Cyberscope

Audit Report

BearStreetCoin

December 2023

Network	ETH
Address	0x7465308AD5D0C4CE63A743Fe18E92ECCcf504BBB
Audited by	© cyberscope

Analysis

● Critical ● Medium ● Minor / Informative ● Pass

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

Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	PISA	Potential Insufficient Swap Amount	Unresolved
●	FPP	Function Public Permissions	Unresolved
●	PAV	Pair Address Validation	Unresolved
●	MEM	Misleading Error Messages	Unresolved
●	MAU	Misleading Address Usage	Unresolved
●	RVA	Redundant Variable Assignment	Unresolved
●	PTRP	Potential Transfer Revert Propagation	Unresolved
●	MEE	Missing Events Emission	Unresolved
●	PVC	Price Volatility Concern	Unresolved
●	MEM	Misleading Error Messages	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L16	Validate Variable Setters	Unresolved
●	L19	Stable Compiler Version	Unresolved

Table of Contents

Review

Contract Name	DeflationaryToken
Compiler Version	v0.8.19+commit.7dd6d404
Optimization	100 runs
Explorer	https://etherscan.io/address/0x7465308ad5d0c4ce63a743fe18e92ecccf504bbb
Address	0x7465308ad5d0c4ce63a743fe18e92ecccf504bbb
Network	ETH
Symbol	BSC
Decimals	18
Total Supply	66,666,666,666

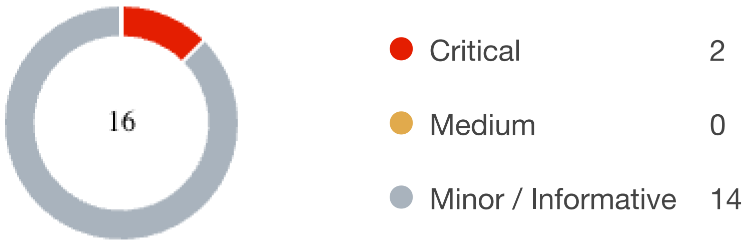
Audit Updates

Initial Audit	15 Dec 2023

Source Files

Filename	SHA256
DeflationaryToken.sol	fa301a0faba37656e2e3851926745c62338e5f5f443fd7ca5cc7f4a3cf644b0e
core/Ownable.sol	8b1a26aa8993faea913d3f026312b3246326ae89f8dfa5a429afcef8f9e857f3
core/Initializable.sol	2df168bd0a87cc28ff791f4fb18ba292a90753b2533159063b1d773d75c5e3fa
core/ERC20.sol	8c68f3fb349e6b98e0499da2332cece629140091cf54180e0f647cf13c0cf8e6
core/libraries/Address.sol	95c8f67fc4fcbab3d7c32f59bcf04712b710cdd3796ddf1bab2ea52bc077aa03
core/interfaces/IERC20Metadata.sol	d064959b77c1744d19b57cc3f4986db65ffce35e73735c6016fcc18bfbabae58
core/interfaces/IERC20.sol	de47eede1ac3f02bd2d27dc2c6833ee0a821802f4ec72d79b93593c3963c9516
core/interfaces/uniswap/IUniswapV2Router02.sol	80064730c808a4ed64181723f7dfc7abd14990067425244d8bc79c40a436e55d
core/interfaces/uniswap/IUniswapV2Factory.sol	499179a9cec50a91715cd867d4e4ebdb43a9f55441d010f0955a4aceca859ae8

Findings Breakdown



Severity		Unresolved	Acknowledged	Resolved	Other
●	Critical	2	0	0	0
●	Medium	0	0	0	0
●	Minor / Informative	14	0	0	0

ST - Stops Transactions

Criticality	Critical
Location	DeflationaryToken.sol#L232,236
Status	Unresolved

Description

The contract owner has the authority to stop the transactions for all users excluding the authorized addresses. The owner may take advantage of it by setting the `maxTxAmount` to zero.

```
if (!_isExcludedFromMaxTransactionLimit[to] &&
!_isExcludedFromMaxTransactionLimit[from]) {
    require(amount <= maxTxAmount, "TOKEN: Buy amount exceeds the
maxTxBuyAmount.");
}
```

