Cyberscope

Audit Report

Based Yoda

April 2024

Network	BASE
Address	0x6bd81aAd9B25Ad1E0b99c47eD01B34eAcF4B8bE7
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
●	PLPI	Potential Liquidity Provision Inadequacy	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L09	Dead Code Elimination	Unresolved
●	L14	Uninitialized Variables in Local Scope	Unresolved
●	L15	Local Scope Variable Shadowing	Unresolved

Table of Contents

Review

Contract Name	Based Yoda
Explorer	https://basescan.org/address/0x6bd81aad9b25ad1e0b99c47ed01b34eacf4b8be7
Symbol	BODA
Decimals	18
Total Supply	100,000,000,000
Badge Eligibility	Yes

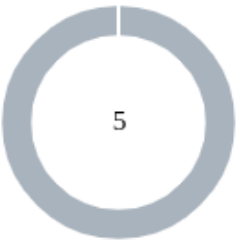
Audit Updates

Initial Audit	05 Apr 2024

Source Files

Filename	SHA256
BasedYoda.sol	28e9dbd98e26b03ed121742e02e4c653441babfaa3a74d3aa3b4d1e3afa0e4ab

Findings Breakdown



Critical	0
Medium	0
Minor / Informative	5

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	0	0	0	0
Minor / Informative	5	0	0	0

PLPI - Potential Liquidity Provision Inadequacy

Criticality	Minor / Informative
Location	BasedYoda.sol#L926
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 _swapTokensForETH(uint256 tokenAmt) private {
        address[] memory path = new address[](2);
        path[0] = address(this);
        path[1] = dexRouter.WETH();


dexRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(
            tokenAmt,
            0,
            path,
            address(marketingAddress),
            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.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	BasedYoda.sol#L552,687,699,700,715,748,765,795,809
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 WETH() external pure returns (address);
function setSwapTokensAtAmt(uint256 _swapTokensAtAmt) external
onlyOwner {
address _address,
bool _withExemption
bool withPair_
function updateLpPairAddress(address _lpPair) public onlyOwner
{
function updateTaxes(uint256 _buyTax, uint256 _sellTax)
external onlyOwner {
function updateMaxWalletAmount(uint256 _maxWallet) external
onlyOwner {
```

```
function updateMaxTxAmount(uint256 _maxTxAmount) external
onlyOwner {
```

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.

L09 - Dead Code Elimination

Criticality	Minor / Informative
Location	BasedYoda.sol#L403
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 _burn(address account, uint256 amount) internal
virtual {
    require(account != address(0), "ERC20: burn from the zero
address");
    _beforeTokenTransfer(account, address(0), amount);
    uint256 accountBalance = _balances[account];
    require(accountBalance >= amount, "ERC20: burn amount
exceeds balance");
    unchecked {
        _balances[account] = accountBalance - amount;
        // Overflow not possible: amount <= accountBalance <=
totalSupply.
        _totalSupply -= amount;
    }
    emit Transfer(account, address(0), amount);
    _afterTokenTransfer(account, address(0), amount);
}
```

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.

L14 - Uninitialized Variables in Local Scope

Criticality	Minor / Informative
Location	BasedYoda.sol#L625
Status	Unresolved

Description

Using an uninitialized local variable can lead to unpredictable behavior and potentially cause errors in the contract. It's important to always initialize local variables with appropriate values before using them.

```
address _v2Router;
```

Recommendation

By initializing local variables before using them, the contract ensures that the functions behave as expected and avoid potential issues.

L15 - Local Scope Variable Shadowing

Criticality	Minor / Informative
Location	BasedYoda.sol#L622
Status	Unresolved

Description

Local scope variable shadowing occurs when a local variable with the same name as a variable in an outer scope is declared within a function or code block. When this happens, the local variable "shadows" the outer variable, meaning that it takes precedence over the outer variable within the scope in which it is declared.

```
uint256 _totalSupply = 100_000_000_000 * (10 ** decimals());
```

Recommendation

It's important to be aware of shadowing when working with local variables, as it can lead to confusion and unintended consequences if not used correctly. It's generally a good idea to choose unique names for local variables to avoid shadowing outer variables and causing confusion.

