Cyberscope

Audit Report

HER Coin

December 2023

SHA256 3002c9e55fd4c155123570d9c33b11f799ee1c76db99681d6d679512c296d523

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	Unresolved
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

Diagnostics

🔴 Critical 🟠 Medium ● Minor / Informative

Severity	Code	Description	Status
●	TCO	Tax Calculation Optimization	Unresolved
●	CR	Code Repetition	Unresolved
●	RES	Redundant Event Statement	Unresolved
●	MEE	Missing Events Emission	Unresolved
●	MCM	Misleading Comment Messages	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L09	Dead Code Elimination	Unresolved
●	L13	Divide before Multiply Operation	Unresolved
●	L15	Local Scope Variable Shadowing	Unresolved
●	L16	Validate Variable Setters	Unresolved
●	L17	Usage of Solidity Assembly	Unresolved
●	L18	Multiple Pragma Directives	Unresolved
●	L19	Stable Compiler Version	Unresolved
●	L20	Succeeded Transfer Check	Unresolved

Table of Contents

Review

Contract Name	HERCoin
Testing Deploy	https://goerli.etherscan.io/address/0x95a32e65e12cb2144fb8557c1978d744f5d05d0b
Symbol	HER
Decimals	18

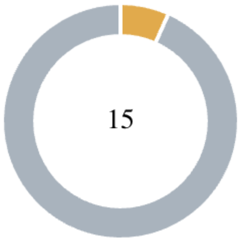
Audit Updates

Initial Audit	04 Dec 2023

Source Files

Filename	SHA256
contracts/HERCoin.sol	3002c9e55fd4c155123570d9c33b11f799ee1c76db99681d6d679512c296d523

Findings Breakdown



● Critical	0
● Medium	1
● Minor / Informative	14

Severity	Unresolved	Acknowledged	Resolved	Other
● Critical	0	0	0	0
● Medium	1	0	0	0
● Minor / Informative	14	0	0	0

MT - Mints Tokens

Criticality	Medium
Location	contracts/HERCoin.sol#L2448
Status	Unresolved

Description

The contract owner has the authority to mint tokens until the `TOTAL_SUPPLY` amount is reached. The owner may take advantage of it by calling the `airdrop` function. As a result, the contract tokens will be highly inflated.

```
   function airdrop(address accounts, uint256 amounts) external
onlyOwner {
            _mint(accounts, amounts);

        uint256 newTotalSupply = totalSupply();
        if (newTotalSupply > TOTAL_SUPPLY) {
            _revert(ErrTotalSupplyExceeded.selector);
        }
    }
```

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.

TCO - Tax Calculation Optimization

Criticality	Minor / Informative
Location	contracts/HERCoin.sol#L2574
Status	Unresolved