The contract owner has the authority to stop the sales for all users excluding the authorized addresses. The owner may take advantage of it by setting the `maxWalletAmount` to zero. As a result, the contract may operate as a honeypot.

```
if (!_isExcludedFromMaxWalletLimit[to]) {
    require(
        (balanceOf(to) + amount) <= maxWalletAmount,
        "TOKEN: Expected wallet amount exceeds the maxWalletAmount."
    );
}
```

Additionally, the contract may operate as a honeypot due to the issues that are described in detail in sections PISA, PTRP, and PVC.

Recommendation

The contract could embody a check for not allowing setting the `maxTxAmount` and `maxWalletAmount` less than a reasonable amount. A suggested implementation could check that the minimum amount should be more than a fixed percentage of the total supply. 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:

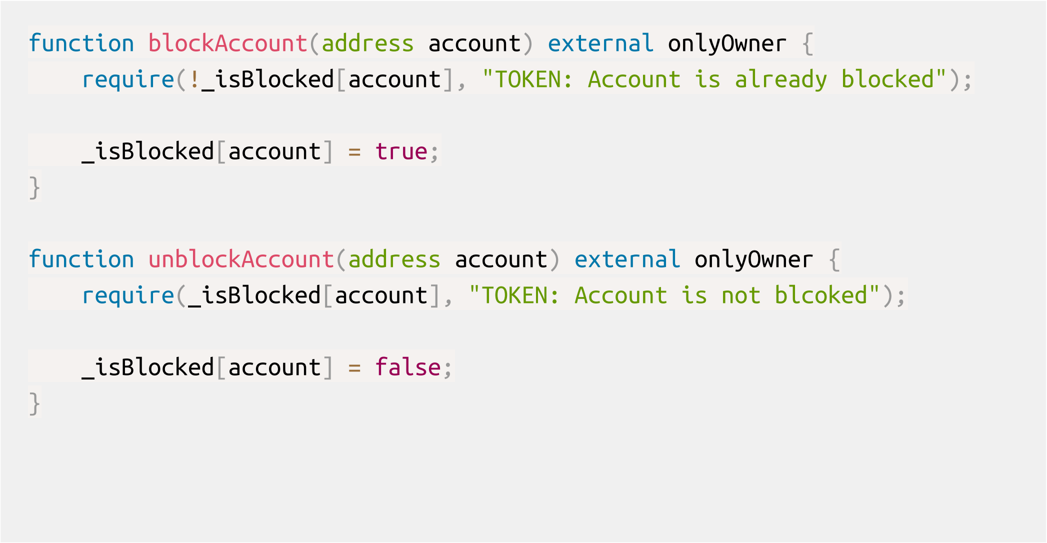
- Renouncing the ownership, which will eliminate the threats but it is non-reversible.

BC - Blacklists Addresses

Criticality	Critical
Location	DeflationaryToken.sol#L153,159
Status	Unresolved

Description

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



```
function blockAccount(address account) external onlyOwner {
    require(!_isBlocked[account], "TOKEN: Account is already blocked");

    _isBlocked[account] = true;
}

function unblockAccount(address account) external onlyOwner {
    require(_isBlocked[account], "TOKEN: Account is not blcoked");

    _isBlocked[account] = false;
}
```

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.

PISA - Potential Insufficient Swap Amount

Criticality	Minor / Informative
Location	DeflationaryToken.sol#L299,329
Status	Unresolved

Description

In the contract's transfer flow, the swap functionality is triggered when the contract's balance surpasses the `minimumTokensBeforeSwap`. If the `minimumTokensBeforeSwap` is set to zero and the contract has no balance, then the swap amount will be calculated as zero as well. Furthermore, the contract imposes a requirement for the swap amount to be at least 1 token. Consequently, if the minimum requirement is not met, the transaction will revert as the contract attempts to swap zero tokens, violating the minimum requirement.

```
uint256 amountToLiquify = (contractBalance * _liquidityFee) / _totalFeePrior / 2;
uint256 amountToSwap = contractBalance - amountToLiquify;

_swapTokensForETH(amountToSwap);
...
uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
    tokenAmount,
    1, // accept any amount of ETH
    path,
    address(this),
    block.timestamp
);
```

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.