Functions Analysis

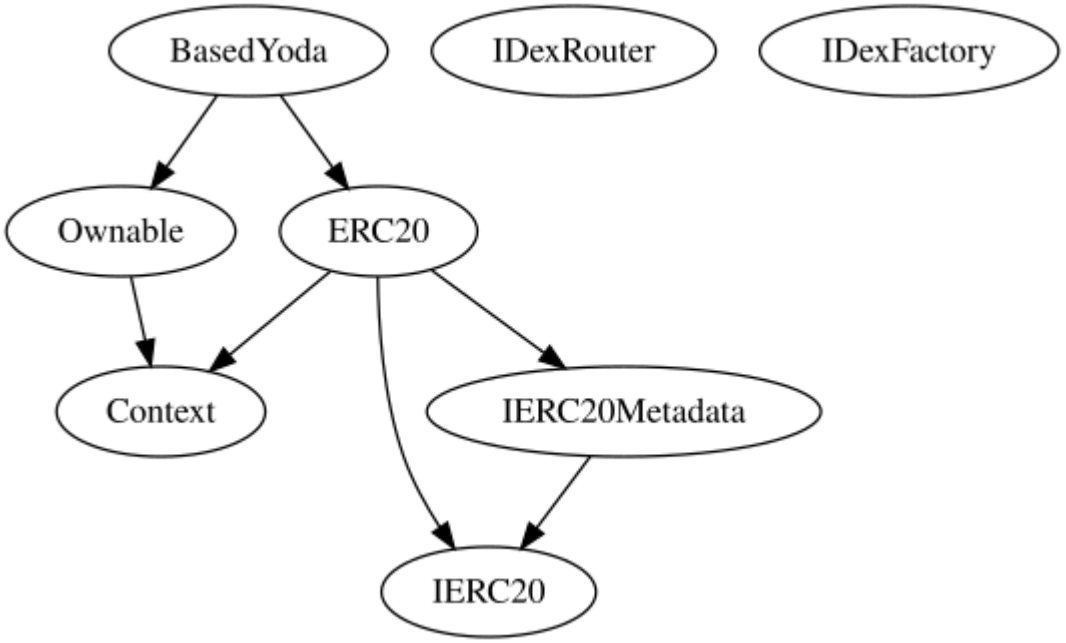
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IERC20Metadata	Interface	IERC20		
	name	External		-
	symbol	External		-
	decimals	External		-
ERC20	Implementation	Context, IERC20, IERC20Metadata		

		Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	_transfer	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_approve	Internal	✓	
	_spendAllowance	Internal	✓	
	_beforeTokenTransfer	Internal	✓	
	_afterTokenTransfer	Internal	✓	
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-

