Cyberscope

Audit Report

DeLabs

October 2023

SHA256 43db8ffd57001fbfbbddeeb05701411d867646395c4480f6357df6c5963db07c

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
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L09	Dead Code Elimination	Unresolved
●	L16	Validate Variable Setters	Unresolved
●	L20	Succeeded Transfer Check	Unresolved

Table of Contents

Review

Contract Name	BombShelterInu
Testing Deploy	https://testnet.bscscan.com/address/0x223d466d43c2a40d98f5b8e7d291bd2b349caf52
Symbol	BOOM
Decimals	18
Total Supply	1,000,000,000

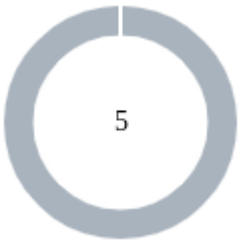
Audit Updates

Initial Audit	17 Oct 2023

Source Files

Filename	SHA256
contracts/BombShelterInu.sol	43db8ffd57001fbfbbddeeb05701411d867646395c4480f6357df6c5963db07c

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

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	contracts/BombShelterInu.sol#L33,113,116,125,126,127,128,129,143,149,154,175,434
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 => uint256) _tOwned
mapping (address => mapping (address => uint256)) _allowances
uint256 constant private startingSupply = 1_000_000_000
string constant private _name = "Bomb Shelter Inu"
string constant private _symbol = "BOOM"
uint8 constant private _decimals = 18
uint256 constant private _tTotal = startingSupply * (10 ** _decimals)

Fees public _taxRates = Fees({
        buyFee: 12,
        sellFee: 88,
        transferFee: 0
    })

...
```

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/BombShelterInu.sol#L414,424,441
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.

```
swapThreshold = (_tTotal * thresholdPercent) / thresholdDivisor
piSwapPercent = priceImpactSwapPercent
cashierGas = gas
```

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	contracts/BombShelterInu.sol#L480
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 _basicTransfer(address from, address to, uint256 amount) internal
returns (bool) {
        _tOwned[from] -= amount;
        _tOwned[to] += amount;
        emit Transfer(from, to, amount);
        return true;
    }
```

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.

L16 - Validate Variable Setters

Criticality	Minor / Informative
Location	contracts/BombShelterInu.sol#L263,357
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.

```
operator = newOperator
lpPair = constructorLP
```

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.

L20 - Succeeded Transfer Check

Criticality	Minor / Informative
Location	contracts/BombShelterInu.sol#L681
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(_owner, TOKEN.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**
IERC20	Interface			
	totalSupply	External		-
	decimals	External		-
	symbol	External		-
	name	External		-
	getOwner	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IFactoryV2	Interface			
	getPair	External		-
	createPair	External	✓	-
IV2Pair	Interface			
	factory	External		-

	getReserves	External		-
	sync	External	✓	-
IRouter01	Interface			
	factory	External		-
	WETH	External		-
	addLiquidityETH	External	Payable	-
	addLiquidity	External	✓	-
	swapExactETHForTokens	External	Payable	-
	getAmountsOut	External		-
	getAmountsIn	External		-
IRouter02	Interface	IRouter01		
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactTokensForTokens	External	✓	-
Initializer	Interface			
	setLaunch	External	✓	-
	getConfig	External	✓	-
	getInits	External	✓	-
	setLpPair	External	✓	-

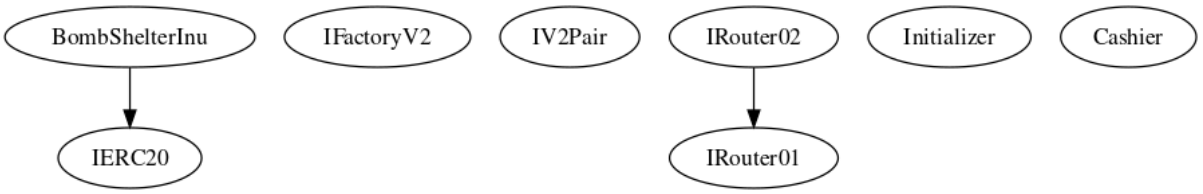
Cashier	Interface			
	setRewardsProperties	External	✓	-
	tally	External	✓	-
	load	External	Payable	-
	cashout	External	✓	-
	giveMeWelfarePlease	External	✓	-
	getTotalDistributed	External		-
	getUserInfo	External		-
	getUserRealizedRewards	External		-
	getPendingRewards	External		-
	initialize	External	✓	-
	getCurrentReward	External		-
BombShelterInu	Implementation	IERC20		
		Public	Payable	-
	transferOwner	External	✓	onlyOwner
	renounceOwnership	External	✓	onlyOwner
	setOperator	Public	✓	-
	renounceOriginalDeployer	External	✓	-
		External	Payable	-
	totalSupply	External		-
	decimals	External		-

