MARSWAP

Hoichi芳一 - (HOICHI)

0xC4EE0aA2d993ca7C9263eCFa26c6f7e13009d2b6

MARSWAP Audit provided by Fintech Global Services

Audit Tools used include Manticore, Visual Code, Mythril, and Remix

Table of Contents

SUMMARY OF AUDIT
DETAILS OF AUDITED PROJECT

Audited Project:	Hoichi 芳一 (HOICHI)
Source Code:	https://etherscan.io/address/0xc4ee0aa2d993ca7c9263ecfa26c6f7e13009d2b6#code (verified)
Solidity File:	HOICHI.sol
Security Audit Date:	Dec. 18 - 2023
Revisions:	Initial Audit Dec.18.2023
Auditing Methods:	Automatic review + Manual review

AUDITING METHODS AND COVERING SECTORS

- Evaluation objective for the security audit:
- Quality of smart contracts code
- Issues and vulnerabilities with security
- Documentation, project specifics and commenting on smart contract
- Correctness of specifications regarding the use-case
- Following the best practices on smart contract

Audit covers these sectors of smart contract for possible vulnerabilities, issues and recommendations for better practices in case of severe or medium issues:

- **Dependence Transaction Order**
- **Single and Cross-Function Reentrancy**
- **Time Dependency**
- **Integer Overflow**
- **Integer Underflow**
- **Mishandled exceptions and call stack lim its**
- **Unsafe external calls**
- **Number rounding errors**
- **Insufficient gas issues**
- **Logical oversights**
- **Access control**
- **Centralization of power**
- **Logic-Specification**
- **Contradiction**
- **Functionality duplication**
- **Malicious contract behaviour and abusable functions**
- **Possible DoS vulnerabilities**

The code review conducted for this audit follows the following structure:

1. Review of the specifications, documentation and commenting providedby the project owners regarding the functionality of the smart contract

2.Automated analysis of the smart contract followed by manual, line-by-line analysis of the smart contract

3.Assessment of smart contract's correctness regarding the documentation and commenting compared to functionality

4.Assessment of following the best practices

5.Recommendations for better practices in case severe or medium vulnerabilities

SMART CONTRACT DETAILS 0xC4EE0aA2d993ca7C9263eCFa26c6f7e13009d2b6

Contract Address:	(verified)
Blockchain:	Ethereum
Language Used:	Solidity
Compiler Version:	v0.8.16+commit.07a7930e
Etherscan Verification:	2022-08-25 (verified)
Type of Smart Contract:	ERC20 Token - Standard Token
Libraries Used:	
Optimization Enabled:	NO with 200 runs

Number of Interfaces: 6
Number of Contracts: 4
Solidity Versions: ^0.8.0 - 0.8.16
Total Lines: 1099
Sell Tax: (2% Contract) 2%
Buy Tax: (1% Contract) 1%
Adjustable Taxes: NO
Total Supply: 369,369,369,369
Circulating Supply: 99,6%
Contract is Proxy: NO
Blacklist Functions: NO
Can Mint: NO
Pausable: NO
Can Limit TX amount: No

SUMMARY OF AUDIT RESULTS

Marswap Security Scope smart contract audit for Hoichi 芳一 (HOICHI) is marked as PASSED result without severe issues on the contract logic and functions of the contract. Review of the project and description of the projects use-case as an digital asset and the contract itself follows the line of good practices. Contract is very well commented and gives the proper understanding for third parties. There are no severe- or lower level vulnerability findings, only informational notifications and suggestions.

CHANGEABLE VARIABLES AND SIDE NOTES

* Contract is not renounced
* Taxes cannot be adjusted or abused
* Development wallet address cannot be changed
* Liquidity tokens are burned

SEVERITY OF RISKS AND VULNERABILITIES

LOW-SEVERITY	MEDIUM-SEVERITY	HIGH-SEVERITY
0	0	0

REPORTED VULNERABILITIES, ISSUES AND INFORMATIONAL NOTES

LEVEL OF SEVERITY	Description	FILE	CODELINES AFFECTED
INFORMATIONAL	A floating pragma is set.	HOICHI	L: 17 C: 0 L: 44 C: 0

FINDINGS IN-DEPTH A FLOATING PRAGMA IS SET

The current pragma Solidity directive is^0.8.0.
It is recommended to specify a fixed and locked compiler version to ensure that the bytecode produced does not vary between the builds. This is especially important if you rely on bytecode-level verification of the code.

SUGGESTION FOR FUTURE REFERENCE

Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively. As the severity of the finding, there are no adjustments needed and the contract has been successfully deployed and marked as informational.

DOCUMENTATION AND COMMENTING

The contract is very well commented on and gives the proper understanding of the contract for developers and other third parties for the understanding in a human-readable format. Overall, the contract follows OpenZeppelin standard contracts and is very clean without any editions made for the core contracts.

CORRECTNESS OF SPECIFICATIONS

Smart contract follows the functionality that is stated in the documentation and description of the contract. The use case is also in line with what is described about the project and no adjustments can be made by the owner except burn tokens.

FOLLOWING THE BEST PRACTICES

The contract follows the best practices from all parts and there are no concerns from the auditor of any malicious use of the contract's functions as the contract is a non-adjustable OpenZeppelin ERC20 Burnable token. Well-commended and clean code.