FPP - Function Public Permissions

Criticality	Minor / Informative
Location	DeflationaryToken.sol#L120
Status	Unresolved

Description

The contract includes an `setAutomatedMarketMakerPair` function that allows users to modify the state of the `automatedMarketMakerPairs` variable, which is supposed to hold Uniswap pair addresses. This function is marked as public, meaning it can be accessed and executed by any user, regardless of their authorization or role. This lack of authorization control poses a significant security risk as it allows unauthorized users to add or remove any address to the `automatedMarketMakerPairs` variable, potentially leading to unexpected and unauthorized actions within the contract.

```
function setAutomatedMarketMakerPair(address pair, bool value) public {
    require(
        automatedMarketMakerPairs[pair] != value,
        "TOKEN: Automated market maker pair is already set to that value"
    );

    automatedMarketMakerPairs[pair] = value;
}
```

Recommendation

To mitigate this security risk and ensure proper authorization control, the team is strongly encouraged to implement an access control mechanism, such as a modifier or role-based access control, for the `setAutomatedMarketMakerPair` function. This mechanism should only allow authorized parties, such as the owner, to execute this function. By restricting access to this critical function, the team can enhance the security and integrity of the contract.

PAV - Pair Address Validation

Criticality	Minor / Informative
Location	DeflationaryToken.sol#L120
Status	Unresolved

Description

The `setAutomatedMarketMakerPair` function allows any user to set any arbitrary value without validation to the `automatedMarketMakerPairs` mapping, which is supposed to hold Uniswap pair addresses. This lack of validation can lead to unintended behavior, including the potential disruption of the contract's intended functionality.

```
function setAutomatedMarketMakerPair(address pair, bool value) public {
    require(
        automatedMarketMakerPairs[pair] != value,
        "TOKEN: Automated market maker pair is already set to that value"
    );

    automatedMarketMakerPairs[pair] = value;
}
```

Recommendation

To mitigate the risks associated with the absence of address validation in the pair address argument, it is recommended to implement comprehensive address validation mechanisms. A recommended approach could be to verify pair existence in the decentralized application. Prior to interacting with the pair address contract, perform checks to verify the existence and validity of the contract at the provided address. This can be achieved by querying the provider's contract or utilizing external libraries that provide contract verification services.

MEM - Misleading Error Messages

Criticality	Minor / Informative
Location	DeflationaryToken.sol#L180,184
Status	Unresolved

Description

The contract is using misleading error messages. These error messages do not accurately reflect the problem, making it difficult to identify and fix the issue. As a result, the users will not be able to find the root cause of the error.

```
require(
    (taxData.liquidityFeeOnBuy + taxData.operationsFeeOnBuy +
taxData.burnFeeOnBuy) <= 25,
    "TOKEN: Tax exceeds maximum value of 30%"
);
require(
    (taxData.liquidityFeeOnSell + taxData.operationsFeeOnSell +
taxData.burnFeeOnSell) <= 25,
    "TOKEN: Tax exceeds maximum value of 30%"
);
```

Recommendation

The team is suggested to provide a descriptive message to the errors. This message can be used to provide additional context about the error that occurred or to explain why the contract execution was halted. This can be useful for debugging and for providing more information to users that interact with the contract.

