



Cyberscope

# Audit Report

## **Hentai**

March 2024

SHA256 21d6603cca1b75912601a03eea2735efcdf2b94aee31068a8ca801ce3a6f29d5

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	Passed

# Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	EPC	Existing Pair Creation	Unresolved
●	IDI	Immutable Declaration Improvement	Unresolved
●	MCM	Misleading Comment Messages	Unresolved
●	MEE	Missing Events Emission	Unresolved
●	RED	Redudant Event Declaration	Unresolved
●	RRS	Redundant Require Statement	Unresolved
●	RSML	Redundant SafeMath Library	Unresolved
●	RVR	Redundant Variable Reassignment	Unresolved
●	L02	State Variables could be Declared Constant	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L16	Validate Variable Setters	Unresolved

# Table of Contents

<b>Analysis</b>	<b>1</b>
<b>Diagnostics</b>	<b>2</b>
<b>Table of Contents</b>	<b>3</b>
<b>Review</b>	<b>5</b>
Audit Updates	5
Source Files	5
<b>Findings Breakdown</b>	<b>6</b>
ST - Stops Transactions	7
Description	7
Recommendation	7
EPC - Existing Pair Creation	8
Description	8
Recommendation	9
IDI - Immutable Declaration Improvement	10
Description	10
Recommendation	10
MCM - Misleading Comment Messages	11
Description	11
Recommendation	11
MEE - Missing Events Emission	12
Description	12
Recommendation	12
RED - Redudant Event Declaration	13
Description	13
Recommendation	13
RRS - Redundant Require Statement	14
Description	14
Recommendation	14
RSML - Redundant SafeMath Library	15
Description	15
Recommendation	15
RVR - Redundant Variable Reassignment	16
Description	16
Recommendation	16
L02 - State Variables could be Declared Constant	17
Description	17
Recommendation	17
L04 - Conformance to Solidity Naming Conventions	18
Description	18

Recommendation	18
L16 - Validate Variable Setters	19
Description	19
Recommendation	19
<b>Functions Analysis</b>	<b>20</b>
<b>Inheritance Graph</b>	<b>23</b>
<b>Flow Graph</b>	<b>24</b>
<b>Summary</b>	<b>25</b>
<b>Disclaimer</b>	<b>26</b>
<b>About Cyberscope</b>	<b>27</b>

## Review

Contract Name	HENTAITOKEN
Testing Deploy	<a href="https://mumbai.polygonscan.com/address/0x06efacd71bec76ce88989774651f2b5c1925d791">https://mumbai.polygonscan.com/address/0x06efacd71bec76ce88989774651f2b5c1925d791</a>

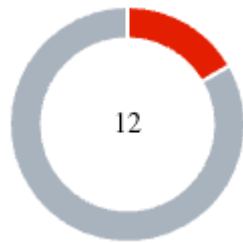
## Audit Updates

Initial Audit	21 Mar 2024
---------------	-------------

## Source Files

Filename	SHA256
contracts/hentai.sol	21d6603cca1b75912601a03eea2735efcdf2b94aee31068a8ca801ce3a6f29d5

## Findings Breakdown



Critical	2
Medium	0
Minor / Informative	10

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	2	0	0	0
Medium	0	0	0	0
Minor / Informative	10	0	0	0

## ST - Stops Transactions

Criticality	Critical
Location	contracts/hentai.sol#L229
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 not adding users to the `antisniper` mapping. As a result, these users will not be able to execute transactions until the contract has completed 30 buy transactions.

```
if (antiSniperEnabled && from != address(this)) {  
    require(antisniper[to], "Failed to snipe");  
}
```

### 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.



## EPC - Existing Pair Creation

Criticality	Critical
Location	contracts/hentai.sol#L296
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");
    uniswapV2Router =
    IUniswapV2Router02(0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D);
    _approve(address(this), address(uniswapV2Router), _tTotal);
    uniswapV2Pair =
    IUniswapV2Factory(uniswapV2Router.factory()).createPair(address(this),
    uniswapV2Router.WETH());
    uniswapV2Router.addLiquidityETH{value:
    address(this).balance}(address(this), balanceOf(address(this)), 0, 0, owner(), block.t
    imESTAMP);
    IERC20(uniswapV2Pair).approve(address(uniswapV2Router), type(uint).max);
    swapEnabled = true;
    tradingOpen = true;
    _tradingOpened = block.timestamp;
}
```

## 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.

## IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	contracts/hentai.sol#L163
Status	Unresolved

### Description

The contract declares state variables that their value is initialized once in the constructor and are not modified afterwards. The `immutable` is a special declaration for this kind of state variables that saves gas when it is defined.

```
_taxWallet
```

### Recommendation

By declaring a variable as immutable, the Solidity compiler is able to make certain optimizations. This can reduce the amount of storage and computation required by the contract, and make it more gas-efficient.

## MCM - Misleading Comment Messages

Criticality	Minor / Informative
Location	contracts/hentai.sol#L145
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.

