Cyberscope

Audit Report

MollarsToken

June 2024

Network	ETH
Address	0x385d65Ed9241E415cFC689C3e0BCf5aB2f0505c2
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
●	EPC	Existing Pair Creation	Unresolved
●	PMRM	Potential Mocked Router Manipulation	Unresolved
●	RSML	Redundant SafeMath Library	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved

Table of Contents

Review

Contract Name	MOLLARS
Compiler Version	v0.8.17+commit.8df45f5f
Optimization	200 runs
Explorer	https://etherscan.io/address/0x385d65ed9241e415cfc689c3e0bcf5ab2f0505c2
Address	0x385d65ed9241e415cfc689c3e0bcf5ab2f0505c2
Network	ETH
Symbol	MOLLARS
Decimals	9
Total Supply	10,000,000
Badge Eligibility	Yes

Audit Updates

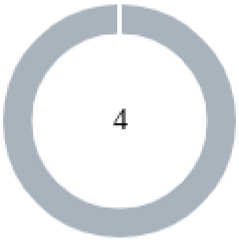
Initial Audit	15 Jan 2024 https://github.com/cyberscope-io/audits/blob/main/mollars/v1/audit.pdf
Corrected Phase 2	18 Jun 2024

Source Files

Filename	SHA256

MOLLARS.sol	2db6724fed31b5eb7f67b16df840471059b8116de599e95ecc2698bba523373c

Findings Breakdown



●	Critical	0
●	Medium	0
●	Minor / Informative	4

	Severity	Unresolved	Acknowledged	Resolved	Other
●	Critical	0	0	0	0
●	Medium	0	0	0	0
●	Minor / Informative	4	0	0	0

EPC - Existing Pair Creation

Criticality	Minor / Informative
Location	MOLLARS.sol#L370
Status	Unresolved

Description

The contract contains a function that does not handle the scenario where a pair already exists prior to its execution. If a pair for the given tokens has already been established, the `createPair` function will revert and not proceed with the creation of a new pair. As a result, if a pair has been previously set up before the function is invoked, the contract will encounter an error when trying to call the `createPair` function. This will prevent the successful execution, essentially leading the function to revert.

```
    function addLp(address _router) external payable onlyOwner {
        router = IDexRouter(_router);
        pair = IDexFactory(router.factory()).createPair(
            address(this),
            router.WETH()
        );
        _allowances[address(this)][address(router)] =
_totalSupply;
        router.addLiquidityETH{value: msg.value}(
            address(this),
            balanceOf(address(this)),
            0,
            0,
            owner(),
            block.timestamp
        );
        IERC20Extended(pair).approve(address(router),
type(uint256).max);
    }
```

Recommendation

To mitigate the risks associated with attempting to create an already existing pair, it is recommended to implement a check to determine whether the pair already exists before proceeding to create a new pair. This can be achieved by utilizing the getPair function of the

Factory contract to retrieve the address of the pair contract for the specified tokens. If the address returned by the getPair function is the zero address, it indicates that the pair does not exist, and the contract can proceed with the createPair function. Conversely, if a non-zero address is returned, it indicates that the pair already exists, and the createPair function will revert.

PMRM - Potential Mocked Router Manipulation

Criticality	Minor / Informative
Location	MOLLARS.sol#L374
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.

```
    function addLp(address _router) external payable onlyOwner {
        router = IDexRouter(_router);
        pair = IDexFactory(router.factory()).createPair(
            address(this),
            router.WETH()
        );
        _allowances[address(this)][address(router)] = _totalSupply;
        router.addLiquidityETH{value: msg.value}(
            address(this),
            balanceOf(address(this)),
            0,
            0,
            owner(),
            block.timestamp
        );
        IERC20Extended(pair).approve(address(router),
type(uint256).max);
    }
```

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.

RSML - Redundant SafeMath Library

Criticality	Minor / Informative
Location	MOLLARS.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 in cases where the explanatory error message is not used.