MAU - Misleading Address Usage

Criticality	Minor / Informative
Location	DeflationaryToken.sol#L311
Status	Unresolved

Description

The contract contains a variable called `burnWallet` address to represent a specific type of address, commonly acknowledged in the blockchain for a particular purpose. However, this wallet address within this contract is used to for receiving ETH as part of the swapping process. As a result, the designated address may not consistently serve its conventional purpose, potentially leading to unintended behaviors within the contract's operation.

```
Address.sendValue(payable(burnWallet), amountETHBurn);
```

Recommendation

It's always a good practice for the contract to contain variable names that are specific and descriptive. The team is advised to keep in mind the clarity and comprehensibility of the code to ensure that it accurately reflects the intended functionality. The designated address, which reflects a specific purpose within the contract, should ideally be renamed to maintain consistency in its functionality and to adhere to common practices, thereby reducing the potential for unexpected behaviors or vulnerabilities within the contract's operation.

RVA - Redundant Variable Assignment

Criticality	Minor / Informative
Location	DeflationaryToken.sol#L317
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's private function `_swapAndLiquify` includes a redundant reassignment of the `_totalFee` variable to its existing value. Specifically, the variable is set back to its original value after certain fee calculations. This operation appears unnecessary and does not contribute to the logic of the function.

```
_totalFee = _totalFeePrior;
```

Recommendation

The team is advised to take these segments into consideration 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.

PTRP - Potential Transfer Revert Propagation

Criticality	Minor / Informative
Location	DeflationaryToken.sol#L310
Status	Unresolved

Description

The contract sends funds to a `operationsWallet` 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.

```
Address.sendValue(payable(operationsWallet), amountETHOperations);
```

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.

MEE - Missing Events Emission

Criticality	Minor / Informative
Location	DeflationaryToken.sol#L126,132,141,150,156,162,168,173,187
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.

```
automatedMarketMakerPairs[pair] = value;
_isExcludedFromFee[account] = excluded;
_isExcludedFromMaxTransactionLimit[account] = excluded;
_isExcludedFromMaxWalletLimit[account] = excluded;
_isBlocked[account] = true;
_isBlocked[account] = false;
liquidityWallet = newLiquidityWallet;
operationsWallet = newOperationsWallet;
baseFeeData = taxData;
```

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.

PVC - Price Volatility Concern

Criticality	Minor / Informative
Location	DeflationaryToken.sol#L202
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.

```
function setMinimumTokensBeforeSwap(uint256 newValue) external onlyOwner {
    require(newValue != minimumTokensBeforeSwap, "TOKEN: Cannot update
minimumTokensBeforeSwap to same value");

    minimumTokensBeforeSwap = newValue;
}
```

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.

MEM - Misleading Error Messages

Criticality	Minor / Informative
Location	DeflationaryToken.sol#L213
Status	Unresolved

Description

The contract is using misleading error messages. These error messages do not accurately reflect the problem, making it difficult to identify and fix the issue. As a result, the users will not be able to find the root cause of the error.

```
require(success)
```

Recommendation

The team is suggested to provide a descriptive message to the errors. This message can be used to provide additional context about the error that occurred or to explain why the contract execution was halted. This can be useful for debugging and for providing more information to users that interact with the contract.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	DeflationaryToken.sol#L69,70,71,72,73,74
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 _owner
string memory _name
string memory _symbol
uint8 _decimals
uint256 _initialSupply
address _routerV2
```

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	DeflationaryToken.sol#L193,199,205
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.

```
maxTxAmount = newValue
maxWalletAmount = newValue
minimumTokensBeforeSwap = newValue
```

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.

L16 - Validate Variable Setters

Criticality	Minor / Informative
Location	DeflationaryToken.sol#L94,95,96,104
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.

```
liquidityWallet = liquidityWallet_
operationsWallet = operationsWallet_
burnWallet = burnWallet_
uniswapV2Pair = _uniswapV2Pair
```

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.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	DeflationaryToken.sol#L3
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 ^0.8.19;
```

