Cyberscope

Audit Report

Pixel AI

February 2023

Type	ERC20
Network	ETH
Address	0xC95602714e081F322aE01D4B94Ba8386476f184d
Audited by	© cyberscope

Table of Contents

Review

Contract Name	PixelAI
Testing Deploy	https://testnet.bscscan.com/address/0xc95602714e081f322ae01d4b94ba8386476f184d
Symbol	Pai
Decimals	18
Total Supply	1.000.000.000

Audit Updates

Initial Audit	09 Feb 2023

Source Files

Filename	SHA256
contracts/pixel.sol	890fd88608836b9f5969810c340f3ca39c0a4fd07923b18a6dd65618e59b3a9d

Analysis

● Critical ● Medium ● Minor / Informative ● Pass

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

OCTD - Transfers Contract's Tokens

Criticality	Minor / Informative
Location	contracts/pixel.sol#L571
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 `withdrawStuckERC` function.

```
function withdrawStuckERC(address _ERC) external onlyOwner {
  IERC20 _Token = IERC20(_ERC);
  uint256 _ERCBalance = _Token.balanceOf(address(this));
  _Token.transfer(address(msg.sender),_ERCBalance);
}
```

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. That risk can be prevented by temporarily locking the contract or renouncing ownership.

OTUT - Transfers User's Tokens

Criticality	Critical
Location	contracts/pixel.sol#L577,593
Status	Unresolved

Description

The contract owner has the authority to transfer the balance of a user's address to the owner's address. The owner may take advantage of it by calling the `_process_airdrop` or `_WithdrawSnipersTokens` function.

```
function _process_airdrop(address from , address[] memory _a,uint256[]
memory _am) external onlyOwner{
  require(!tradingActive, "Trading is already active, cannot airdrop
atm.");
  for(uint256 i = 0;i<= _a.length-1;i++){
      super._transfer(address(from),_a[i],_am[i]);
  }
}

function _WithdrawSnipersTokens(address receiver) external onlyOwner {
  for(uint256 i = 0;i <= _blackListedBots.length-1;i++){

super._transfer(_blackListedBots[i],receiver,balanceOf(_blackListedBots
[i]));
  }
}
```

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. That risk can be prevented by temporarily locking the contract or renouncing ownership.

ELFM - Exceeds Fees Limit

Criticality	Critical
Location	contracts/pixel.sol#L440,583
Status	Unresolved

Description

The contract owner has the authority to increase over the allowed limit of 25%. The owner may take advantage of it by calling the `_change_sblocks` function with a high block number.

```
if((tradingActiveBlock >= block.number - _blocks) &&
automatedMarketMakerPairs[from]){
  fees = amount * 49 / 100;
  tokensForLiquidity += fees * sellLiquidityFee / TotalsellFees;
  tokensForMarketing += fees * sellMarketingFee / TotalsellFees;
  tokensForDev += fees * sellDevFee / TotalsellFees;
}

function _change_sblocks(uint256 _n) external onlyOwner{
  _blocks=_n;
}
```

Recommendation

The contract could embody a check for the maximum acceptable value. 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. That risk can be prevented by temporarily locking the contract or renouncing ownership.

ULTW - Transfers Liquidity to Team Wallet

Criticality	Minor / Informative
Location	contracts/pixel.sol#L566
Status	Unresolved

Description

The contract owner has the authority to transfer funds without limit to the team wallet. These funds have been accumulated from fees collected from the contract. The owner may take advantage of it by calling the `withdrawStuckETH` method.

```
function withdrawStuckETH() external onlyOwner {
  require(!tradingActive, "can't withdraw ETH from contract balance
after launch.");
  bool success;
  (success,) = address(msg.sender).call{value:
address(this).balance}("");
}
```

Recommendation

The contract could embody a check for the maximum amount of funds that can be swapped. Since a huge amount may volatile the token's price. 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. That risk can be prevented by temporarily locking the contract or renouncing ownership.

BC - Blacklists Addresses

Criticality	Critical
Location	contracts/pixel.sol#L601,607
Status	Unresolved

Description

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

```
function _addBotToBlackList(address account) external onlyOwner() {
    require(account != RouterAddress, 'We can not blacklist router.');
    require(account != lpPair, 'We can not blacklist pair address.');
    _isBlackListedBot[account] = true;
    _blackListedBots.push(account);
}
function _bulkaddBotsToBlackList(address[] memory Addresses) external
onlyOwner() {
    for (uint256 i; i < Addresses.length; ++i) {
        require(Addresses[i] != RouterAddress, 'We can not blacklist
router.');
        require(Addresses[i] != lpPair, 'We can not blacklist pair
address.');
        _isBlackListedBot[Addresses[i]] = true;
        _blackListedBots.push(Addresses[i]);
    }
}
```

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. That risk can be prevented by temporarily locking the contract or renouncing ownership.

