Cyberscope

Audit Report

MicroPets

October 2023

Network BSC

Address 0xd7e8221abd7b8010f68b8a79ff3422fb9d794f98

Audited by © cyberscope

Analysis

Critical Medium Minor / Informative Pass

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

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
Medium	PBV	Percentage Boundaries Validation	Unresolved
Minor	RFI	Redundant Function Implementation	Unresolved
Minor	FO	Function Optimization	Unresolved
Minor	RV	Redundant Variable	Unresolved
Minor	PTRP	Potential Transfer Revert Propagation	Unresolved
Minor	PVC	Price Volatility Concern	Unresolved
Minor	PTAI	Potential Transfer Amount Inconsistency	Unresolved
Minor	MEE	Missing Events Emission	Unresolved
Minor	RSML	Redundant SafeMath Library	Unresolved
Minor	L02	State Variables could be Declared Constant	Unresolved
Minor	L04	Conformance to Solidity Naming Conventions	Unresolved
Minor	L07	Missing Events Arithmetic	Unresolved
Minor	L09	Dead Code Elimination	Unresolved
Minor	L19	Stable Compiler Version	Unresolved

●	L20	Succeeded Transfer Check	Unresolved

Table of Contents

Review

Contract Name	MicroPets
Compiler Version	v0.8.16+commit.07a7930e
Optimization	200 runs
Explorer	https://bscscan.com/address/0xd7e8221abd7b8010f68b8a79ff3422fb9d794f98
Address	0xd7e8221abd7b8010f68b8a79ff3422fb9d794f98
Network	BSC
Decimals	18

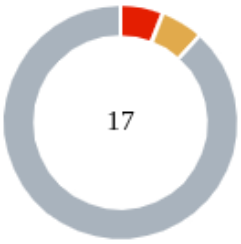
Audit Updates

Initial Audit	30 Oct 2023

Source Files

Filename	SHA256
Utils.sol	a81996b7539309494b258d006efc063124a085c5e27e182edaabe4d1358eadb5
Uniswap.sol	375dc68512149365bf130dc289ed683898eed5a7ba9547abd31b7c897e3d8bc8
Micropets.sol	cab6caf9a4fff8063c71094d00f4618e23c55dc4c85c0c665433e6ccae887279

Findings Breakdown



Critical	1
Medium	1
Minor / Informative	15

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	1	0	0	0
Medium	1	0	0	0
Minor / Informative	15	0	0	0

ELFM - Exceeds Fees Limit

Criticality	Critical
Location	Micropets.sol#L569
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 `setShares` function with a high percentage value.

```
    function setShares(
            ...
            uint8 buyTax,
            uint8 sellTax,
            uint24 autoSwapTier
        ) external onlyOwner {
            _setShares(
                ...
                buyTax,
                sellTax,
                autoSwapTier
            );
        }
```

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.

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.

OTUT - Transfers User's Tokens

Criticality	Minor / Informative
Location	Micropets.sol#L674
Status	Unresolved

Description

The contract allows users to migrate their token balances from an external address (`0xA77346760341460B42C230ca6D21d4c8E743Fa9c`) to the current contract address using the `migrate` function. However, users have the authority to transfer their balance of the token from another address, which remains out of the audit scope, to the current contract address. As a result a large amount of tokens can be transferred to the current address, leading to inconsistencies in the token balances of this address. The transferred amount can enable users to migrate tokens from the previous address, which might not have any value, to this address where the tokens might have value. This could artificially inflate the value of the tokens in the current contract.

```
function migrate() external {
        require(migrationRunning, "Migration over");

        uint256 V1Balance =
IERC20(0xA77346760341460B42C230ca6D21d4c8E743Fa9c)
            .balanceOf(_msgSender());

        require(V1Balance > 0, "Invalid migration");

        safeTransferFrom(
            0xA77346760341460B42C230ca6D21d4c8E743Fa9c,
            _msgSender(),
            address(this),
            V1Balance
        );

        uint256 newTokenAmount = V1Balance.div(migrationRate);
        _transferStandard(migrationVault, _msgSender(), newTokenAmount);
    }
```

Recommendation

It is recommended to implement a more streamlined method for token migration. The contract should be designed to handle the receipt of tokens in a manner that prevents any inconsistencies. Specifically, each migration could be limited to a fixed percentage of each holder's balance, ensuring that no user can migrate an excessive amount at once. This approach will ensure that migrations are conducted in a controlled manner, reducing the risk of sudden, large-scale migrations that could disrupt the token's ecosystem.

