Cyberscope

Audit Report

Capone

May 2023

Network	BSC
Address	0xf77F55995dfaC3786dc341eC1D6342a803441613
Audited by	© cyberscope

Table of Contents

Review

Contract Name	CAPONE
Compiler Version	v0.8.4+commit.c7e474f2
Optimization	99999 runs
Explorer	https://bscscan.com/address/0xf77f55995dfac3786dc341ec1d6342a803441613
Address	0xf77f55995dfac3786dc341ec1d6342a803441613
Network	BSC
Symbol	CAPONE
Decimals	18
Total Supply	10,000,000,000

Audit Updates

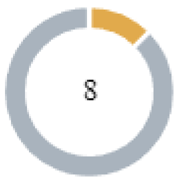
Initial Audit	15 May 2023 https://github.com/cyberscope-io/audits/blob/main/capone/v1/audit.pdf
Corrected Phase 2	17 May 2023

Source Files

Filename	SHA256
@openzeppelin/contracts/access/AccessControl.sol	0ab66c9c0b45fca5efad935058e889bd5bb5599eb95b0d17ec924f64ebcaf38f
@openzeppelin/contracts/access/IAccessControl.sol	d03c1257f2094da6c86efa7aa09c1c07ebd33dd31046480c5097bc2542140e45
@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol	af5c8a77965cc82c33b7ff844deb9826166689e55dc037a7f2f790d057811990
@openzeppelin/contracts/token/ERC20/IERC20.sol	94f23e4af51a18c2269b355b8c7cf4db8003d075c9c541019eb8dcf4122864d5
@openzeppelin/contracts/utils/Context.sol	1458c260d010a08e4c20a4a517882259a23a4baa0b5bd9add9fb6d6a1549814a
@openzeppelin/contracts/utils/introspection/ERC165.sol	8806a632d7b656cadb8133ff8f2acae4405b3a64d8709d93b0fa6a216a8a6154
@openzeppelin/contracts/utils/introspection/IERC165.sol	701e025d13ec6be09ae892eb029cd83b3064325801d73654847a5fb11c58b1e5
@openzeppelin/contracts/utils/Strings.sol	8597c62818dcbc6cf85c21179b90b714fb4f70a4347ca2eed23e88c87b08b8a1
@uniswap/v2-core/contracts/interfaces/IUniswapV2Factory.sol	51d056199e3f5e41cb1a9f11ce581aa3e190cc982db5771ffeef8d8d1f962a0d
@uniswap/v2-periphery/contracts/interfaces/IUniswapV2Router01.sol	0439ffe0fd4a5e1f4e22d71ddbda76d63d61679947d158cba4ee0a1da60cf663
@uniswap/v2-periphery/contracts/interfaces/IUniswapV2Router02.sol	a2900701961cb0b6152fc073856b972564f7c798797a4a044e83d2ab8f0e8d38
contracts/CAPONE.sol	c48ea07d4b79c7ecec469119695392013f7803fe5b55279b7f00853cebac1cbf

contracts/Distribution.sol	2aee6aaf8a9e96593c63590fde4c3e54911a8a48aaae18c207ea100f70768657
contracts/interfaces/IDistribution.sol	2ad3a198a8ab4b833d1b3a13c0ae6c5fe61a871655c8af34074a395e5bf81fd6

Findings Breakdown



●	Critical	0
●	Medium	1
●	Minor / Informative	7

Severity		Unresolved	Acknowledged	Resolved	Other
●	Critical	0	0	0	0
●	Medium	1	0	0	0
●	Minor / Informative	7	0	0	0

Analysis

● Critical ● Medium ● Minor / Informative ● Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Passed
●	OCTD	Transfers Contract's Tokens	Passed
●	OTUT	Transfers User's Tokens	Passed
●	ELFM	Exceeds Fees Limit	Passed
●	ULTW	Transfers Liquidity to Team Wallet	Passed
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	AFI	Accumulated Fees Inconsistency	Unresolved
●	RME	Redundant Mapping Entries	Unresolved
●	MVN	Misleading Variables Naming	Unresolved
●	CR	Code Repetition	Unresolved
●	IDI	Immutable Declaration Improvement	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L13	Divide before Multiply Operation	Unresolved
●	L19	Stable Compiler Version	Unresolved