Recommendation

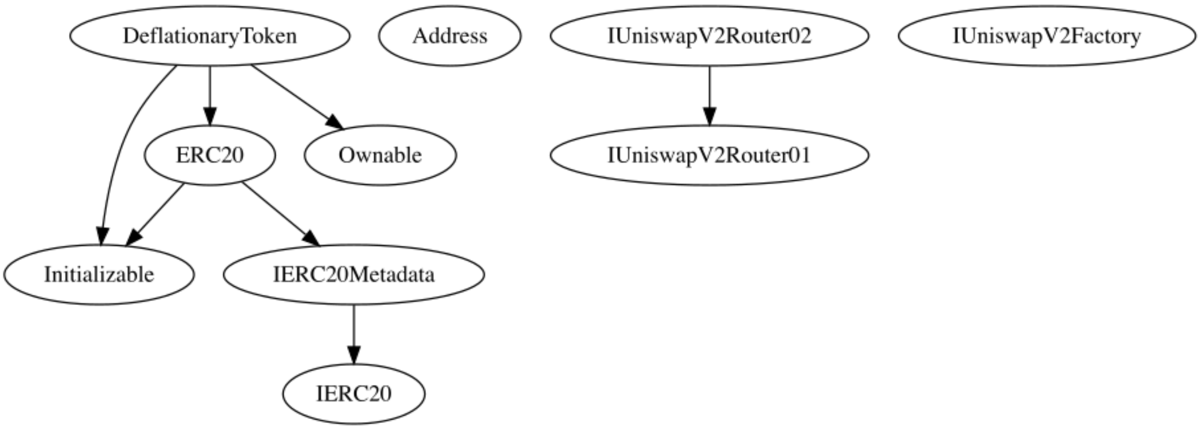
The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest 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.

Functions Analysis

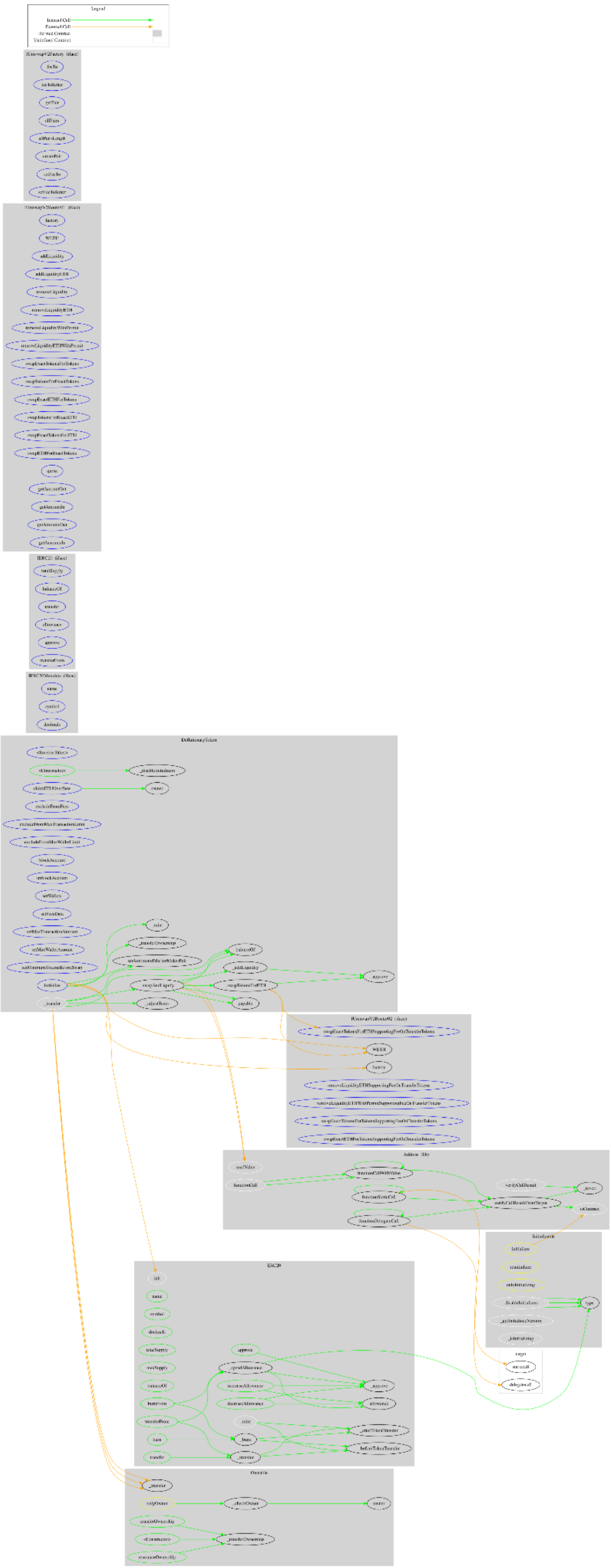
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
DeflationaryToken	Implementation	Initializable, ERC20, Ownable		
		External	Payable	-
		Public	✓	-
	initialize	External	✓	initializer
	setAutomatedMarketMakerPair	Public	✓	-
	excludeFromFees	External	✓	onlyOwner
	excludeFromMaxTransactionLimit	External	✓	onlyOwner
	excludeFromMaxWalletLimit	External	✓	onlyOwner
	blockAccount	External	✓	onlyOwner
	unblockAccount	External	✓	onlyOwner
	setWallets	External	✓	onlyOwner
	setFeesData	External	✓	onlyOwner
	setMaxTransactionAmount	External	✓	onlyOwner
	setMaxWalletAmount	External	✓	onlyOwner
	setMinimumTokensBeforeSwap	External	✓	onlyOwner
	claimETHOverflow	External	✓	onlyOwner
	_transfer	Internal	✓	