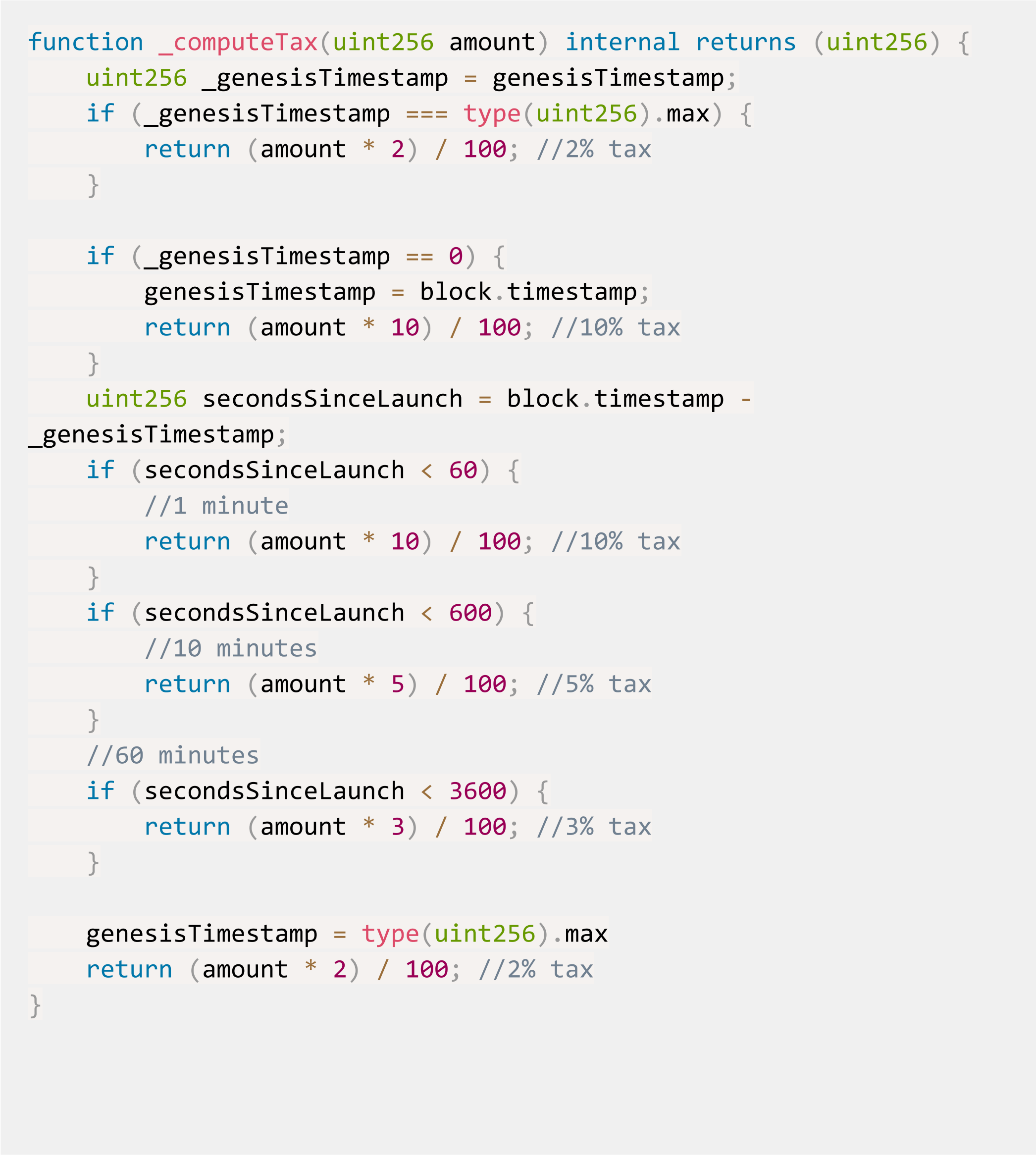
Description

The contract calculates the applicable tax by using the `_computeTax` method. The method applies different taxes for the first 3600 seconds. Later, it will always apply a 2% tax. That means that after 3600 seconds the contract will reduntantly execute all the previous if statements.

```
function _computeTax(uint256 amount) internal returns (uint256) {
    uint256 _genesisTimestamp = genesisTimestamp;
    if (_genesisTimestamp == 0) {
        genesisTimestamp = block.timestamp;
        return (amount * 10) / 100; //10% tax
    }
    uint256 secondsSinceLaunch = block.timestamp -
_genesisTimestamp;
    if (secondsSinceLaunch < 60) {
        //1 minute
        return (amount * 10) / 100; //10% tax
    }
    if (secondsSinceLaunch < 600) {
        //10 minutes
        return (amount * 5) / 100; //5% tax
    }
    //60 minutes
    if (secondsSinceLaunch < 3600) {
        return (amount * 3) / 100; //3% tax
    }
    return (amount * 2) / 100; //2% tax
}
```