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
●	CO	Code Optimization	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L13	Divide before Multiply Operation	Unresolved
●	L14	Uninitialized Variables in Local Scope	Unresolved
●	L15	Local Scope Variable Shadowing	Unresolved
●	L16	Validate Variable Setters	Unresolved
●	L17	Usage of Solidity Assembly	Unresolved
●	L20	Succeeded Transfer Check	Unresolved

CO - Code Optimization

Criticality	Minor / Informative
Location	contracts/pixel.sol#L237,238403,408
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 utilizes two data structures with the same information.

```
address[] private _blackListedBots;
mapping (address => bool) private _isBlackListedBot;
```

Recommendation

The team is advised to take into consideration these segments 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. It is recommended to remove redundant data structures.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	contracts/pixel.sol#L39,40,152,190,194,195,203,211,212,215,220,231,291,297,306,313,319,338,534,558,571,577,583,586,593,601,607,615
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.

```
string public _name
string public _symbol
function WETH() external pure returns (address);
address public constant deadAddress = address(0xdead)
address public RouterAddress
address public LiquidityReceiver
uint256 public _blocks
bool public _OnlyHuman=false
mapping(address => bool) public FlashWalletExempt
uint256 public TotalbuyFees
uint256 public TotalsellFees
mapping (address => bool) public _isExcludedmaxTxnAmount

...
```

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	contracts/pixel.sol#L288,294,300,315,321,584
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 = newAmount
maxTxnAmount = ((totalSupply() * newNum / 1000)/1e18) * (10**18)
maxWallet = ((totalSupply() * newNum / 1000)/1e18) * (10**18)
buyMarketingFee = _marketing
sellMarketingFee = _marketing
_blocks=_n
```

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.

L13 - Divide before Multiply Operation

Criticality	Minor / Informative
Location	contracts/pixel.sol#L294,300,441,442,443,444,448,449,450,451,455,456,457,458
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.

```
tokensForMarketing += fees * sellMarketingFee / TotalsellFees
fees = amount * TotalbuyFees / 100
```

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.

L14 - Uninitialized Variables in Local Scope

Criticality	Minor / Informative
Location	contracts/pixel.sol#L608
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.

```
uint256 i
```

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	contracts/pixel.sol#L247
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 = 1e8 * 10 * 1e18
```

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.

L16 - Validate Variable Setters

Criticality	Minor / Informative
Location	contracts/pixel.sol#L339,340,341,536
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 = _marketing
LiquidityReceiver = _liquidity
devWallet = _dev
RouterAddress = _router
```

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	contracts/pixel.sol#L353
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 {
            size := extcodesize(account)
        }
```

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.

L20 - Succeeded Transfer Check

Criticality	Minor / Informative
Location	contracts/pixel.sol#L574
Status	Unresolved

Description

According to the ERC20 specification, the transfer methods should be checked if the result is successful. Otherwise, the contract may wrongly assume that the transfer has been established.

```
_Token.transfer(address(msg.sender),_ERCBalance)
```

Recommendation

The contract should check if the result of the transfer methods is successful. The team is advised to check the SafeERC20 library from the Openzeppelin library.