PBV - Percentage Boundaries Validation

Criticality	Medium
Location	Micropets.sol#L382,390,553
Status	Unresolved

Description

The contract is using the variables `coinShareLP` , `coinShareMarketing` , `coinShareDevelopment` , `coinShareStaking` , and `tokenShareReserve` for calculations. However, these variable are used in multiplication operations and if are set through the `_setShares` to a value greater than `100` , it could lead to incorrect calculations, potentially causing unintended behavior or financial discrepancies within the contract's operations.

```
uint256 rewardShare = tokensToSwap
    .mul(tokenConfigs.tokenShareReserve)
    .div(100);

function _calcuclateShare(
    uint8 share,
     uint256 amount
) internal pure returns (uint256) {
    return amount.mul(share).div(100);
}

function _setShares(
        uint8 coinShareLP,
        uint8 coinShareMarketing,
        uint8 coinShareDevelopment,
        uint8 coinShareStaking,
        uint8 tokenShareReserve,
        uint8 buyTax,
        uint8 sellTax,
        uint24 autoSwapTier
    ) internal {
        tokenConfigs.coinShareLP = coinShareLP;
        tokenConfigs.coinShareMarketing = coinShareMarketing;
        tokenConfigs.coinShareDevelopment =
coinShareDevelopment;
        tokenConfigs.coinShareStaking = coinShareStaking;
        tokenConfigs.tokenShareReserve = tokenShareReserve;
        tokenConfigs.buyTax = buyTax;
        tokenConfigs.sellTax = sellTax;
        tokenConfigs.autoSwapTier = autoSwapTier;
    }
```

Recommendation

It is recommended to ensure that the values of `coinShareLP`, `coinShareMarketing`, `coinShareDevelopment`, `coinShareStaking`, and `tokenShareReserve` cannot exceed `100` in total. This can be achieved by adding validation checks within the internal `_setShares` function, to prevent the sum of these variables to be set to values greater than `100`.

RFI - Redundant Function Implementation

Criticality	Minor / Informative
Location	Micropets.sol#L280,433,441,449
Status	Unresolved

Description

The contract contains two distinct internal functions, `_transferToPair` and `_transferFromPair`, which handle token transfers to and from a liquidity pool. Both of these functions call the `_transferWithTax` function, passing different tax parameters based on the direction of the transfer (to or from the pool). This design introduces unnecessary complexity and redundancy, as the contract could streamline its logic by directly invoking the `_transferWithTax` function with the appropriate parameters, eliminating the need for the two extra intermediary functions.

```
        if (isToPool) {
             handleTaxAutomation();
             _transferToPair(from, to, amount);
         } else {
             _transferFromPair(from, to, amount);
         }

   function _transferToPair(
         address sender,
         address recipient,
         uint256 amount
     ) internal {
         _transferWithTax(sender, recipient, amount,
tokenConfigs.sellTax);
     }

     function _transferFromPair(
         address sender,
         address recipient,
         uint256 amount
     ) internal {
         _transferWithTax(sender, recipient, amount,
tokenConfigs.buyTax);
     }

     function _transferWithTax(
     ...)
     { ...
     }
```

Recommendation

It is recommended to refactor the contract to directly call the `_transferWithTax` function with the appropriate tax parameters (`tokenConfigs.sellTax or tokenConfigs.buyTax`) based on the transfer direction. This would simplify the contract's logic, reduce gas overhead, and enhance code maintainability by removing the redundant `_transferToPair` and `_transferFromPair` functions.