```
uint256 private _taxSwapThreshold= 100000 * 10**_decimals; //0.01%
```

### 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.

## MEE - Missing Events Emission

<b>Criticality</b>	Minor / Informative
<b>Location</b>	contracts/hentai.sol#L304,311
<b>Status</b>	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.

```
tradingOpen = true;  
antisniper[accounts[i]] = true;
```

### 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.

## RED - Redudant Event Declaration

Criticality	Minor / Informative
Location	contracts/hentai.sol#L154
Status	Unresolved

### Description

The contract uses events that are not emitted within the contract's functions. As a result, these declared events are redundant and serve no purpose within the contract's current implementation.

```
event MaxTxAmountUpdated(uint _maxTxAmount);
```

### Recommendation

To optimize contract performance and efficiency, it is advisable to regularly review and refactor the codebase, removing the unused event declarations. This proactive approach not only streamlines the contract, reducing deployment and execution costs but also enhances readability and maintainability.

## RRS - Redundant Require Statement

Criticality	Minor / Informative
Location	contracts/hentai.sol#L24
Status	Unresolved

### Description

The contract utilizes a `require` statement within the `add` function aiming to prevent overflow errors. This function is designed based on the SafeMath library's principles. In Solidity version 0.8.0 and later, arithmetic operations revert on overflow and underflow, making the overflow check within the function redundant. This redundancy could lead to extra gas costs and increased complexity without providing additional security.

```
function add(uint256 a, uint256 b) internal pure returns (uint256) {  
    uint256 c = a + b;  
    require(c >= a, "SafeMath: addition overflow");  
    return c;  
}
```

### Recommendation

It is recommended to remove the `require` statement from the `add` function since the contract is using a Solidity pragma version equal to or greater than 0.8.0. By doing so, the contract will leverage the built-in overflow and underflow checks provided by the Solidity language itself, simplifying the code and reducing gas consumption. This change will uphold the contract's integrity in handling arithmetic operations while optimizing for efficiency and cost-effectiveness.

## RSML - Redundant SafeMath Library

Criticality	Minor / Informative
Location	contracts/hentai.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>.



## RVR - Redundant Variable Reassignment

Criticality	Minor / Informative
Location	contracts/hentai.sol#L242
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 assigns the same value to the variable `taxAmount` more than once in the transfer flow. As are result, the second assignment is redundant.

```
taxAmount = amount.mul(_tax).div(10000);
```

### 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.

## L02 - State Variables could be Declared Constant

Criticality	Minor / Informative
Location	contracts/hentai.sol#L117,118,119,120,125,126,127,128,129,145,146
Status	Unresolved

### Description

State variables can be declared as constant using the constant keyword. This means that the value of the state variable cannot be changed after it has been set. Additionally, the constant variables decrease gas consumption of the corresponding transaction.

```
uint256 private _tier1 = 4000
uint256 private _tier2 = 2000
uint256 private _tier3 = 1000
uint256 private _tier4 = 33
uint256 private _antiSniperCount = 30
uint256 private _reducingPeriod1 = 60
uint256 private _reducingPeriod2 = 90
uint256 private _reducingPeriod3 = 45 minutes
uint256 private _preventSwapBefore= 31
uint256 private _taxSwapThreshold= 100000 * 10**_decimals
uint256 private _maxTaxSwap= 28000000 * 10**_decimals
```

### Recommendation

Constant state variables can be useful when the contract wants to ensure that the value of a state variable cannot be changed by any function in the contract. This can be useful for storing values that are important to the contract's behavior, such as the contract's address or the maximum number of times a certain function can be called. The team is advised to add the constant keyword to state variables that never change.

## L04 - Conformance to Solidity Naming Conventions

<b>Criticality</b>	Minor / Informative
<b>Location</b>	contracts/hentai.sol#L62,100,116,132
<b>Status</b>	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 public _owner
function WETH() external pure returns (address);
uint256 public _tax = 0
uint256 public _tradingOpened
```

### 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>.

## L16 - Validate Variable Setters

<b>Criticality</b>	Minor / Informative
<b>Location</b>	contracts/hentai.sol#L67,163
<b>Status</b>	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.

```
_owner = msgSender  
_taxWallet = payable(taxWallet)
```

### 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.

## Functions Analysis

Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
<b>Context</b>	Implementation			
	_msgSender	Internal		
<b>IERC20</b>	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
<b>SafeMath</b>	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		

Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
<b>IUniswapV2Factory</b>	Interface			
	createPair	External	✓	-
<b>IUniswapV2Router02</b>	Interface			
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
	factory	External		-
	WETH	External		-
	addLiquidityETH	External	Payable	-
<b>HENTAITOKEN</b>	Implementation	Context, IERC20, Ownable		
		Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-