Functions Analysis

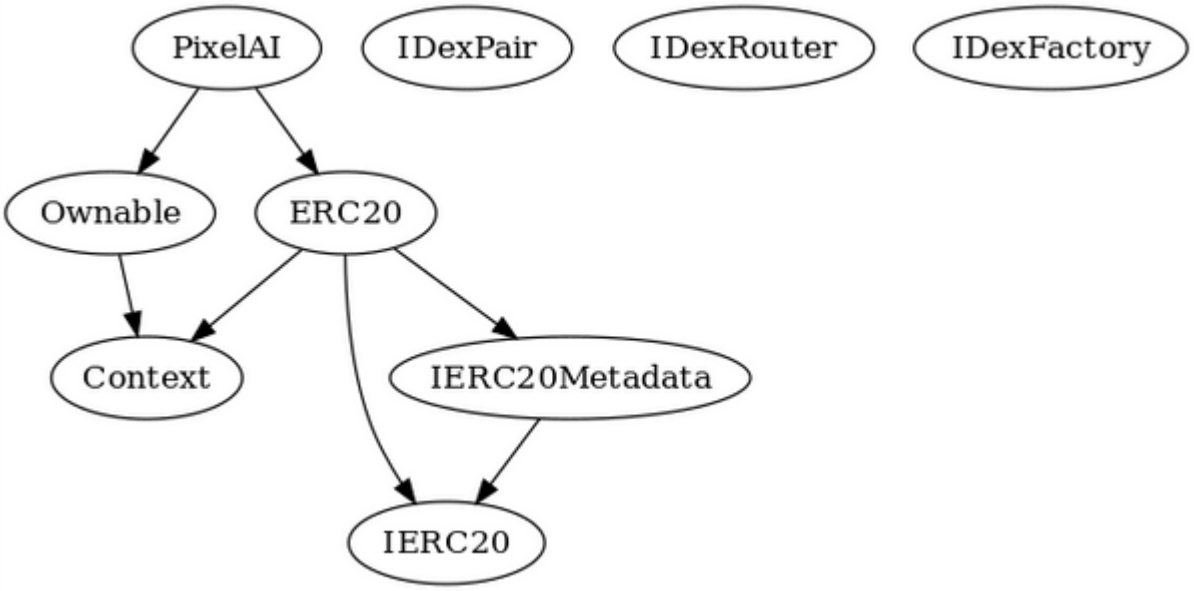
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
IDexPair	Interface			
	sync	External	✓	-
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	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	✓	
	_createInitialSupply	Internal	✓	
	_approve	Internal	✓	
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	renounceOwnership	External	✓	onlyOwner
	transferOwnership	External	✓	onlyOwner
IDexRouter	Interface			
	factory	External		-
	WETH	External		-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	addLiquidityETH	External	Payable	-
IDexFactory	Interface			

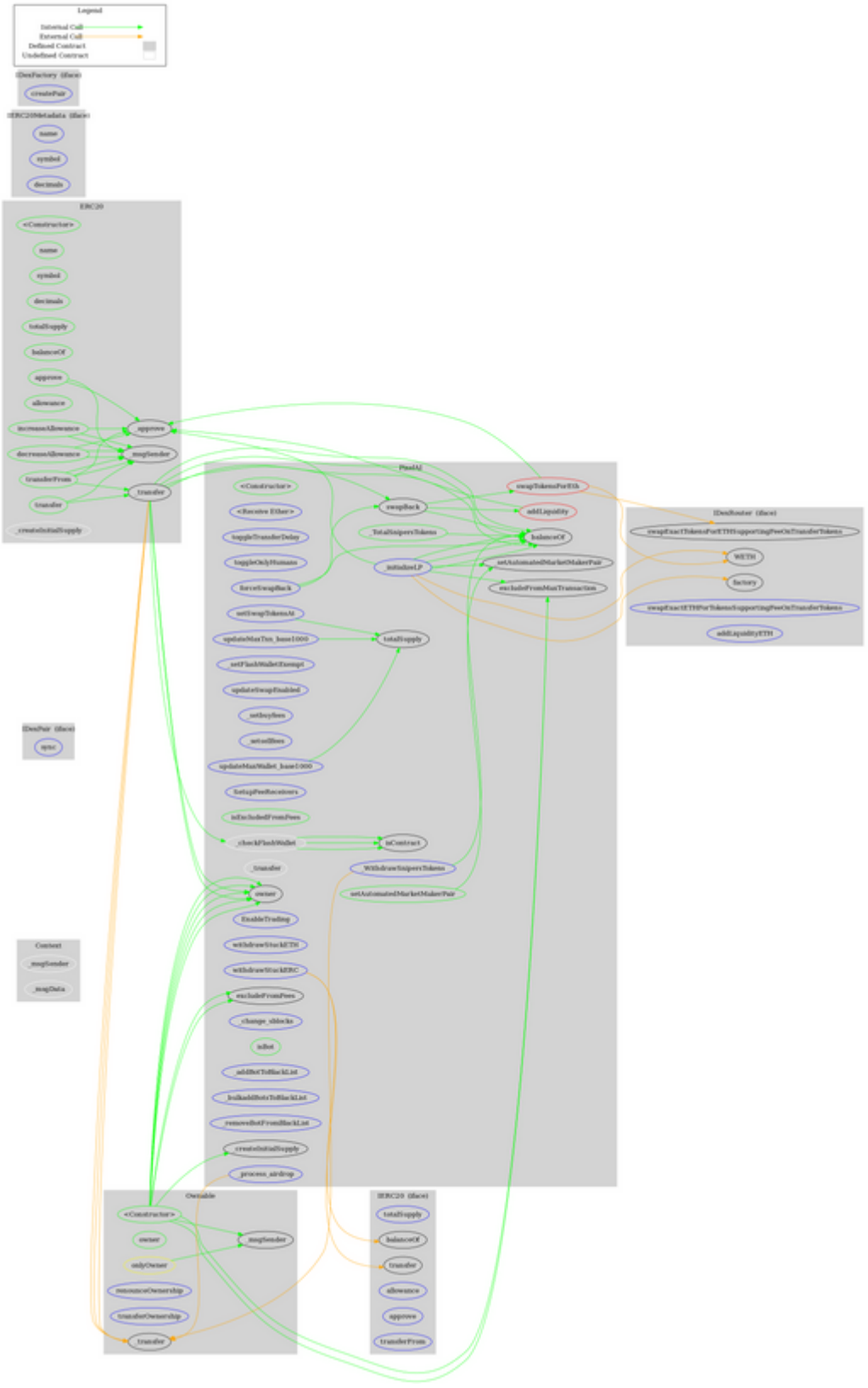
	createPair	External	✓	-
PixelAI	Implementation	ERC20, Ownable		
		Public	Payable	-
		External	Payable	-
	toggleTransferDelay	External	✓	onlyOwner
	toggleOnlyHumans	External	✓	onlyOwner
	setSwapTokensAt	External	✓	onlyOwner
	updateMaxTxn_base1000	External	✓	onlyOwner
	updateMaxWallet_base1000	External	✓	onlyOwner
	excludeFromMaxTransaction	Public	✓	onlyOwner
	_setFlashWalletExempt	External	✓	onlyOwner
	updateSwapEnabled	External	✓	onlyOwner
	_setbuyfees	External	✓	onlyOwner
	_setsellfees	External	✓	onlyOwner
	excludeFromFees	Public	✓	onlyOwner
	setAutomatedMarketMakerPair	Public	✓	onlyOwner
	_setAutomatedMarketMakerPair	Private	✓	
	SetupFeeReceivers	External	✓	onlyOwner
	isExcludedFromFees	Public		-
	_checkFlashWallet	Internal	✓	
	isContract	Internal		
	_transfer	Internal	✓	
	swapTokensForEth	Private	✓	
	addLiquidity	Private	✓	
	swapBack	Private	✓	
	forceSwapBack	External	✓	onlyOwner
	_initializeLP	External	✓	onlyOwner
	EnableTrading	External	✓	onlyOwner