FO - Function Optimization

Criticality	Minor / Informative
Location	Micropets.sol#L340
Status	Unresolved

Description

The contract contains two functions named `manualSwapAndLiquify`. Both functions implement the same logic, with the primary distinction being that one of the functions requires the `tokenAmountToSwap` parameter, which determines the amount of tokens to be swapped. This implementation introduces redundancy in the codebase, as two separate functions are being used to achieve a similar outcome.

```
 function manualSwapAndLiquify() external onlyOwner {
        if (!inSwapAndLiquify && !inSplitShares) {
            uint256 contractTokenBalance =
balanceOf(address(this));
            if (contractTokenBalance >=
minimumTokensBeforeSwap) {
                swapAndLiquify(contractTokenBalance);
            }
        }
    }

    function manualSwapAndLiquify(
        uint256 tokenAmountToSwap
    ) external onlyOwner {
        if (!inSwapAndLiquify && !inSplitShares) {
            uint256 contractTokenBalance =
balanceOf(address(this));

            require(
                contractTokenBalance >= tokenAmountToSwap,
                "Invalid amount"
            );

            if (tokenAmountToSwap >= minimumTokensBeforeSwap) {
                swapAndLiquify(tokenAmountToSwap);
            }
        }
    }
```

Recommendation

It is recommended to streamline the code by consolidating these functions into a single function that can handle both use cases. Specifically, the contract could utilize one function that accepts the `tokenAmountToSwap` as an optional parameter. If the parameter is provided, the function could check its validity and proceed with the swap based on the specified amount. If the parameter is not provided, the function can use the default amount to swap. This approach reduces code duplication and enhances the maintainability and clarity of the contract.

RV - Redundant Variable

Criticality	Minor / Informative
Location	Micropets.sol#L72,553,591
Status	Unresolved

Description

The contract contains the `autoSwapTier` variable within the `Configs` struct. This variable can be set through functions like `setShares` or `setAutoSwapTier`. However, the `autoSwapTier` variable is not utilized in any of the contract's functionalities. This makes the variable redundant and can lead to higher gas consumption.

```
    struct Configs {
        ...
        uint24 autoSwapTier;
    }

    function setShares(
        ...
        uint24 autoSwapTier
    ) external onlyOwner {
        _setShares(
            ...
            autoSwapTier
        );
    }

    function setAutoSwapTier(uint24 autoSwapTier) external
onlyOwner {
        tokenConfigs.autoSwapTier = autoSwapTier;
    }
```

Recommendation

It is recommended to remove the `autoSwapTier` variable from the `Configs` struct and any associated functions or logic that sets this variable, such as `setAutoSwapTier`. Keeping unused variables not only increases the gas cost for contract deployment but also makes the codebase less maintainable and more prone to errors in the future. Cleaning

up such redundant elements will streamline the contract, making it more efficient and easier to understand.

PTRP - Potential Transfer Revert Propagation

Criticality	Minor / Informative
Location	Micropets.sol#L312
Status	Unresolved

Description

The contract sends funds to a `to` address 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.

```
(bool sentETH, ) = payable(to).call{value: value}("");
require(sentETH, "Failed to send ETH");
```

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	Micropets.sol#L294,579,360
Status	Unresolved

Description

The contract accumulates tokens from the taxes to swap them for ETH. The variable `minimumTokensBeforeSwap` 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.

```
uint256 contractTokenBalance = balanceOf(address(this));
if (contractTokenBalance >= minimumTokensBeforeSwap) {
    swapAndLiquify(contractTokenBalance);
...

function setMinimumTokensBeforeSwap(
    uint256 _minimumTokensBeforeSwap
) external onlyOwner {
    minimumTokensBeforeSwap = _minimumTokensBeforeSwap;
}

    function manualSwapAndLiquify(
        uint256 tokenAmountToSwap
    ) external onlyOwner {
        ...

            if (tokenAmountToSwap >= minimumTokensBeforeSwap) {
                swapAndLiquify(tokenAmountToSwap);
            }
        }
    }
```

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.