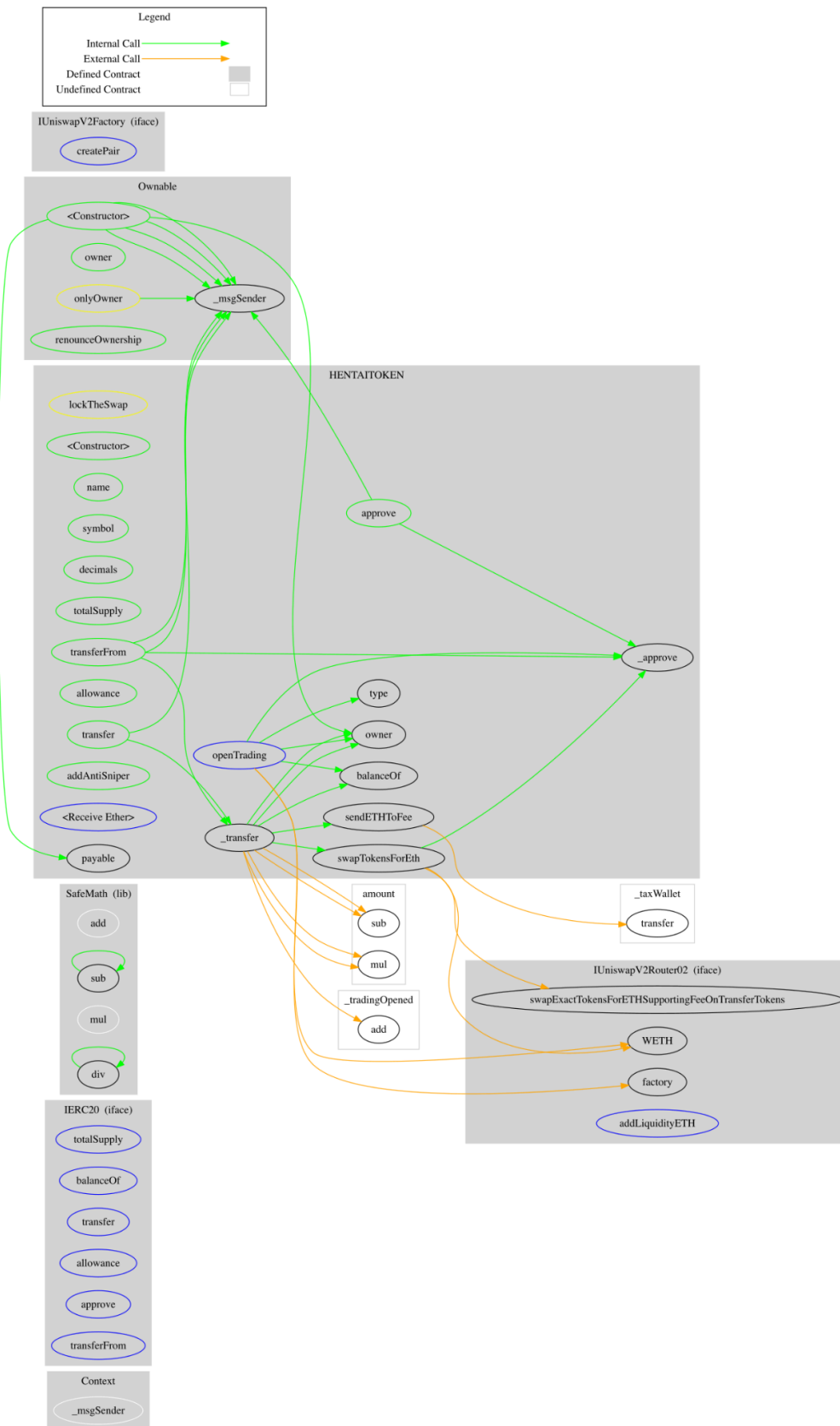
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	_approve	Private	✓	
	_transfer	Private	✓	
	swapTokensForEth	Private	✓	lockTheSwap
	sendETHToFee	Private	✓	
	openTrading	External	✓	onlyOwner
	addAntiSniper	Public	✓	onlyOwner
		External	Payable	-

## Inheritance Graph





# Flow Graph



## Summary

hentai contract implements a token mechanism. This audit investigates security issues, business logic concerns, and potential improvements. hentai is an interesting project that has a friendly and growing community. The Smart Contract analysis reported no compiler errors 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 contract includes an automatic fee mechanism that adjusts the fees based on the number of buy transactions that have been executed.

1. The fee starts at 0% until 30 buy transactions have been completed.
2. Then the fee goes up to 40% until 60 buy transactions have been completed.
3. Then the fee is reduced to 20% until 60 buy transactions have been completed.
4. Then the fee is reduced to 10% until 90 buy transactions have been completed.
5. Finally, the fee is locked at 0.33% 45 minutes after the trading is enabled and the fee has reached the 10% stage.

## 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>