Recommendation

The team is adviced to improve the performance of the `_computeTax` method taking into consideration the long-run. Once the contract reaches the 3600 seconds, it could set to the `genesisTimestamp` the max value. At the beggening of the method, it could return fast if the `genesisTimestamp` equals to the max value.



```
function _computeTax(uint256 amount) internal returns (uint256) {
    uint256 _genesisTimestamp = genesisTimestamp;
    if (_genesisTimestamp === type(uint256).max) {
        return (amount * 2) / 100; //2% tax
    }

    if (_genesisTimestamp == 0) {
        genesisTimestamp = block.timestamp;
        return (amount * 10) / 100; //10% tax
    }
    uint256 secondsSinceLaunch = block.timestamp -
_genesisTimestamp;
    if (secondsSinceLaunch < 60) {
        //1 minute
        return (amount * 10) / 100; //10% tax
    }
    if (secondsSinceLaunch < 600) {
        //10 minutes
        return (amount * 5) / 100; //5% tax
    }
    //60 minutes
    if (secondsSinceLaunch < 3600) {
        return (amount * 3) / 100; //3% tax
    }

    genesisTimestamp = type(uint256).max
    return (amount * 2) / 100; //2% tax
}
```

CR - Code Repetition

Criticality	Minor / Informative
Location	contracts/HERCoin.sol#L2380
Status	Unresolved

Description

The contract contains repetitive code segments. There are potential issues that can arise when using code segments in Solidity. Some of them can lead to issues like gas efficiency, complexity, readability, security, and maintainability of the source code. It is generally a good idea to try to minimize code repetition where possible.

```
_addressData[_owner] = AddressData({isPool: false,
isExcludedFromTax: true});
_addressData[address(this)] = AddressData({isPool: false,
isExcludedFromTax: true});
_addressData[_owner] = AddressData({isPool: false,
isExcludedFromTax: true});
```

Recommendation

The team is advised to avoid repeating the same code in multiple places, which can make the contract easier to read and maintain.

RES - Redundant Event Statement

Criticality	Minor / Informative
Location	contracts/HERCoin.sol#L2303
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 `ErrNotLiquidtyProvider` event statement is not used in the contract's implemantation.

```
error ErrNotLiquidtyProvider();
```

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. It is recommend removing the unused event statement from the contract..

MEE - Missing Events Emission

Criticality	Minor / Informative
Location	contracts/HERCoin.sol#L2496,2504
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 emergencyOverrideAddressData(address _address, bool
_isPool, bool _isExcludedFromTax) external onlyOwner {
        _addressData[_address] = AddressData({isPool: _isPool,
isExcludedFromTax: _isExcludedFromTax});
    }

    function setLimitsStatus(bool _limitsOn) external onlyOwner {
        limitsOn = _limitsOn;
    }
```

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.

MCM - Misleading Comment Messages

Criticality	Minor / Informative
Location	contracts/HERCoin.sol#L2311
Status	Unresolved

Description

The contract is using misleading comment messages. These comment messages do not accurately reflect the actual implementation, making it difficult to understand the source code. As a result, the users will not comprehend the source code's actual implementation.

Specifically, the contract calculates `MAX_HOLDING_AMOUNT_PER_WALLET` as 1% of the `TOTAL_SUPPLY`, but the accompanying comment incorrectly states that this value is `3%`. Similarly, both `MAX_BUY_ORDER` and `MAX_SELL_ORDER` are calculated as 0.5% of the `TOTAL_SUPPLY`, whereas the comments suggest these values are equal to `3%`. These discrepancies between the code and its documentation can lead to confusion and misinterpretation of the contract's intended functionality, potentially affecting stakeholders' understanding and trust in the contract.

```
   uint256 public constant MAX_HOLDING_AMOUNT_PER_WALLET =
TOTAL_SUPPLY * uint256(1) / uint256(100); //3%

    /// @dev adjust these values as needed
    uint256 public constant MAX_BUY_ORDER = TOTAL_SUPPLY *
uint256(5) / uint256(1000); //3%

    /// @dev adjust these values as needed
    uint256 public constant MAX_SELL_ORDER = TOTAL_SUPPLY *
uint256(5) / uint256(1000); //3
```

Recommendation

The team is advised to carefully review the comment in order to reflect the actual implementation. To improve code readability, the team should use more specific and descriptive comment messages. It is recommended to revise the comments in the contract to accurately reflect the actual implementation. The comment for `MAX_HOLDING_AMOUNT_PER_WALLET` should be updated to indicate that it represents `1%`

of the `TOTAL_SUPPLY` , not `3%` . Similarly, the comments for `MAX_BUY_ORDER` and `MAX_SELL_ORDER` should be corrected to state that they are each set to `0.5%` of the `TOTAL_SUPPLY` .

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	contracts/HERCoin.sol#L1187,1198,1470,2274,2289,2465,2482,2496,2504,2521
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 _EIP712Name() internal view returns (string memory)
{
        return _name.toStringWithFallback(_nameFallback);
    }

    function _EIP712Version() internal view returns (string
memory) {
        return _version.toStringWithFallback(_versionFallback);
    }

    function DOMAIN_SEPARATOR() external view returns
(bytes32);
```

```
...
```

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.