PTAI - Potential Transfer Amount Inconsistency

Criticality	Minor / Informative
Location	Micropets.sol#L674
Status	Unresolved

Description

The `transfer()` and `transferFrom()` functions are used to transfer a specified amount of tokens to an address. The fee or tax is an amount that is charged to the sender of an ERC20 token when tokens are transferred to another address. According to the specification, the transferred amount could potentially be less than the expected amount. This may produce inconsistency between the expected and the actual behavior.

The following example depicts the diversion between the expected and actual amount.

Tax	Amount	Expected	Actual
No Tax	100	100	100
10% Tax	100	100	90

```
   function migrate() external {
        require(migrationRunning, "Migration over");

        uint256 V1Balance =
IERC20(0xA77346760341460B42C230ca6D21d4c8E743Fa9c)
            .balanceOf(_msgSender());

        require(V1Balance > 0, "Invalid migration");

        safeTransferFrom(
            0xA77346760341460B42C230ca6D21d4c8E743Fa9c,
            _msgSender(),
            address(this),
            V1Balance
        );

        uint256 newTokenAmount = V1Balance.div(migrationRate);
        _transferStandard(migrationVault, _msgSender(),
newTokenAmount);
    }
```

Recommendation

The team is advised to take into consideration the actual amount that has been transferred instead of the expected.

It is important to note that an ERC20 transfer tax is not a standard feature of the ERC20 specification, and it is not universally implemented by all ERC20 contracts. Therefore, the contract could produce the actual amount by calculating the difference between the transfer call.

```
 Actual Transferred Amount = Balance After Transfer - Balance
Before Transfer
```

MEE - Missing Events Emission

Criticality	Minor / Informative
Location	Micropets.sol#L471
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.

```
    function includeInFee(address account) external onlyOwner {
        excludedFromFee[account] = false;
    }

    function excludeFromFee(address account) external onlyOwner
{
        excludedFromFee[account] = true;
    }
```

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.

RSML - Redundant SafeMath Library

Criticality	Minor / Informative
Location	Micropets.sol
Status	Unresolved

Description

SafeMath is a popular Solidity library that provides a set of functions for performing common arithmetic operations in a way that is resistant to integer overflows and underflows.

Starting with Solidity versions that are greater than or equal to 0.8.0, the arithmetic operations revert to underflow and overflow. As a result, the native functionality of the Solidity operations replaces the SafeMath library. Hence, the usage of the SafeMath library adds complexity, overhead and increases gas consumption unnecessarily.

```
library SafeMath {...}
```

Recommendation

The team is advised to remove the SafeMath library. Since the version of the contract is greater than `0.8.0` then the pure Solidity arithmetic operations produce the same result.

If the previous functionality is required, then the contract could exploit the `unchecked { ... }` statement.

Read more about the breaking change on https://docs.soliditylang.org/en/v0.8.16/080-breaking-changes.html#solidity-v0-8-0-breaking-changes.

L02 - State Variables could be Declared Constant

Criticality	Minor / Informative
Location	Micropets.sol#L47
Status	Unresolved

Description

State variables can be declared as constant using the constant keyword. This means that the value of the state variable cannot be changed after it has been set. Additionally, the constant variables decrease gas consumption of the corresponding transaction.

```
ISwapRouter public swapRouter
```

Recommendation

Constant state variables can be useful when the contract wants to ensure that the value of a state variable cannot be changed by any function in the contract. This can be useful for storing values that are important to the contract's behavior, such as the contract's address or the maximum number of times a certain function can be called. The team is advised to add the constant keyword to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	Micropets.sol#L504,510,516,522,528,580,586,595,600
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 payable _marketingAddress
address payable _developmentAddress
address payable _vaultAddress
address payable _coinStakingAddress
address _tokenReserveAddress
uint256 _minimumTokensBeforeSwap
uint256 _minimumETHToTransfer
bool _enabled
```

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	Micropets.sol#L582,588,609,648
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.