AFI - Accumulated Fees Inconsistency

Criticality	Medium
Location	contracts/CAPONE.sol#L221,310,336
Status	Unresolved

Description

The contract resets the accumulated fees variables without performing a distribution of the accumulated rewards. This creates an inconsistency between the actual accumulated tokens from the fees, as the reset effectively clears the stored values without properly accounting for their distribution.

```
function resetRewardsAmount() external
onlyRole(DEFAULT_ADMIN_ROLE) {
  rewardSellAmount = 0;
  rewardBuyAmount = 0;

  emit RewardsAmountReseted();
}

function resetLiquidityFee() external
onlyRole(DEFAULT_ADMIN_ROLE) {
  liquidityFeeAmount = 0;

  emit LiquidityFeeReseted();
}

function resetSwapFee() external onlyRole(DEFAULT_ADMIN_ROLE) {
  swapFeeAmount = 0;

  emit SwapFeeReseted();
}
```

Recommendation

It is recommended to perform proper distribution of the accumulated rewards before resetting the accumulated variables. This ensures that users receive their deserved rewards based on their interactions with the contract.

RME - Redundant Mapping Entries

Criticality	Minor / Informative
Location	contracts/CAPONE.sol#L164,173,179
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 performs redundant mapping entries. The contract changes the mapping entries boolean even if they are already set to the corresponding state.

```
function setLpToken(address _lpToken, bool _lp) external
onlyRole(DEFAULT_ADMIN_ROLE) {
    require(_lpToken != address(0), "BEP20: invalid LP
address");
    require(_lpToken != pair, "ERC20: exclude default pair");

    isLpToken[_lpToken] = _lp;

    emit LpTokenUpdated(_lpToken, _lp);
}

function setExcludedFromFee(address _address, bool
_isExcludedFromFee) public onlyRole(DEFAULT_ADMIN_ROLE) {
    excludedFromFee[_address] = _isExcludedFromFee;

    emit ExcludedFromFee(_address, _isExcludedFromFee);
}

function setExcludedFromSwap(address _address, bool
_isExcludedFromSwap) public onlyRole(DEFAULT_ADMIN_ROLE) {
    excludedFromSwap[_address] = _isExcludedFromSwap;

    emit ExcludedFromSwap(_address, _isExcludedFromSwap);
}
```

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.

The contract could prevalidate that the state of the mapping is different.

MVN - Misleading Variables Naming

Criticality	Minor / Informative
Location	contracts/CAPONE.sol#L39,390
Status	Unresolved

Description

Variables can have misleading names if their names do not accurately reflect the value they contain or the purpose they serve. The contract uses some variable names that are too generic or do not clearly convey the information stored in the variable. Misleading variable names can lead to confusion, making the code more difficult to read and understand.

The contract incorporates the capability to distribute burn fees among multiple addresses. On the contrary, the burn mechanism facilitates the transfer of tokens to a dead address for the purpose of burning them.

```
uint256 public burnFeeBuyRate;
uint256 public burnFeeSellRate;
uint256 public burnFeeTransferRate;
address[] public burnFeeReceivers;
uint256[] public burnFeeReceiversRate;

for (uint256 i = 0; i < burnFeeReceivers.length; i++) {
    _transferAmount(_from, burnFeeReceivers[i],
_calcFee(burnFeeRes, burnFeeReceiversRate[i]));
}
```

Recommendation

It's always a good practice for the contract to contain variable names that are specific and descriptive. The team is advised to keep in mind the readability of the code.

CR - Code Repetition

Criticality	Minor / Informative
Location	contracts/CAPONE.sol#L186,271,291,317
Status	Unresolved

Description

The contract contains repetitive code segments. There are potential issues that can arise when using code segments in Solidity. Some of them can lead to issues like gas efficiency, complexity, readability, security, and maintainability of the source code. It is generally a good idea to try to minimize code repetition where possible.

```
require(_burnFeeReceivers.length == _burnFeeReceiversRate.length,
"size");

uint256 totalRate = 0;
for (uint256 i = 0; i < _burnFeeReceiversRate.length; i++) {
    totalRate += _burnFeeReceiversRate[i];
}
require(totalRate == 10000, "rate");
```

Recommendation

The team is advised to avoid repeating the same code in multiple places, which can make the contract easier to read and maintain. The authors could try to reuse code wherever possible, as this can help reduce the complexity and size of the contract. For instance, the contract could reuse the common code segments in an internal function in order to avoid repeating the same code in multiple places.

IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	contracts/CAPONE.sol#L102,103,109
Status	Unresolved

Description

The contract is using variables that initialize them only in the constructor. The other functions are not mutating the variables. These variables are not defined as `immutable`.

```
name
symbol
distribution
```

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	contracts/CAPONE.sol#L123,127,132,136,141,151,156,164,173,179,185,205,213,228,238,248,264,270,290,316,342
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.

```
address _account
address _recipient
uint256 _amount
address _owner
address _spender
address _sender
uint256 _addedValue
uint256 _subtractedValue
address _lpToken
bool _lp
address _address
bool _isExcludedFromFee
bool _isExcludedFromSwap
address[] calldata _rewardSwapReceivers

...
```

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.

L13 - Divide before Multiply Operation

Criticality	Minor / Informative
Location	contracts/CAPONE.sol#L417,444
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.

```
uint256 liquidityFeeHalf = liquidityFeeAmount / 2
uint256 liquidityFeeToken1Amount = _calcFee(token1Balance,
liquidityFeeHalf * 10000 / amountToSwap)
```

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	contracts/CAPONE.sol#L2
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 ^0.8.2;
```

Recommendation

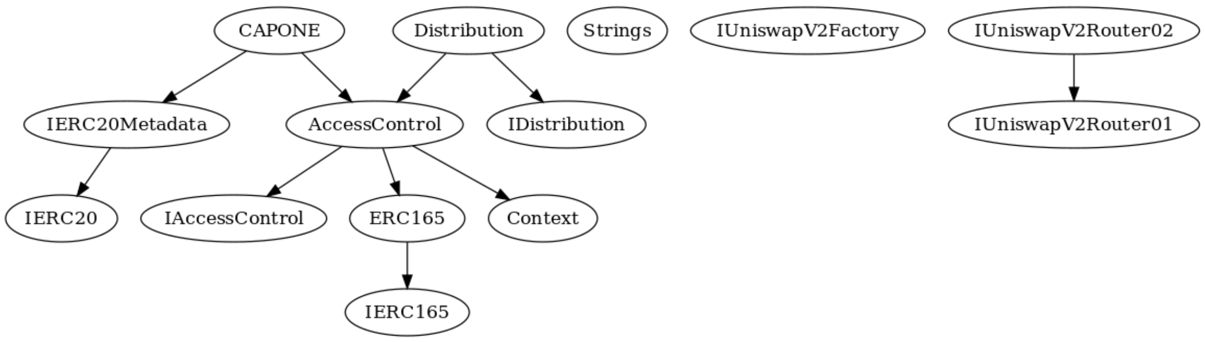
The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest 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

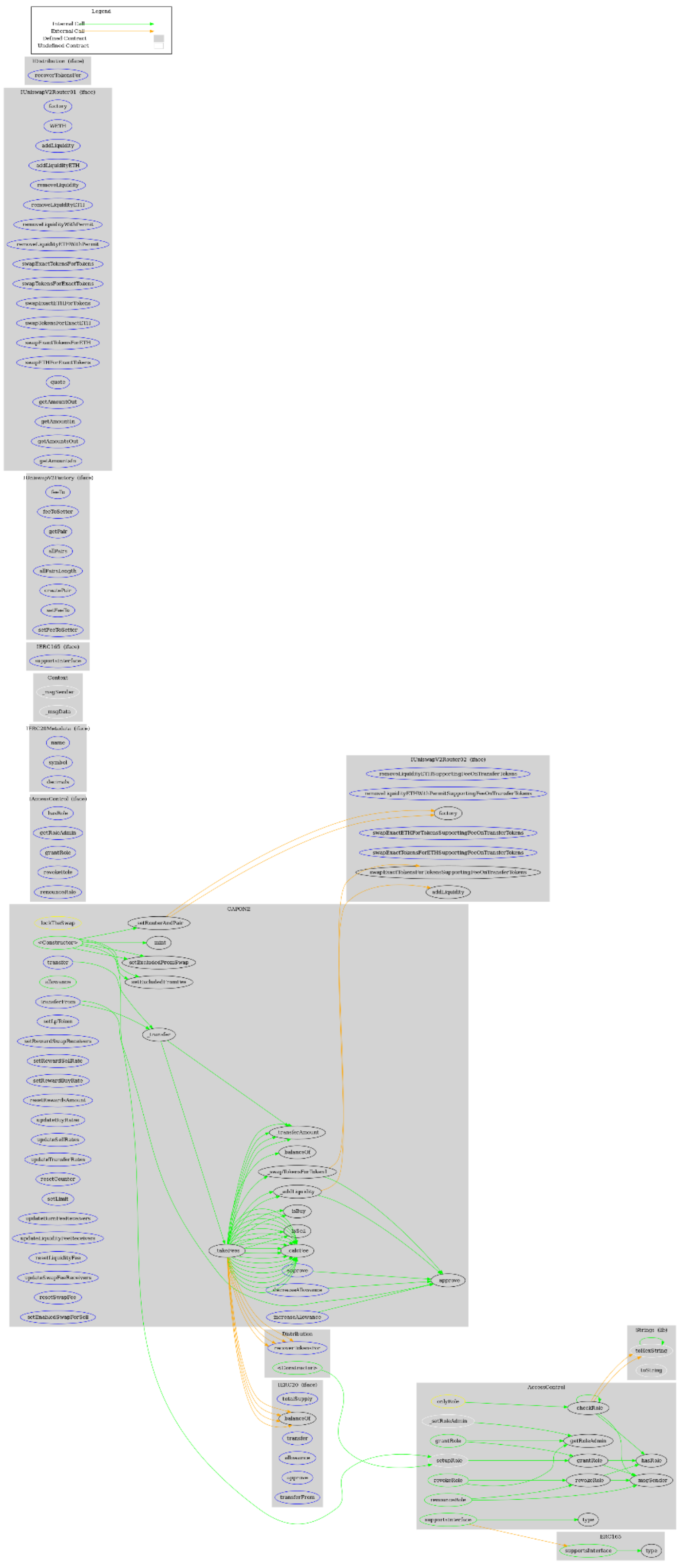
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