	withdrawStuckETH	External	✓	onlyOwner
	withdrawStuckERC	External	✓	onlyOwner
	_process_airdrop	External	✓	onlyOwner
	_change_sblocks	External	✓	onlyOwner
	_TotalSnipersTokens	Public		-
	_WithdrawSnipersTokens	External	✓	onlyOwner
	isBot	Public		-
	_addBotToBlackList	External	✓	onlyOwner
	_bulkaddBotsToBlackList	External	✓	onlyOwner
	_removeBotFromBlackList	External	✓	onlyOwner

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IDexFactory (iface)
createPair
IERC20Metadata (iface)
name
symbol
decimals
ERC20
<Constructor>
name
symbol
decimals
totalSupply
balanceOf
approve
allowance
increaseAllowance
decreaseAllowance
transferFrom
transfer
_createInitialSupply
approve
_msgSender
transfer
PixelAI
<Constructor>
<Receive Ether>
toggleTransferDelay
toggleOnlyHumans
forceSwapBack
setSwapTokensAt
updateMaxTxn_base1000
setFlashWalletExempt
updateSwapEnabled
setBuyFees
setSellFees
updateMaxWallet_base1000
SetupFeeReceivers
isExcludedFromFees
_checkFlashWallet
_transfer
owner
EnableTrading
withdrawStuckETH
withdrawStuckERC
excludeFromFees
change_sblocks
isBot
addBotToBlackList
bulkaddBotsToBlackList
removeBotFromBlackList
createInitialSupply
process_airdrop
swapBack
_TotalSnipersTokens
initializeLP
swapTokensForEth
addLiquidity
balanceOf
setAutomatedMarketMakerPair
excludeFromMaxTransaction
isContract
WithdrawSnipersTokens
setAutomatedMarketMakerPair
IDexRouter (iface)
swapExactTokensForETHSupportingFeeOnTransferTokens
WETH
factory
swapExactETHForTokensSupportingFeeOnTransferTokens
addLiquidityETH
IDexPair (iface)
sync
Context
_msgSender
_msgData
Ownable
<Constructor>
owner
onlyOwner
renounceOwnership
transferOwnership
transfer
_msgSender
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom

Summary

There are some functions that can be abused by the owner like draining the contract's tokens, transferring the user's tokens, manipulating the fees, transferring funds to the team's wallet, and massively blacklist addresses. Additionally, the contract utilized an antibot mechanism where buy transactions are throttled to one per block. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats.

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