Cyberscope

Audit Report

HTS

June 2024

SHA256 4528c788a45b47e5884c17ebae1dff73145a91513e0dec41bdd381031d309c50

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
Medium	EPC	Existing Pair Creation	Acknowledged
Minor	CCR	Contract Centralization Risk	Acknowledged
Minor	MTEE	Missing Transfer Event Emission	Acknowledged
Minor	PLPI	Potential Liquidity Provision Inadequacy	Acknowledged
Minor	RSML	Redundant SafeMath Library	Acknowledged
Minor	L04	Conformance to Solidity Naming Conventions	Acknowledged
Minor	L20	Succeeded Transfer Check	Acknowledged

Table of Contents

Review

Contract Name	HTS
Testing Deploy	https://testnet.bscscan.com/address/0xad8a4d12508573e8ebf85c33efe213b41f438fc0
Symbol	HTS
Decimals	18
Total Supply	100,000,000
Badge Eligibility	Yes

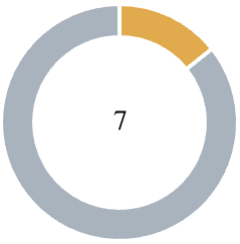
Audit Updates

Initial Audit	06 Jun 2024
Corrected Phase 2	06 Jun 2024

Source Files

Filename	SHA256
HTS.sol	4528c788a45b47e5884c17ebae1dff73145a91513e0dec41bdd381031d309c50

Findings Breakdown



Critical	0
Medium	1
Minor / Informative	6

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	0	1	0	0
Minor / Informative	0	6	0	0

EPC - Existing Pair Creation

Criticality	Medium
Location	contracts/home3.sol#L258
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 openTrading() external onlyOwner {
    require(!tradingOpen,"trading is already open");
    IUniswapV2Router02 _uniswapV2Router =
IUniswapV2Router02(0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D);
    uniswapV2Router = _uniswapV2Router;
    _approve(address(this), address(uniswapV2Router), _tTotal);
    address _uniswapV2pair =
IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),
_uniswapV2Router.WETH());
    uniswapV2Pair[_uniswapV2pair] = true;
    uniswapV2Router.addLiquidityETH{value:
address(this).balance}(address(this),balanceOf(address(this)),0,0,owner(),b
lock.timestamp);
    tradingOpen = true;
    IERC20(_uniswapV2pair).approve(address(uniswapV2Router),
type(uint).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.

Team Update

Team will be holding tokens before trading. So no one will be able to create a pair to break this logic.

CCR - Contract Centralization Risk

Criticality	Minor / Informative
Location	contracts/home3.sol#L165,174,264,299,314,326,336,347,358,369,376,384
Status	Unresolved

Description

The contract's functionality and behavior are heavily dependent on external parameters or configurations. While external configuration can offer flexibility, it also poses several centralization risks that warrant attention. Centralization risks arising from the dependence on external configuration include Single Point of Control, Vulnerability to Attacks, Operational Delays, Trust Dependencies, and Decentralization Erosion.

```
function unlockFunction(Functions _func) external onlyOwner
function lockFunction(Functions _func) external onlyOwner
function openTrading() external onlyOwner
function whitelistAddress(address _addr, bool _bool) external
onlyOwner
function rescueERC20(IERC20 token, uint256 amount) external
onlyOwner
function changeSwapAmount(uint256 _newThreshold) external onlyOwner
function changeBuyTax(uint256 _newTax) external onlyOwner
function changeSellTax(uint256 _newTax) external onlyOwner
function setTreasury(address payable _treasuryWallet) external
onlyOwner notLocked(Functions.changeTreWallet)
function addLPPair(address _address) external onlyOwner
function manualswap() external onlyOwner
function manualSend() external onlyOwner
```

Recommendation

To address this finding and mitigate centralization risks, it is recommended to evaluate the feasibility of migrating critical configurations and functionality into the contract's codebase itself. This approach would reduce external dependencies and enhance the contract's self-sufficiency. It is essential to carefully weigh the trade-offs between external configuration flexibility and the risks associated with centralization.