L09 - Dead Code Elimination

Criticality	Minor / Informative
Location	contracts/HERCoin.sol#L91,190,200,210,220,230,250,260,384,407,414,422,431,521,528,536,547,563,647,661,697,708,750,761,799,812,842,852,881,906,913,922,941,948,983,1002,1016,1260,1293,1304,1316,2117
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 _useCheckedNonce(address owner, uint256 nonce)
internal virtual {
        uint256 current = _useNonce(owner);
        if (nonce != current) {
            revert InvalidAccountNonce(owner, current);
        }
    }

function getAddressSlot(bytes32 slot) internal pure returns
(AddressSlot storage r) {
        /// @solidity memory-safe-assembly
        assembly {
            r.slot := slot
        }
    }

...
```

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.

L13 - Divide before Multiply Operation

Criticality	Minor / Informative
Location	contracts/HERCoin.sol#L609,612,624,628,629,630,631,632,633,639
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.

```
denominator := div(denominator, twos)
inverse *= 2 - denominator * inverse
```

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	contracts/HERCoin.sol#L2233,2373
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.

```
(string memory name) EIP712(name, "1")
(address _owner) payable ERC20("HER Coin", "HER")

ERC20Permit("HER Coin") Ownable(_owner)
```

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.

L16 - Validate Variable Setters

Criticality	Minor / Informative
Location	contracts/HERCoin.sol#L2468,2486
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.

```
(bool os,) = payable(to).call{value:
address(this).balance}("");
...
tokenRedeemer = _tokenRedeemer;
```

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.

L17 - Usage of Solidity Assembly

Criticality	Minor / Informative
Location	contracts/HERCoin.sol#L192,202,212,222,232,242,252,262,336,570,887,985,1031,1268,2622
Status	Unresolved

Description

Using assembly can be useful for optimizing code, but it can also be error-prone. It's important to carefully test and debug assembly code to ensure that it is correct and does not contain any errors.

Some common types of errors that can occur when using assembly in Solidity include Syntax, Type, Out-of-bounds, Stack, and Revert.

```
assembly {
            r.slot := slot
        }

assembly {
            r.slot := store.slot
...
            mstore(add(str, 0x20), sstr)
        }

assembly {
       let mm := mulmod(x, y, not(0))
       prod1 := sub(sub(mm, prod0), lt(mm, prod0))
            }

...
```

Recommendation

It is recommended to use assembly sparingly and only when necessary, as it can be difficult to read and understand compared to Solidity code.

L18 - Multiple Pragma Directives

Criticality	Minor / Informative
Location	contracts/HERCoin.sol#L5,54,104,136,273,398,444,862,958,1046,1208,1385,1477,1642,1669,1771,1853,1881,2199,2282
Status	Unresolved

Description

If the contract includes multiple conflicting pragma directives, it may produce unexpected errors. To avoid this, it's important to include the correct pragma directive at the top of the contract and to ensure that it is the only pragma directive included in the contract.