```
minimumTokensBeforeSwap = _minimumTokensBeforeSwap
minimumETHToTransfer = _minimumETHToTransfer
migrationRate = newRate
uniswapV2Router = _newPancakeRouter
```

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	Micropets.sol#L315
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 safeTransferToken(address token, address to, uint
value) internal {
        (bool success, bytes memory data) = token.call(
            abi.encodeWithSelector(0xa9059cbb, to, value)
        );
        require(
            success && (data.length == 0 || abi.decode(data,
(bool))),
            "Failed to send token"
        );
    }
```

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.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	Micropets.sol#L17
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 ^0.8.16;
```

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.

L20 - Succeeded Transfer Check

Criticality	Minor / Informative
Location	Micropets.sol#L696
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.

```
V1.transfer(receiver, V1.balanceOf(address(this)))
```

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		
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		

	div	Internal		
	mod	Internal		
	mod	Internal		
Address	Library			
	isContract	Internal		
	sendValue	Internal	✓	
	functionCall	Internal	✓	
	functionCall	Internal	✓	
	functionCallWithValue	Internal	✓	
	functionCallWithValue	Internal	✓	
	_functionCallWithValue	Private	✓	
Ownable	Implementation	Context		
	_setOwner	Internal	✓	
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
IUniswapV3Factory	Interface			
	owner	External		-
	feeAmountTickSpacing	External		-

	getPool	External		-
	createPool	External	✓	-
	setOwner	External	✓	-
	enableFeeAmount	External	✓	-
IUniswapV3SwapCallback	Interface			
	uniswapV3SwapCallback	External	✓	-
IMulticall	Interface			
	multicall	External	Payable	-
ISwapRouter	Interface	IUniswapV3SwapCallback, IMulticall		
	exactInputSingle	External	Payable	-
	exactInput	External	Payable	-
	exactOutputSingle	External	Payable	-
	exactOutput	External	Payable	-
IUniswapV3PoolActions	Interface			
	initialize	External	✓	-
	mint	External	✓	-
	collect	External	✓	-
	burn	External	✓	-

	swap	External	✓	-
	flash	External	✓	-
	increaseObservationCardinalityNext	External	✓	-
