Cyberscope

Audit Report
Marley

March 2023

Type	ERC20
Network	ETH
Address	0xAAa54a497Fe6DD94D22084DDBEE92E66CEb35AEC
Audited by	© cyberscope

Table of Contents

Review

Contract Name	Marley
Compiler Version	v0.8.19+commit.7dd6d404
Optimization	200 runs
Explorer	https://etherscan.io/address/0xaaa54a497fe6dd94d22084ddbee92e66ceb35aec
Address	0xAAa54a497Fe6DD94D22084DDBEE92E66CEb35AEC
Network	ETH
Symbol	MARLEY
Decimals	18
Total Supply	10,000,000,000

Audit Updates

Initial Audit	13 Mar 2023 https://github.com/cyberscope-io/audits/tree/main/1-mrly/v1/audit.pdf
Corrected Phase 2	14 Mar 2023

Source Files

Filename	SHA256
MARLEY.sol	b2261ce6ca9e447c7118ebcdea88d4222c112b9f7ec1387a22806aaa510748de

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	Unresolved
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

ULTW - Transfers Liquidity to Team Wallet

Criticality	Minor / Informative
Location	Marley.sol#L667
Status	Unresolved

Description

The tokens that were going to be sent to the liquidity pool via transaction tax can be removed by the team by using the `withdrawStuckETH` ETH function.

```
function withdrawStuckETH() external onlyOwner {
  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. Some suggestions are:

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-sign wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.
- Renouncing the ownership will eliminate the threats but it is non-reversible.

Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	DDP	Decimal Division Precision	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L13	Divide before Multiply Operation	Unresolved
●	L15	Local Scope Variable Shadowing	Unresolved

DDP - Decimal Division Precision

Criticality	Minor / Informative
Location	Marley.sol#L563,571
Status	Unresolved

Description

Division of decimal (fixed point) numbers can result in rounding errors due to the way that division is implemented in Solidity. Thus, it may produce issues with precise calculations with decimal numbers.

Solidity represents decimal numbers as integers, with the decimal point implied by the number of decimal places specified in the type (e.g. decimal with 18 decimal places). When a division is performed with decimal numbers, the result is also represented as an integer, with the decimal point implied by the number of decimal places in the type. This can lead to rounding errors, as the result may not be able to be accurately represented as an integer with the specified number of decimal places.

Hence, the splitted shares will not have the exact precision and some funds may not be calculated as expected.

The splitted variable `tokensForLiquidity` and `tokensForMarketing` will not have the exact precision.

```
fees = amount * sellTotalFees /100;
tokensForLiquidity += fees * sellLiquidityFee / sellTotalFees;
tokensForMarketing += fees * sellMarketingFee / sellTotalFees;
fees = amount * buyTotalFees / 100;
tokensForLiquidity += fees * buyLiquidityFee / buyTotalFees;
tokensForMarketing += fees * buyMarketingFee / buyTotalFees;
```

Recommendation

The contract could calculate the subtraction of the divided funds in the last calculation in order to avoid the division rounding issue.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	MARLEY.sol#L273,348,487,495,655,668
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);
mapping (address => bool) public _isExcludedMaxTransactionAmount
uint256 _marketingFee
uint256 _liquidityFee
uint256 _burnFee
address _token
address _to
address _marketingAddress
```