	_adjustTaxes	Private	✓	
	_swapAndLiquify	Private	✓	
	_swapTokensForETH	Private	✓	
	_addLiquidity	Private	✓	

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IUniswapV2Factory (iface)
feeTo
feeToSetter
getPair
allPairs
allPairsLength
createPair
setFeeTo
setFeeToSetter
IUniswapV2Router01 (iface)
factory
WETH
addLiquidity
addLiquidityETH
removeLiquidity
removeLiquidityETH
removeLiquidityWithPermit
removeLiquidityETHWithPermit
swapExactTokensForTokens
swapTokensForExactTokens
swapExactETHForTokens
swapTokensForExactETH
swapExactTokensForETH
swapETHForExactTokens
quote
getAmountOut
getAmountIn
getAmountsOut
getAmountsIn
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
IERC20Metadata (iface)
name
symbol
decimals
DeflationaryToken
<Receive Ether>
<Constructor>
_disableInitializers
claimETHOverflow
owner
excludeFromFees
excludeFromMaxTransactionLimit
excludeFromMaxWalletLimit
blockAccount
unblockAccount
setWallets
setFeesData
setMaxTransactionAmount
setMaxWalletAmount
setMinimumTokensBeforeSwap
initialize
_transfer
_mint
_transferOwnership
setAutomatedMarketMakerPair
balanceOf
_addLiquidity
swapAndLiquify
swapTokensForETH
_approve
_adjustTaxes
payable
IUniswapV2Router02 (iface)
swapExactTokensForETHSupportingFeeOnTransferTokens
WETH
factory
removeLiquidityETHSupportingFeeOnTransferTokens
removeLiquidityETHWithPermitSupportingFeeOnTransferTokens
swapExactTokensForTokensSupportingFeeOnTransferTokens
swapExactETHForTokensSupportingFeeOnTransferTokens
AddressUtils
sendValue
functionCall
functionCallWithValue
functionStaticCall
functionDelegateCall
verifyCallResult
verifyCallResultFromTarget
_revert
isContract
Initializable
initializer
reinitializer
onlyInitializing
_disableInitializers
_getInitializedVersion
_isInitializing
type
target
staticcall
delegatecall
ERC20
init
name
symbol
decimals
totalSupply
maxSupply
balanceOf
burnFrom
transferFrom
burn
transfer
approve
_spendAllowance
increaseAllowance
decreaseAllowance
_mint
_burn
_transfer
_approve
allowance
_afterTokenTransfer
_beforeTokenTransfer
Ownable
_transfer
onlyOwner
_checkOwner
owner
transferOwnership
<Constructor>
_transferOwnership
renounceOwnership

Summary

BearStreetCoin 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 stop transactions and massively blacklist addresses. 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 25% 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