```
pragma solidity >=0.5.0;
pragma solidity ^0.8.20;
```

Recommendation

It is important to include only one pragma directive at the top of the contract and to ensure that it accurately reflects the version of Solidity that the contract is written in.

By including all required compiler options and flags in a single pragma directive, the potential conflicts could be avoided and ensure that the contract can be compiled correctly.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	contracts/HERCoin.sol#L54,104,136,273,398,444,862,958,1046,1208,1385,1477,1642,1669,1771,1853,1881,2199,2282
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 ^0.8.20;
...
```

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	contracts/HERCoin.sol#L2471
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.

```
ERC20(_token).transfer(to,
ERC20(_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**
PoolAddress	Library			
	getPoolKey	Internal		
	computeAddress	Internal		
Nonces	Implementation			
	nonces	Public		-
	_useNonce	Internal	✓	
	_useCheckedNonce	Internal	✓	
IERC5267	Interface			
	eip712Domain	External		-
StorageSlot	Library			
	getAddressSlot	Internal		
	getBooleanSlot	Internal		
	getBytes32Slot	Internal		
	getUint256Slot	Internal		
	getStringSlot	Internal		

	getStringSlot	Internal		
	getBytesSlot	Internal		
	getBytesSlot	Internal		
ShortStrings	Library			
	toShortString	Internal		
	toString	Internal		
	byteLength	Internal		
	toShortStringWithFallback	Internal	✓	
	toStringWithFallback	Internal		
	byteLengthWithFallback	Internal		
SignedMath	Library			
	max	Internal		
	min	Internal		
	average	Internal		
	abs	Internal		
Math	Library			
	tryAdd	Internal		
	trySub	Internal		
	tryMul	Internal		
	tryDiv	Internal		

	tryMod	Internal		
	max	Internal		
	min	Internal		
	average	Internal		
	ceilDiv	Internal		
	mulDiv	Internal		
	mulDiv	Internal		
	sqrt	Internal		
	sqrt	Internal		
	log2	Internal		
	log2	Internal		
	log10	Internal		
	log10	Internal		
	log256	Internal		
	log256	Internal		
	unsignedRoundsUp	Internal		
Strings	Library			
	toString	Internal		
	toStringSigned	Internal		
	toHexString	Internal		
	toHexString	Internal		
	toHexString	Internal		

	equal	Internal		
MessageHashUtils	Library			
	toEthSignedMessageHash	Internal		
	toEthSignedMessageHash	Internal		
	toDataWithIntendedValidatorHash	Internal		
	toTypedDataHash	Internal		
EIP712	Implementation	IERC5267		
		Public	✓	-
	_domainSeparatorV4	Internal		