Team Update

None of the onlyOwner functions can break the trading of tokens and can't really create delay in Operations. On top of that, we have timelock on functions we think can be impact us in long run so it won't be an issue (treasury wallet and renounce)

MTEE - Missing Transfer Event Emission

Criticality	Minor / Informative
Location	contracts/home3.sol#L281
Status	Unresolved

Description

The contract does not emit an event when portions of the main amount are transferred during the transfer process. This lack of event emission results in decreased transparency and traceability regarding the flow of tokens, and hinders the ability of decentralized applications (dApps), such as blockchain explorers, to accurately track and analyze these transactions.

More specifically, when the contract deducts the fees from the amount being transferred, no event is emitted.

```
balance[address(this)] = balance[address(this)].add(stContract);
```

Recommendation

It is advisable to incorporate the emission of detailed event logs following each asset transfer. These logs should encapsulate key transaction details, including the identities of the sender and receiver, and the quantity of assets transferred. Implementing this practice will enhance the reliability and transparency of transaction tracking systems, ensuring accurate data availability for ecosystem participants.

PLPI - Potential Liquidity Provision Inadequacy

Criticality	Minor / Informative
Location	contracts/home3.sol#L242
Status	Unresolved

Description

The contract operates under the assumption that liquidity is consistently provided to the pair between the contract's token and the native currency. However, there is a possibility that liquidity is provided to a different pair. This inadequacy in liquidity provision in the main pair could expose the contract to risks. Specifically, during eligible transactions, where the contract attempts to swap tokens with the main pair, a failure may occur if liquidity has been added to a pair other than the primary one. Consequently, transactions triggering the swap functionality will result in a revert.

```
function swapTokensForEth(uint256 tokenAmount) private lockTheSwap {
    address[] memory path = new address[](2);
    path[0] = address(this);
    path[1] = uniswapV2Router.WETH();
    _approve(address(this), address(uniswapV2Router), tokenAmount);
    uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
        tokenAmount, 0, path, address(this), block.timestamp
    );
}
```

Recommendation

The team is advised to implement a runtime mechanism to check if the pair has adequate liquidity provisions. This feature allows the contract to omit token swaps if the pair does not have adequate liquidity provisions, significantly minimizing the risk of potential failures.

Furthermore, the team could ensure the contract has the capability to switch its active pair in case liquidity is added to another pair.

Additionally, the contract could be designed to tolerate potential reverts from the swap functionality, especially when it is a part of the main transfer flow. This can be achieved by executing the contract's token swaps in a non-reversible manner, thereby ensuring a more resilient and predictable operation.

RSML - Redundant SafeMath Library

Criticality	Minor / Informative
Location	HTS.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	HTS.sol#L100,120,128,129,130,140,141,142,143,145,146,147,148,166,175,299,327,337,347,357,368,426
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);
uint256 private constant _tTotal = 1e26
string private constant _name = "Home3"
string private constant _symbol = "HTS"
uint8 private constant _decimals = 18
event swapAmountUpdated(uint256 _newThreshold);
event buyTaxUpdated(uint256 _newTax);
event sellTaxUpdated(uint256 _newTax);
event treasuryUpdated(address _newWallet);
event functionUnlockInitiated(Functions _func);
event functionLocked(Functions _func);
event whitelistUpdated(address _addr,bool _bool);
event lpPairAdded(address _addr);
Functions _func

...
```

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.

L20 - Succeeded Transfer Check

Criticality	Minor / Informative
Location	HTS.sol#L319
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(treasuryWallet, amount)
```