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

Recommendation

The team is advised to remove the SafeMath library in cases where the revert error message is not used. 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.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	MOLLARS.sol#L141,262,263,264,265,370
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);
string private constant _name = "MollarsToken"
string private constant _symbol = "MOLLARS"
uint8 private constant _decimals = 9
uint256 private constant _totalSupply = 10_000_000 *
10**_decimals
address _router
```

Recommendation

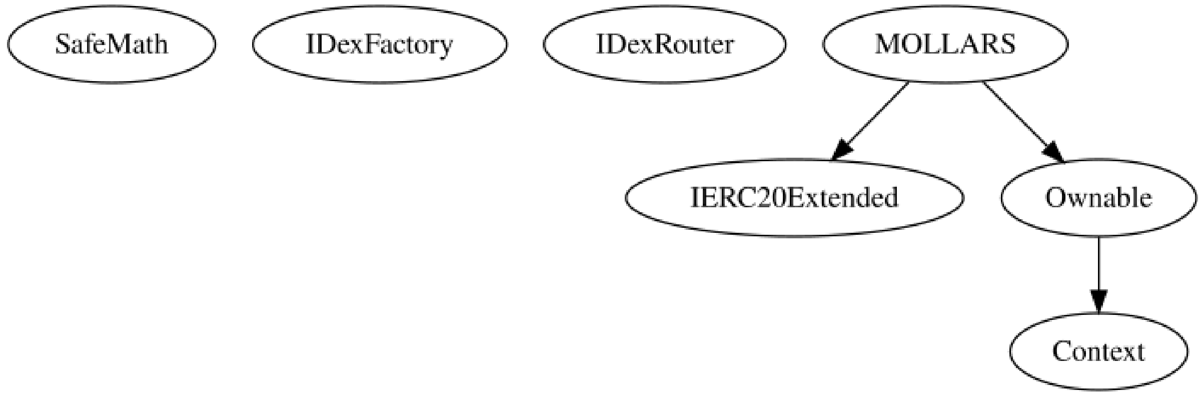
By following the Solidity naming convention guidelines, the codebase increased the readability, maintainability, and makes it easier to work with.

Find more information on the Solidity documentation https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-convention.

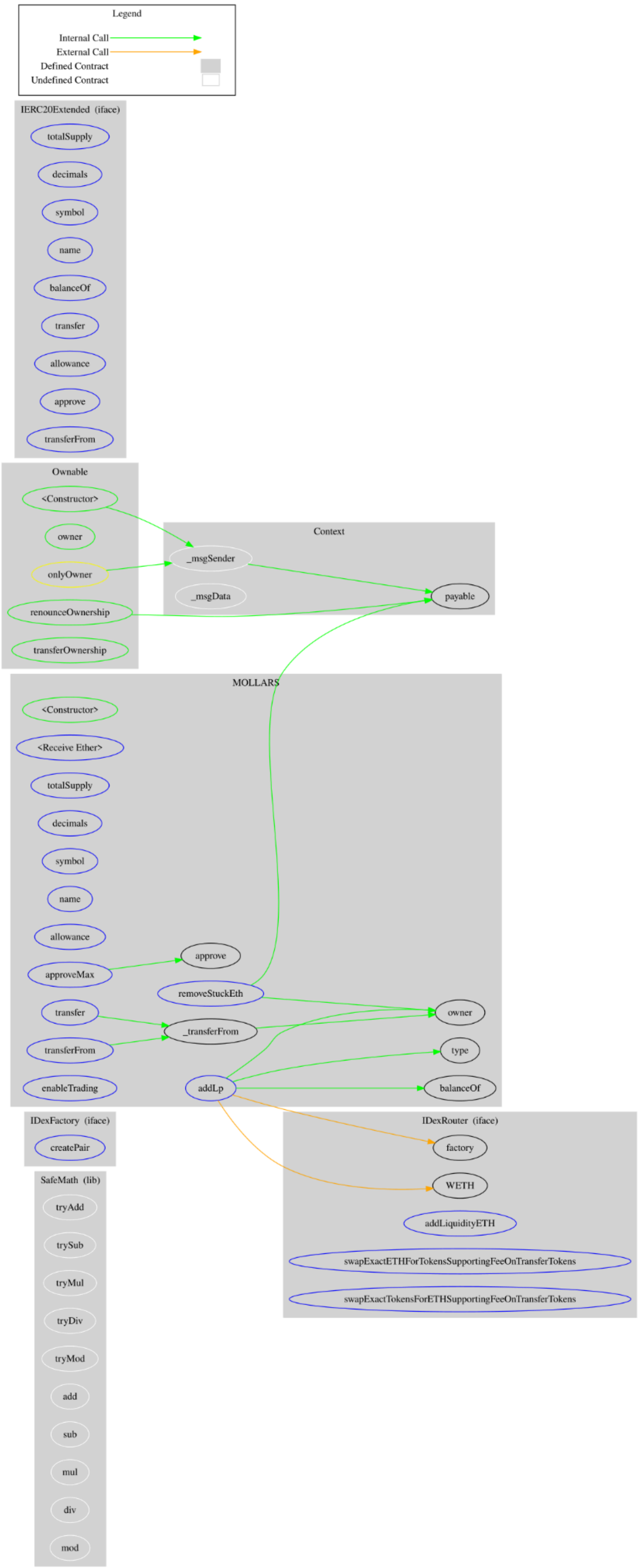
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
MOLLARS	Implementation	IERC20Extended, Ownable		
		Public	✓	Ownable
		External	Payable	-
	totalSupply	External		-
	decimals	External		-
	symbol	External		-
	name	External		-
	balanceOf	Public		-
	allowance	External		-
	approve	Public	✓	-
	approveMax	External	✓	-
	transfer	External	✓	-
	transferFrom	External	✓	-
	_transferFrom	Internal	✓	
	addLp	External	Payable	onlyOwner
	enableTrading	External	✓	onlyOwner
	removeStuckEth	External	✓	onlyOwner

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IERC20Extended (iface)
totalSupply
decimals
symbol
name
balanceOf
transfer
allowance
approve
transferFrom
Ownable
<Constructor>
owner
onlyOwner
renounceOwnership
transferOwnership
Context
_msgSender
_msgData
payable
MOLLARS
<Constructor>
<Receive Ether>
totalSupply
decimals
symbol
name
allowance
approveMax
approve
removeStuckEth
transfer
_transferFrom
transferFrom
enableTrading
addLp
owner
type
balanceOf
IDexFactory (iface)
createPair
IDexRouter (iface)
factory
WETH
addLiquidityETH
swapExactETHForTokensSupportingFeeOnTransferTokens
swapExactTokensForETHSupportingFeeOnTransferTokens
SafeMath (lib)
tryAdd
trySub
tryMul
tryDiv
tryMod
add
sub
mul
div
mod

Summary

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

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