	_buildDomainSeparator	Private		
	_hashTypedDataV4	Internal		
	eip712Domain	Public		-
	_EIP712Name	Internal		
	_EIP712Version	Internal		
ECDSA	Library			
	tryRecover	Internal		
	recover	Internal		
	tryRecover	Internal		
	recover	Internal		
	tryRecover	Internal		

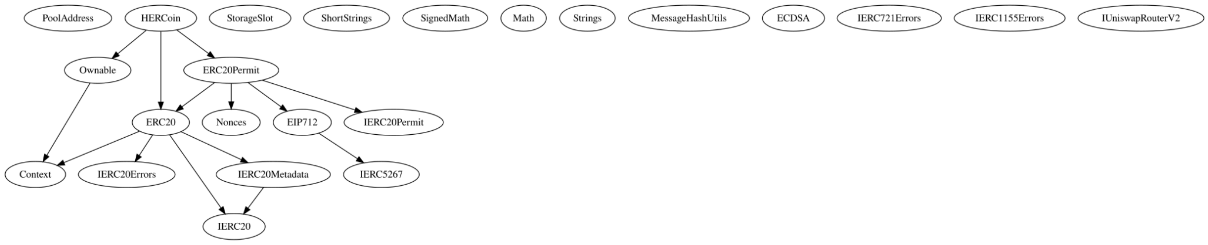
	recover	Internal		
	_throwError	Private		
IERC20Permit	Interface			
	permit	External	✓	-
	nonces	External		-
	DOMAIN_SEPARATOR	External		-
IERC20Errors	Interface			
IERC721Errors	Interface			
IERC1155Errors	Interface			
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	_checkOwner	Internal		
	renounceOwnership	Public	✓	onlyOwner

	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	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, IERC20Errors		
		Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-

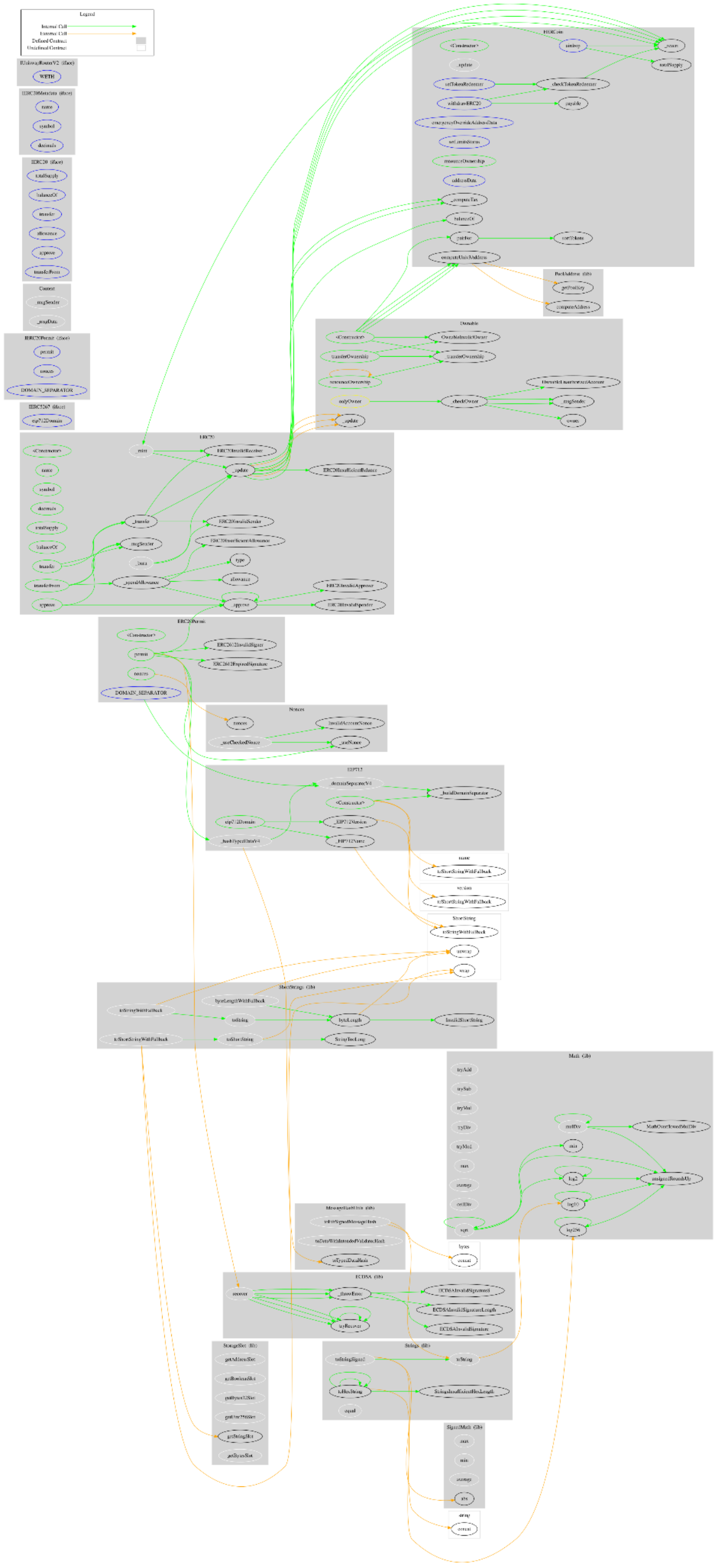
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	_transfer	Internal	✓	
	_update	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_approve	Internal	✓	
	_approve	Internal	✓	
	_spendAllowance	Internal	✓	
ERC20Permit	Implementation	ERC20, IERC20Permit, EIP712, Nonces		
		Public	✓	EIP712
	permit	Public	✓	-
	nonces	Public		-
	DOMAIN_SEPARATOR	External		-
IUniswapRouterV2	Interface			
	WETH	External		-

HERCoin	Implementation	ERC20, ERC20Permit, Ownable		
		Public	Payable	ERC20 ERC20Permit Ownable
	_update	Internal	✓	
	airdrop	External	✓	onlyOwner
	withdrawERC20	External	✓	-
	setTokenRedeemer	External	✓	-
	emergencyOverrideAddressData	External	✓	onlyOwner
	setLimitsStatus	External	✓	onlyOwner
	renounceOwnership	Public	✓	onlyOwner
	addressData	External		-
	sortTokens	Internal		
	pairFor	Internal		
	_computeTax	Internal	✓	
	computeUniv3Address	Internal		
	_checkTokenRedeemer	Internal		
	_revert	Private		

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IUniswapRouterV2 (iface)
WETH
IERC20Metadata (iface)
name
symbol
decimals
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
Context
_msgSender
_msgData
IERC20Permit (iface)
permit
nonces
DOMAIN_SEPARATOR
IERC5267 (iface)
eip712Domain
HERCoin
<Constructor>
airdrop
_revert
totalSupply
_update
setTokenRedeemer
_checkTokenRedeemer
withdrawERC20
payable
emergencyOverrideAddressData
setLimitStatus
renounceOwnership
addressData
_computeTax
balanceOf
pairFor
sortTokens
computeUniv3Address
PoolAddress (lib)
getPoolKey
computeAddress
Ownable
<Constructor>
OwnableInvalidOwner
transferOwnership
_transferOwnership
renounceOwnership