CAPONE	Implementation	IERC20Metadata, AccessControl		
		Public	✓	-
	balanceOf	Public		-
	transfer	External	✓	-
	allowance	Public		-
	approve	External	✓	-
	transferFrom	External	✓	-
	increaseAllowance	External	✓	-
	decreaseAllowance	External	✓	-
	setLpToken	External	✓	onlyRole
	setExcludedFromFee	Public	✓	onlyRole
	setExcludedFromSwap	Public	✓	onlyRole
	setRewardSwapReceivers	External	✓	onlyRole
	setRewardSellRate	External	✓	onlyRole
	setRewardBuyRate	External	✓	onlyRole
	resetRewardsAmount	External	✓	onlyRole
	updateBuyRates	External	✓	onlyRole

	updateSellRates	External	✓	onlyRole
	updateTransferRates	External	✓	onlyRole
	resetCounter	External	✓	onlyRole
	setLimit	External	✓	onlyRole
	updateBurnFeeReceivers	External	✓	onlyRole
	updateLiquidityFeeReceivers	External	✓	onlyRole
	resetLiquidityFee	External	✓	onlyRole
	updateSwapFeeReceivers	External	✓	onlyRole
	resetSwapFee	External	✓	onlyRole
	setEnabledSwapForSell	External	✓	onlyRole
	_transfer	Internal	✓	
	_takeFees	Internal	✓	
	_transferAmount	Internal	✓	
	_mint	Internal	✓	
	_approve	Internal	✓	
	_setRouterAndPair	Internal	✓	
	_calcFee	Internal		
	_isSell	Internal		
	_isBuy	Internal		
	_swapTokensForToken1	Internal	✓	lockTheSwap
	_addLiquidity	Internal	✓	lockTheSwap

Inheritance Graph



Flow Graph



Summary

Capone contract implements a token mechanism. This audit investigates security issues, business logic concerns, and potential improvements. Capone 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 a max 9% fee. Additionally, the contract has a fee limit mechanism.

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