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

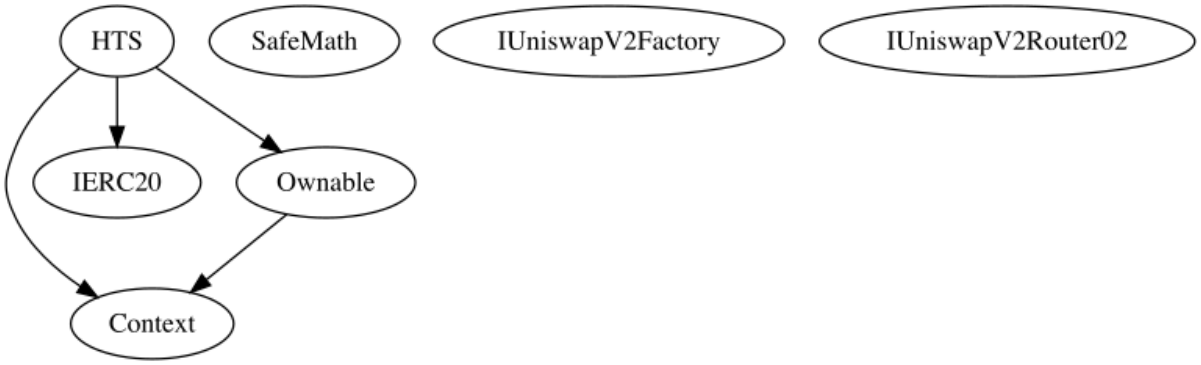
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
	_msgSender	Internal		
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
Ownable	Implementation	Context		

		Public	✓	-
	owner	Public		-
	renounceOwnership	External	✓	onlyOwner notLocked
IUniswapV2Factory	Interface			
	createPair	External	✓	-
IUniswapV2Router02	Interface			
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
	factory	External		-
	WETH	External		-
	addLiquidityETH	External	Payable	-
HTS	Implementation	Context, IERC20, Ownable		
		Public	✓	-
	unlockFunction	External	✓	onlyOwner
	lockFunction	External	✓	onlyOwner
	transferFrom	External	✓	-
	_approve	Private	✓	
	_transfer	Private	✓	

	swapTokensForEth	Private	✓	lockTheSwap
	sendTaxToTreasury	Private	✓	
	openTrading	External	✓	onlyOwner
	_tokenTransfer	Private	✓	
	whitelistAddress	External	✓	onlyOwner
		External	Payable	-
	rescueERC20	External	✓	onlyOwner
	changeSwapAmount	External	✓	onlyOwner
	changeBuyTax	External	✓	onlyOwner
	changeSellTax	External	✓	onlyOwner
	setTreasury	External	✓	onlyOwner notLocked
	addLPPair	External	✓	onlyOwner
	manualswap	External	✓	onlyOwner
	manualSend	External	✓	onlyOwner
	name	External		-
	symbol	External		-
	decimals	External		-
	totalSupply	External		-
	balanceOf	Public		-
	transfer	External	✓	-
	allowance	External		-

	approve	External	✓	-
	isWhitelisted	External		-

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
IUniswapV2Factory (iface)
createPair
SafeMath (lib)
add
sub
Ownable
onlyOwner
_msgSender
owner
<Constructor>
notLocked
renounceOwnership
HTS
owner
lockTheSwap
<Constructor>
unlockFunction
lockFunction
transferFrom
whitelistAddress
<Receive Ether>
rescueERC20
changeSwapAmount
changeBuyTax
changeSellTax
setTreasury
addLPPair
name
symbol
decimals
totalSupply
transfer
allowance
isWhitelisted
manualswap
swapTokensForEth
_transfer
_tokenTransfer
balanceOf
sendTaxToTreasury
manualSend
openTrading
type
approve
_approve
Context
_msgSender
IUniswapV2Router02 (iface)
swapExactTokensForETHSupportingFeeOnTransferTokens
WETH
factory
addLiquidityETH

Summary

HTS 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. There is also a limit of max 5% fees.

This audit investigates security issues, business logic concerns and potential improvements. The team has acknowledged the findings.

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