxes to swap them for ETH. The variable `swapAtAmount` sets a threshold where 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.

```
Shell
function setSwapAtAmount(uint256 _newSwapAtAmount)
external onlyOwner {
    swapAtAmount = _newSwapAtAmount;

}
```

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.

ROT - Redundant Ownership Transfer

Criticality	Minor / Informative
Location	CryptoOne.sol#L64
Status	Unresolved

Description

The contract's constructor initializes the `Ownable` contract by transferring ownership to the `realOwner` address. However later in the constructor it also calls the `transferOwnership` with `realOwner` as argument. This call is redundant as the ownership is already transferred to the `realOwner` .

```
Shell
constructor(address _router, address
_sellTaxWallet, address _buyTaxWallet, uint256
initialSupply, address realOwner)
Ownable(realOwner) {
        ...
        transferOwnership(realOwner);

}
```

Recommendation

The team is recommended to remove redundancies to enhance code optimization and readability.

MEE - Missing Events Emission

Criticality	Minor / Informative
Location	CryptoOne.sol#L77,84,91,99,152,213
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.

```shell
function setBuyTaxWallet(address payable
newBuyTaxWallet) public onlyOwner
function setSellTaxWallet(address payable
newSellTaxWallet) public onlyOwner
function excludeFromFees(address account, bool
excluded) public onlyOwner
function setSwapAtAmount(uint256 _newSwapAtAmount)
external onlyOwner
function SwapFees() private lockTheSwap
nonReentrant

function manualSwap() external onlyOwner
```

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.

CCR - Contract Centralization Risk

Criticality	Minor / Informative
Location	CryptoOne.sol#L77,84,91,99,213
Status	Unresolved

Description

The contract's functionality and behavior are heavily dependent on external parameters or configurations. While external configuration can offer flexibility, it also poses several centralization risks that warrant attention. Centralization risks arising from the dependence on external configuration include Single Point of Control, Vulnerability to Attacks, Operational Delays, Trust Dependencies, and Decentralization Erosion.

```shell
function setBuyTaxWallet(address payable
newBuyTaxWallet) public onlyOwner
function setSellTaxWallet(address payable
newSellTaxWallet) public onlyOwner
function excludeFromFees(address account, bool
excluded) public onlyOwner
function setSwapAtAmount(uint256 _newSwapAtAmount)
external onlyOwner

function manualSwap() external onlyOwner
```

Recommendation

To address this finding and mitigate centralization risks, it is recommended to evaluate the feasibility of migrating critical configurations and functionality into the contract's codebase itself. This approach would reduce external dependencies and enhance the contract's self-sufficiency. It is essential to carefully weigh the trade-offs between external configuration flexibility and the risks associated with centralization.

MC - Missing Check

Criticality	Minor / Informative
Location	CryptoOne.sol#L91,99
Status	Unresolved

Description

The contract is processing variables that have not been properly sanitized and checked that they form the proper shape. These variables may produce vulnerability issues.

`excludeFromFees` should ensure that `account` is not address(0) and that `CryptoOne` contract cannot be included in fees.

```
Shell
function excludeFromFees(address account, bool
excluded) public onlyOwner {
    _isExcludedFromFees[account] = excluded;

}
```

`setSwapAtAmount` should only accept reasonable `_newSwapAtAmount` as recommended in the `PVC` finding.

```shell
function setSwapAtAmount(uint256 _newSwapAtAmount)
external onlyOwner {
    swapAtAmount = _newSwapAtAmount;

}
```

Recommendation

The team is advised to properly check the variables according to the required specifications.

PMRM - Potential Mocked Router Manipulation

Criticality	Minor / Informative
Location	CryptoOne.sol#L55
Status	Unresolved

Description

The contract includes a method that allows the owner to modify the router address and create a new pair. While this feature provides flexibility, it introduces a security threat. The owner could set the router address to any contract that implements the router's interface, potentially containing malicious code. In the event of a transaction triggering the swap functionality with such a malicious contract as the router, the transaction may be manipulated.

```shell
router = IUniswapV2Router02(_router);
```

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.

IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	CryptoOne.sol#L53,57
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.

```
Shell
tax

swapPair
```

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.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	CryptoOne.sol#L99,152
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.

```
Shell
uint256 _newSwapAtAmount

...
```

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.

L07 - Missing Events Arithmetic

Criticality	Minor / Informative
Location	CryptoOne.sol#L100
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.

```
Shell
swapAtAmount = _newSwapAtAmount
```

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.

Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
CryptoOne	Implementation	Ownable, ERC20, ReentrancyGuard		
		Public	✓	Ownable
	_setupApprovals	Internal	✓	
	createPair	Internal	✓	
	setBuyTaxWallet	Public	✓	onlyOwner
	setSellTaxWallet	Public	✓	onlyOwner
	excludeFromFees	Public	✓	onlyOwner
	isExcludedFromFees	Public		-
	setSwapAtAmount	External	✓	onlyOwner
	_transfer	Internal	✓	
	SwapFees	Private	✓	lockTheSwap nonReentrant
	manualSwap	External	✓	onlyOwner
	currentTaxSplit	Public		-
		External	Payable	-

Summary

Crypto One contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. Crypto One is an interesting project that has a friendly and growing community. The Smart Contract analysis reported no compiler error 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 2% 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 TAC 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.

A **TAC Security** Company

The Cyberscope team

cyberscope.io