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	MARLEY.sol#L458,488,496
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 * (10**18)
buyMarketingFee = _marketingFee
sellMarketingFee = _marketingFee
```

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	MARLEY.sol#L560,561,567,568,569
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 / sellTotalFees
fees = amount * buyTotalFees / 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.

L15 - Local Scope Variable Shadowing

Criticality	Minor / Informative
Location	MARLEY.sol#L394
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 = 10000000000 * 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.

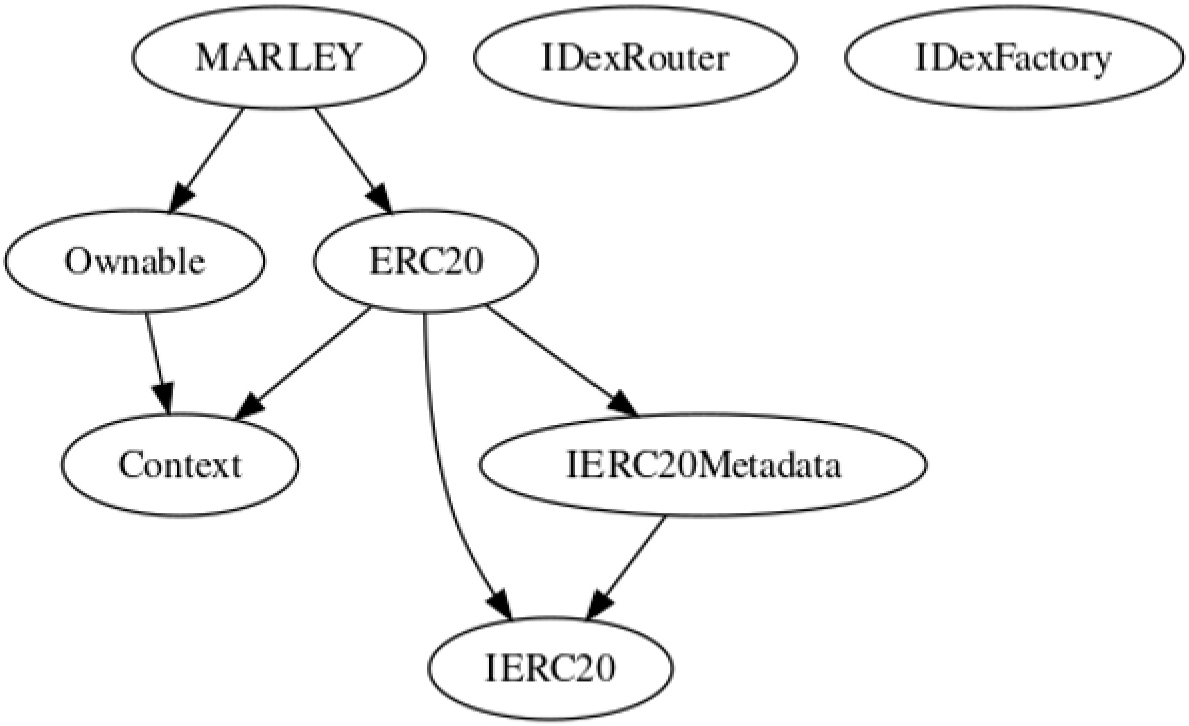
Functions Analysis

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		-

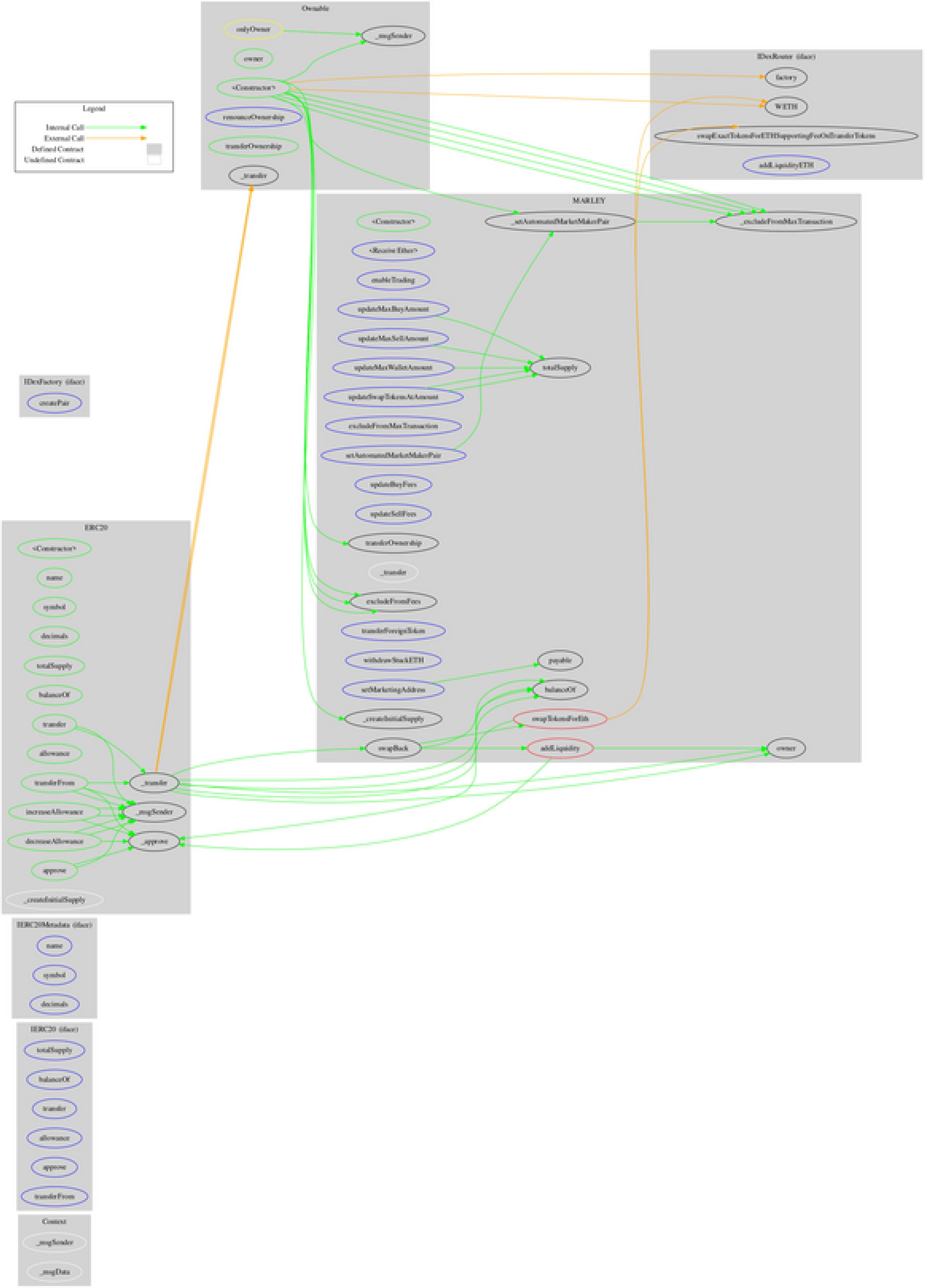
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	_transfer	Internal	✓	
	_createInitialSupply	Internal	✓	
	_approve	Internal	✓	
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	renounceOwnership	External	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
IDexRouter	Interface			
	factory	External		-
	WETH	External		-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
	addLiquidityETH	External	Payable	-
IDexFactory	Interface			
	createPair	External	✓	-
MARLEY	Implementation	ERC20, Ownable		
		Public	✓	ERC20

		External	Payable	-
	enableTrading	External	✓	onlyOwner
	updateMaxBuyAmount	External	✓	onlyOwner
	updateMaxSellAmount	External	✓	onlyOwner
	updateMaxWalletAmount	External	✓	onlyOwner
	updateSwapTokensAtAmount	External	✓	onlyOwner
	_excludeFromMaxTransaction	Private	✓	
	excludeFromMaxTransaction	External	✓	onlyOwner
	setAutomatedMarketMakerPair	External	✓	onlyOwner