	symbol	External		-
	name	External		-
	getOwner	External		-
	balanceOf	Public		-
	allowance	External		-
	approve	External	✓	-
	_approve	Internal	✓	
	approveContractContingency	Public	✓	onlyOwner
	transfer	External	✓	-
	transferFrom	External	✓	-
	setNewRouter	External	✓	onlyOwner
	setLpPair	External	✓	onlyOwner
	setInitializers	Public	✓	onlyOwner
	isExcludedFromFees	External		-
	isExcludedFromDividends	External		-
	isExcludedFromProtection	External		-
	isExcludedFromLimits	External		-
	setExcludedFromLimits	External	✓	onlyOwner
	setDividendExcluded	Public	✓	onlyOwner
	setExcludedFromFees	Public	✓	onlyOwner
	setExcludedFromProtection	External	✓	onlyOwner
	setWallets	External	✓	onlyOwner
	getTokenAmountAtPriceImpact	External		-

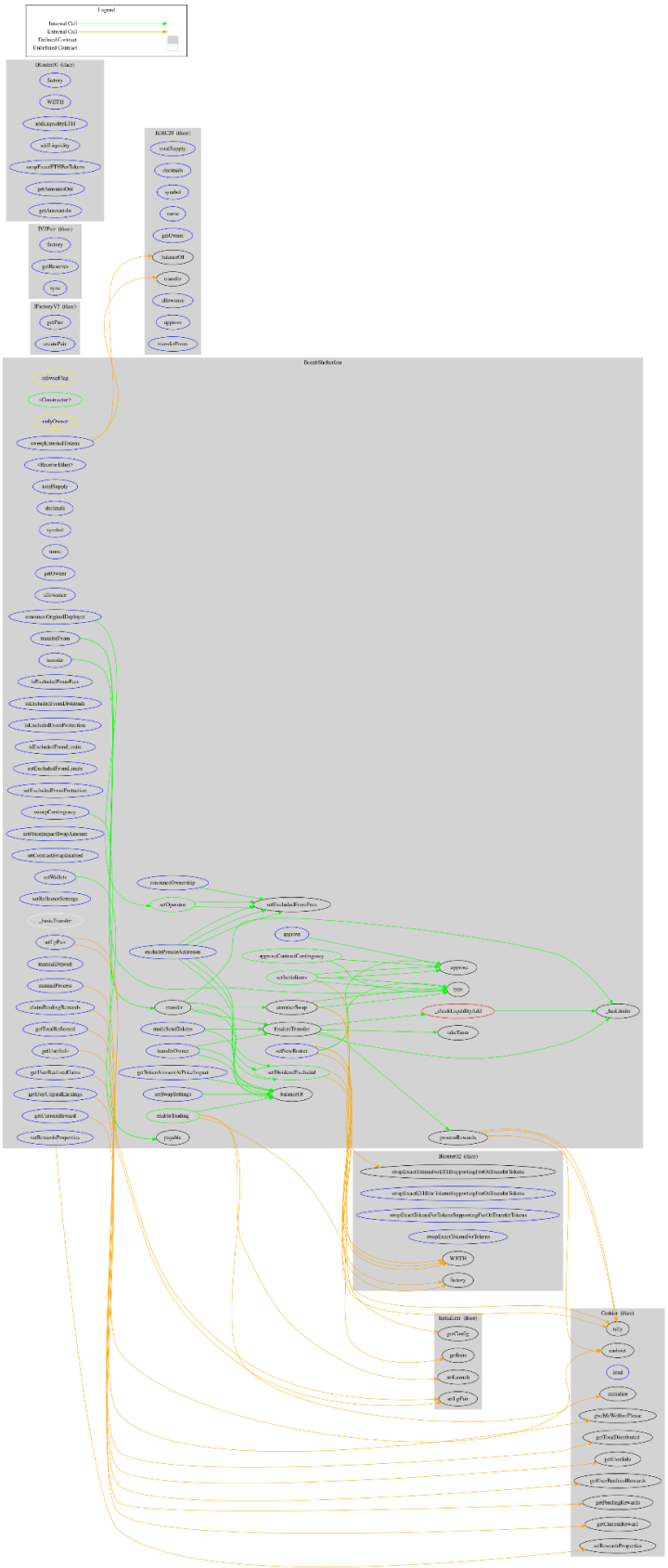
	setSwapSettings	External	✓	onlyOwner
	setPriceImpactSwapAmount	External	✓	onlyOwner
	setContractSwapEnabled	External	✓	onlyOwner
	setRewardsProperties	External	✓	onlyOwner
	setReflectorSettings	External	✓	onlyOwner
	excludePresaleAddresses	External	✓	onlyOwner
	_hasLimits	Internal		
	_basicTransfer	Internal	✓	
	_transfer	Internal	✓	
	contractSwap	Internal	✓	inSwapFlag
	_checkLiquidityAdd	Private	✓	
	enableTrading	Public	✓	onlyOwner
	finalizeTransfer	Internal	✓	
	processRewards	Internal	✓	
	manualProcess	External	✓	-
	takeTaxes	Internal	✓	
	multiSendTokens	External	✓	onlyOwner
	manualDeposit	External	✓	onlyOwner
	sweepContingency	External	✓	onlyOwner
	sweepExternalTokens	External	✓	onlyOwner
	claimPendingRewards	External	✓	-
	getTotalReflected	External		-
	getUserInfo	External		-