IUniswapV3PoolDerivedState	Interface			
	observe	External		-
	snapshotCumulativesInside	External		-
IUniswapV3PoolEvents	Interface			
IUniswapV3PoolImmutables	Interface			
	factory	External		-
	token0	External		-
	token1	External		-
	fee	External		-
	tickSpacing	External		-
	maxLiquidityPerTick	External		-
IUniswapV3PoolOwnerActions	Interface			
	setFeeProtocol	External	✓	-
	collectProtocol	External	✓	-

IUniswapV3PoolState	Interface			
	slot0	External		-
	feeGrowthGlobal0X128	External		-
	feeGrowthGlobal1X128	External		-
	protocolFees	External		-
	liquidity	External		-
	ticks	External		-
	tickBitmap	External		-
	positions	External		-
	observations	External		-
IUniswapV3Pool	Interface	IUniswapV3PoolImmutables, IUniswapV3PoolState, IUniswapV3PoolDerivedState, IUniswapV3PoolActions, IUniswapV3PoolOwnerActions, IUniswapV3PoolEvents		
MigrationVAult	Implementation			
IUniswapV2Factory	Interface			
	feeTo	External		-
	feeToSetter	External		-

	getPair	External		-
	allPairs	External		-
	allPairsLength	External		-
	createPair	External	✓	-
	setFeeTo	External	✓	-
	setFeeToSetter	External	✓	-
IUniswapV2Pair	Interface			
	name	External		-
	symbol	External		-
	decimals	External		-
	totalSupply	External		-
	balanceOf	External		-
	allowance	External		-
	approve	External	✓	-
	transfer	External	✓	-
	transferFrom	External	✓	-
	DOMAIN_SEPARATOR	External		-
	PERMIT_TYPEHASH	External		-
	nonces	External		-
	permit	External	✓	-
	MINIMUM_LIQUIDITY	External		-
	factory	External		-

	token0	External		-
	token1	External		-
	getReserves	External		-
	price0CumulativeLast	External		-
	price1CumulativeLast	External		-
	kLast	External		-
	burn	External	✓	-
	swap	External	✓	-
	skim	External	✓	-
	sync	External	✓	-
	initialize	External	✓	-
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	IUniswapV2 Router01		
	removeLiquidityETHSupportingFeeOnTransferTokens	External	✓	-
	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External	✓	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
MicroPets	Implementation	Context, IERC20, Ownable		
	initialize	Public	✓	-
	name	External		-
	symbol	External		-

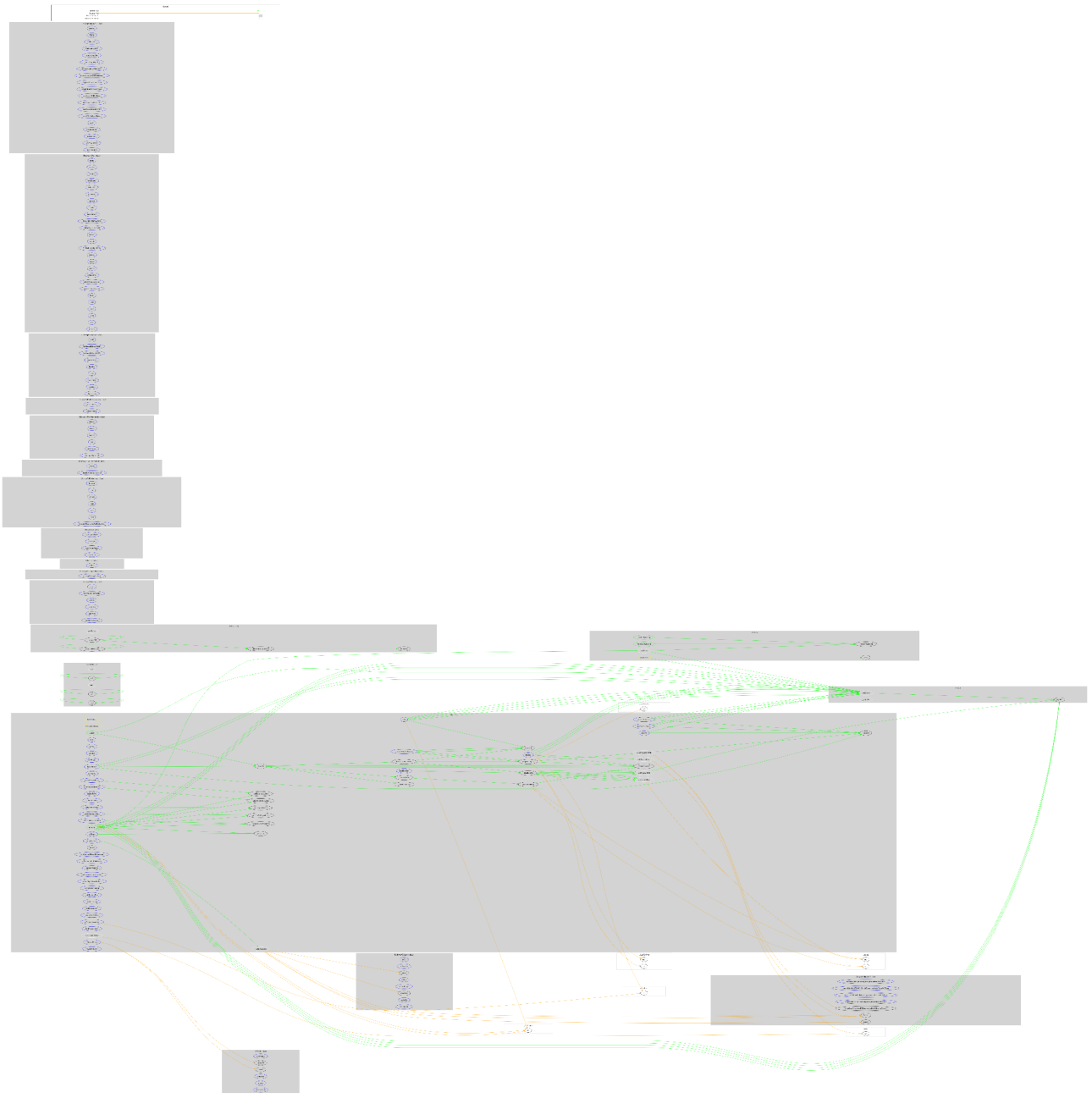
	decimals	External		-
	totalSupply	External		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
	increaseAllowance	External	✓	-