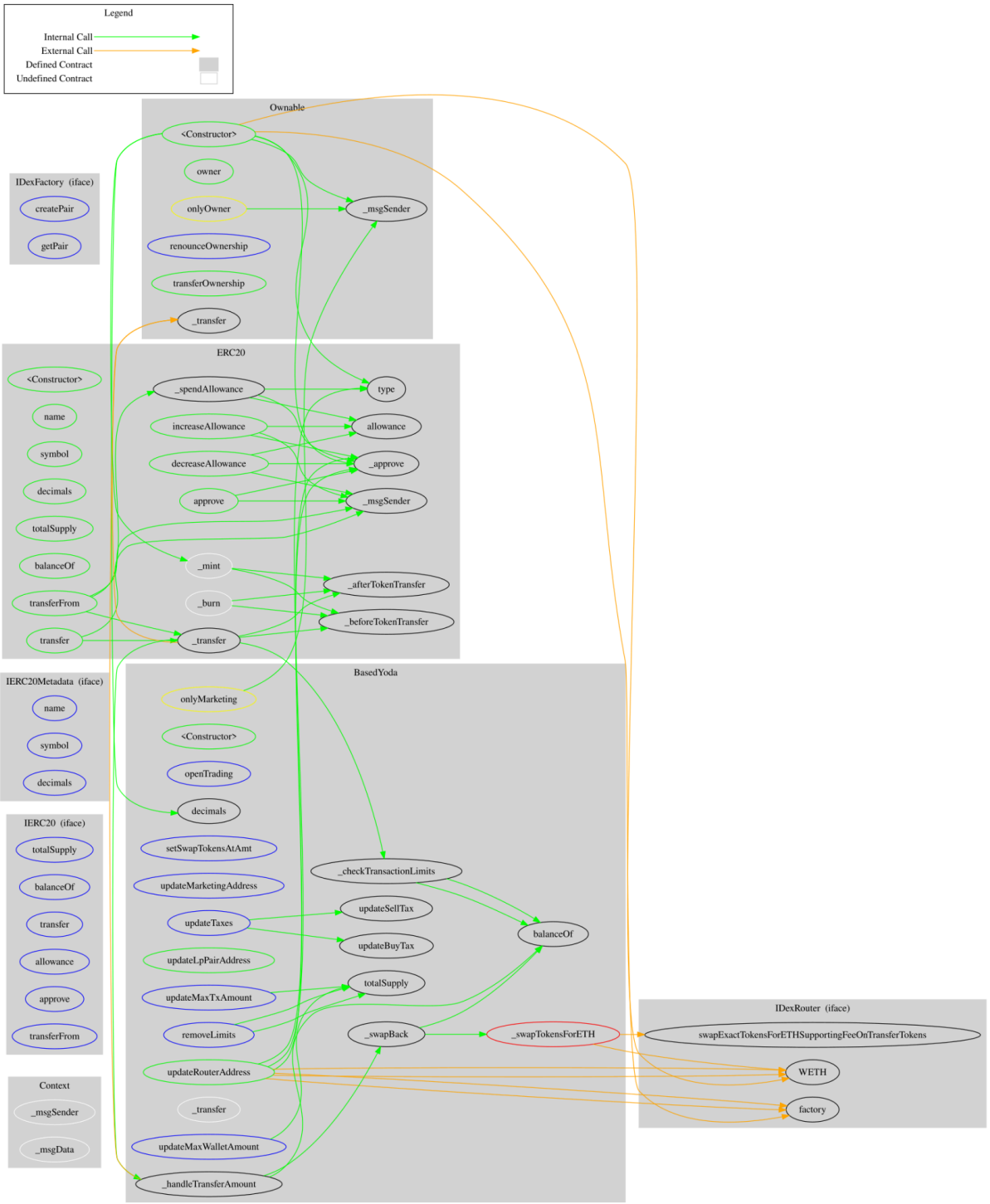
	renounceOwnership	External	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
IDexRouter	Interface			
	factory	External		-
	WETH	External		-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
IDexFactory	Interface			
	createPair	External	✓	-
	getPair	External		-
BasedYoda	Implementation	ERC20, Ownable		
		Public	✓	ERC20
	openTrading	External	✓	onlyOwner
	removeLimits	External	✓	onlyOwner
	setSwapTokensAtAmt	External	✓	onlyOwner
	updateMarketingAddress	External	✓	onlyOwner
	updateRouterAddress	Public	✓	onlyOwner
	updateLpPairAddress	Public	✓	onlyOwner
	updateTaxes	External	✓	onlyOwner
	updateBuyTax	Public	✓	onlyOwner
	updateSellTax	Public	✓	onlyOwner

	updateMaxWalletAmount	External	✓	onlyOwner
	updateMaxTxAmount	External	✓	onlyOwner
	_transfer	Internal	✓	
	_checkTransactionLimits	Internal		
	_handleTransferAmount	Internal	✓	
	_swapTokensForETH	Private	✓	
	_swapBack	Private	✓	

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
Ownable
<Constructor>
owner
onlyOwner
renounceOwnership
transferOwnership
_transfer
_msgSender
IDexFactory (iface)
createPair
getPair
ERC20
<Constructor>
name
symbol
decimals
totalSupply
balanceOf
transferFrom
transfer
_spendAllowance
increaseAllowance
decreaseAllowance
approve
type
allowance
_approve
_msgSender
_mint
_burn
_transfer
_afterTokenTransfer
_beforeTokenTransfer
IERC20Metadata (iface)
name
symbol
decimals
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
Context
_msgSender
_msgData
BasedYoda
onlyMarketing
<Constructor>
openTrading
decimals
setSwapTokensAtAmt
updateMarketingAddress
updateTaxes
updateLpPairAddress
updateMaxTxAmount
removeLimits
updateRouterAddress
_transfer
updateMaxWalletAmount
_handleTransferAmount
_checkTransactionLimits
updateSellTax
updateBuyTax
totalSupply
_swapBack
balanceOf
_swapTokensForETH
IDexRouter (iface)
swapExactTokensForETHSupportingFeeOnTransferTokens
WETH
factory

Summary

Based Yoda contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. Based Yoda 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.

The contract's ownership has been renounced. The information regarding the transaction can be accessed through the following link:

https://basescan.org/tx/0x90279a75133cfa1c5f4005aa07a35d600e6d6b53224beefa97590528e1470ffc

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