	_setAutomatedMarketMakerPair	Private	✓	
	updateBuyFees	External	✓	onlyOwner
	updateSellFees	External	✓	onlyOwner
	excludeFromFees	Public	✓	onlyOwner
	_transfer	Internal	✓	
	swapTokensForEth	Private	✓	
	addLiquidity	Private	✓	
	swapBack	Private	✓	
	transferForeignToken	External	✓	onlyOwner
	withdrawStuckETH	External	✓	onlyOwner
	setMarketingAddress	External	✓	onlyOwner

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
Ownable
onlyOwner
_msgSender
owner
<Constructor>
renounceOwnership
transferOwnership
_transfer
IDexRouter (iface)
factory
WETH
swapExactTokensForETHSupportingFeeOnTransferTokens
addLiquidityETH
MARLEY
<Constructor>
_setAutomatedMarketMakerPair
_excludeFromMaxTransaction
<Receive Ether>
enableTrading
updateMaxBuyAmount
updateMaxSellAmount
updateMaxWalletAmount
totalSupply
updateSwapTokensAtAmount
excludeFromMaxTransaction
setAutomatedMarketMakerPair
updateBuyFees
updateSellFees
transferOwnership
_transfer
excludeFromFees
transferForeignToken
withdrawStuckETH
payable
setMarketingAddress
balanceOf
_createInitialSupply
swapTokensForEth
swapBack
addLiquidity
owner
IDexFactory (iface)
createPair
ERC20
<Constructor>
name
symbol
decimals
totalSupply
balanceOf
transfer
allowance
transferFrom
_transfer
increaseAllowance
_msgSender
decreaseAllowance
_approve
approve
_createInitialSupply
IERC20Metadata (iface)
name
symbol
decimals
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
Context
_msgSender
_msgData

Summary

Marley 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 transferring funds to the team's wallet. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats. There is also a limit of max 6% fees.

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