	decreaseAllowance	External	✓	-
	_approve	Private	✓	
	_transfer	Private	✓	
	handleTaxAutomation	Internal	✓	
	safeTransferETH	Internal	✓	
	safeTransferToken	Internal	✓	
	safeTransferFrom	Internal	✓	
	manualSwapAndLiquify	External	✓	onlyOwner
	manualSwapAndLiquify	External	✓	onlyOwner
	swapTokensForETH	Internal	✓	
	swapAndLiquify	Internal	✓	lockForSwap
	_calcuclateShare	Internal		
	_distributeTax	Internal	✓	lockForSplitShare
	distributeTax	External	✓	onlyOwner
	_transferStandard	Internal	✓	

	_transferToPair	Internal	✓	
	_transferFromPair	Internal	✓	
	_transferWithTax	Internal	✓	
	includeInFee	External	✓	onlyOwner
	excludeFromFee	External	✓	onlyOwner
	_setMarketingAddress	Internal	✓	
	_setDevelopmentAddress	Internal	✓	
	_setLpVaultAddress	Internal	✓	
	_setCoinStakingAddress	Internal	✓	
	_setTokenReserveAddress	Internal	✓	
	setMarketingAddress	External	✓	onlyOwner
	setDevelopmentAddress	External	✓	onlyOwner
	setLpVaultAddress	External	✓	onlyOwner
	setCoinStakingAddress	External	✓	onlyOwner
	setTokenReserveAddress	External	✓	onlyOwner
	_setShares	Internal	✓	
	setShares	External	✓	onlyOwner
	getTax	External		-
	setMinimumTokensBeforeSwap	External	✓	onlyOwner
	setMinimumETHToTransfer	External	✓	onlyOwner
	setAutoSwapTier	External	✓	onlyOwner
	setSwapAndLiquifyEnabled	External	✓	onlyOwner
	setAutoSplitSharesEnables	External	✓	onlyOwner

	setMigrationRunning	External	✓	onlyOwner
	setMigrationRate	External	✓	onlyOwner
	enableUniswap	External	✓	onlyOwner
	addPoolAddress	External	✓	onlyOwner
	removePoolAddress	External	✓	onlyOwner
	_setupExchange	Internal	✓	
	setupExchange	External	✓	onlyOwner
	totalTaxCollected	External		onlyOwner
	burn	External	✓	-
	getMigrationAmount	External		-
	migrate	External	✓	-
	retrieveOldPets	External	✓	onlyOwner
		External	Payable	-

Inheritance Graph

Flow Graph



Summary

MicroPets contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. There are some functions that can be abused by the owner like manipulate the fees. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats.

Initial Audit, 30 Oct 2023

At the time of the audit report, the contract with address

0xd7e8221ABD7B8010f68B8a79FF3422fB9D794f98 is pointed out by the following proxy

address: 0x2466858ab5edAd0BB597FE9f008F568B00d25Fe3.

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