	getUserRealizedGains	External		-
	getUserUnpaidEarnings	External		-
	getCurrentReward	External		-

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IRouter01 (iface)
factory
WETH
addLiquidityETH
addLiquidity
swapExactTokensForTokens
getAmountOut
getAmountsIn
IUniswapV2Pair (iface)
factory
getReserves
sync
IFactoryV2 (iface)
getPair
createPair
IERC20 (iface)
totalSupply
decimals
symbol
name
getOwner
balanceOf
transfer
allowance
approve
transferFrom
BoomShiberInu
isInternalTx
<Constructor>
onlyOwner
recoverExternalTokens
<Receive Ether>
totalSupply
decimals
symbol
name
getOwner
allowance
renounceOriginalDeployer
transferFrom
transfer
isExcludedFromFees
isExcludedFromDividends
isExcludedFromProtection
isExcludedFromLimits
setExcludedFromLimits
setExcludedFromProtection
swapContingency
setMinimumSwapAmount
setContractSwapEnabled
setWallets
setRouterSettings
_basicTransfer
setLPPair
manualDeposit
manualProcess
claimPendingRewards
getTotalReflected
getUserInfo
getUserRealizedGains
getUserUnpaidEarnings
getCurrentReward
setRewardsProperties
renounceOwnership
setOperator
setExcludedFromFees
approve
excludePresaleAddresses
approveContractContingency
setInitializers
open
transfer
manualSendTokens
transferOwner
getTokenAmountAtPriceImpact
setSwapSettings
enableTrading
payMe
contractSwap
finalizeTransfer
setNewRouter
setDividendPayload
balanceOf
_checkLiquidityAdd
takeTaxes
_basicAssets
processRewards
IRouter02 (iface)
swapExactTokensForETHSupportingFeeOnTransferTokens
swapExactETHForTokensSupportingFeeOnTransferTokens
swapExactTokensForTokensSupportingFeeOnTransferTokens
swapExactTokensForTokens
WETH
factory
InitializeLP (iface)
getConfig
getInfo
setLaunch
setLPPair
Caller (iface)
fully
enabled
total
initialize
getMyWalletPayout
getTotalDistributed
getUserInfo
getUserRealizedRewards
getPendingRewards
getCurrentReward
setRewardsProperties

Summary

DeLabs contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. DeLabs 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 fees are locked at 0,12% for buys and 0,88% for sales.

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