OwnableUnauthorizedAccount
onlyOwner
_checkOwner
_msgSender
owner
_update
ERC20
<Constructor>
_mint
ERC20InvalidReceiver
name
_update
ERC20InsufficientBalance
symbol
decimals
totalSupply
_transfer
ERC20InvalidSender
balanceOf
_msgSender
ERC20InsufficientAllowance
transfer
_burn
type
transferFrom
_spendAllowance
allowance
ERC20InvalidApprover
approve
_approve
ERC20InvalidSpender
ERC20Permit
<Constructor>
permit
ERC2612InvalidSigner
ERC2612ExpiredSignature
nonces
DOMAIN_SEPARATOR
Nonces
nonces
InvalidAccountNonce
_useCheckedNonce
_useNonce
EIP712
_domainSeparatorV4
_buildDomainSeparator
<Constructor>
eip712Domain
_EIP712Version
_hashTypedDataV4
_EIP712Name
name
toShortStringWithFallback
version
toShortStringWithFallback
ShortString
toStringWithFallback
unwrap
wrap
ShortStrings (lib)
byteLengthWithFallback
toStringWithFallback
toString
byteLength
InvalidShortString
toShortStringWithFallback
toShortString
StringTooLong
Math (lib)
tryAdd
trySub
tryMul
tryDiv
mulDiv
MathOverflowedMulDiv
tryMod
min
max
average
log2
unsignedRoundsUp
ceilDiv
log10
sqrt
log256
MessageHashUtils (lib)
toEthSignedMessageHash
toDataWithIntendedValidatorHash
toTypedDataHash
bytes
concat
ECDSA (lib)
recover
_throwError
ECDSAInvalidSignatureS
ECDSAInvalidSignatureLength
tryRecover
ECDSAInvalidSignature
StorageSlot (lib)
getAddressSlot
getBooleanSlot
getBytes32Slot
getUint256Slot
getStringSlot
getBytesSlot
Strings (lib)
toStringSigned
toString
toHexString
StringsInsufficientHexLength
equal
SignedMath (lib)
max
min
average
abs
string
concat

Summary

HER Coin 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 mint tokens. if the contract owner abuses the mint functionality, then the contract will be highly inflated. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats. The smart contract includes a dynamic fee mechanism that adjusts the tax rate based on the time elapsed since the genesisTimestamp was set. This mechanism is designed to apply to both buy and sell transactions. Specifically, the tax rates are structured as follows:

- For the First 1 Minute: The tax is set at a high rate of 10%.
- From 1 to 10 Minutes: After the first minute and until the 10th minute, the tax rate is reduced to 5%.
- From 10 to 60 Minutes: Between the 10th and 60th minute, the tax rate is further reduced to 3%.
- After 60 Minutes: Once 60 minutes have passed since the genesisTimestamp, the tax rate is set at 2%.

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