HISTORY OF REVISIONS, FUNCTIONS AND VARIABLES

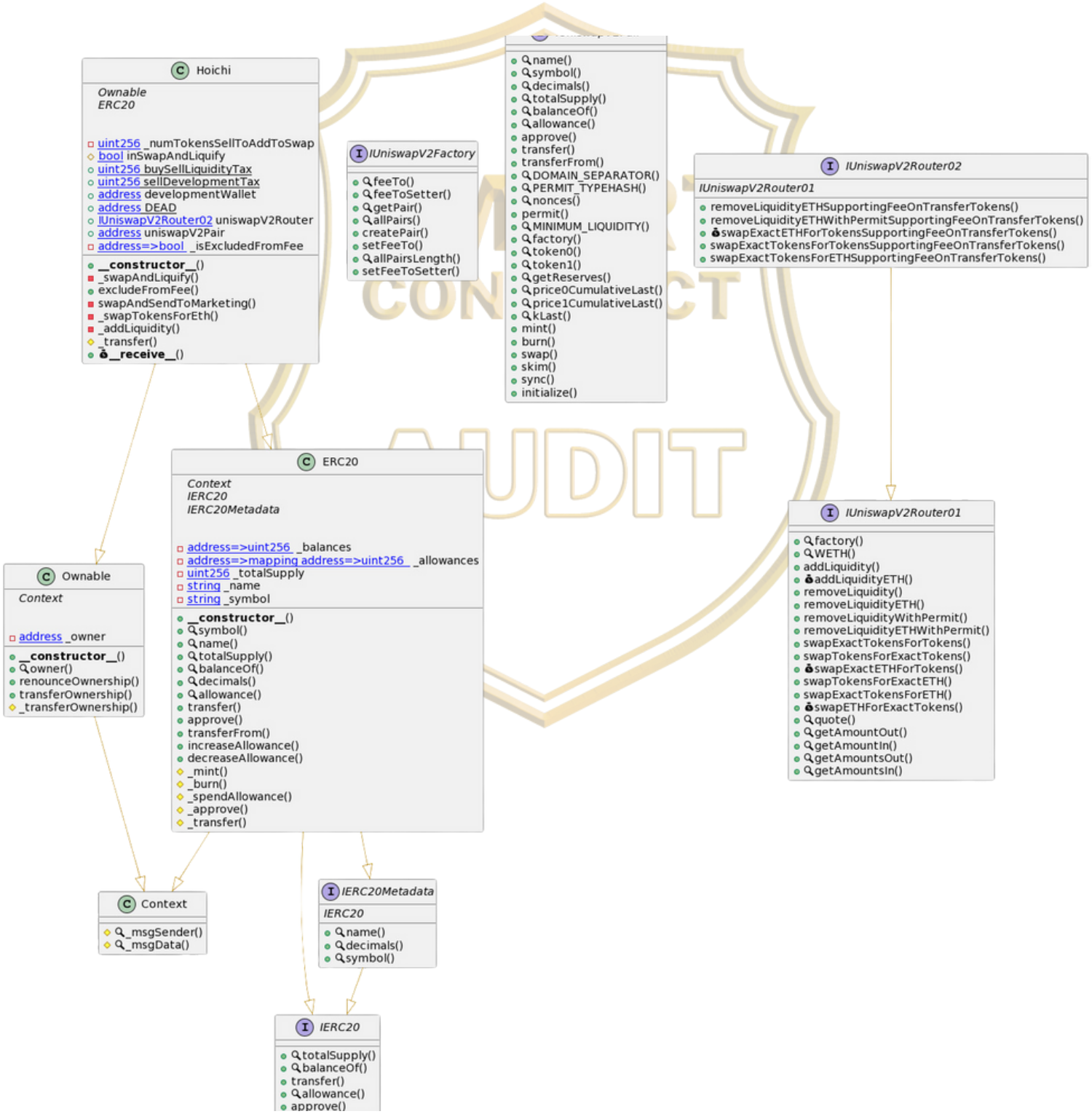
FUNCTIONS AND SIGNATURES

Sighash | Function Signature

==========

Function Name	Sighash	Function Signature
owner	8da5cb5b	owner()
renounceOwnership	715018a6	renounceOwnership()
transferOwnership	f2fde38b	transferOwnership(address)
feeTo	017e7e58	feeTo()
feeToSetter	094b7415	feeToSetter()
getPair	e6a43905	getPair(address,address)
allPairs	1e3dd18b	allPairs(uint256)
createPair	c9c65396	createPair(address,address)
setFeeTo	f46901ed	setFeeTo(address)
allPairsLength	574f2ba3	allPairsLength()
setFeeToSetter	a2e74af6	setFeeToSetter(address)
name	06fdde03	name()
symbol	95d89b41	symbol()
decimals	313ce567	decimals()
totalSupply	18160ddd	totalSupply()
balanceOf	70a08231	balanceOf(address)
allowance	dd62ed3e	allowance(address,address)
approve	095ea7b3	approve(address,uint256)
transfer	a9059cbb	transfer(address,uint256)
transferFrom	23b872dd	transferFrom(address,address,uint256)
DOMAIN_SEPARATOR	3644e515	DOMAIN_SEPARATOR()
PERMIT_TYPEHASH	30adf81f	PERMIT_TYPEHASH()
nonces	7ecebe00	nonces(address)
permit	d505accf	permit(address,address,uint256,uint256,uint8,bytes32,bytes32)
MINIMUM_LIQUIDITY	ba9a7a56	MINIMUM_LIQUIDITY()
factory	c45a0155	factory()
token0	0dfe1681	token0()
token1	d21220a7	token1()
getReserves	0902f1ac	getReserves()
price0CumulativeLast	5909c0d5	price0CumulativeLast()
price1CumulativeLast	5a3d5493	price1CumulativeLast()
kLast	7464fc3d	kLast()
mint	6a627842	mint(address)
burn	89afcb44	burn(address)
swap	022c0d9f	swap(uint256,uint256,address,bytes)
skim	bc25cf77	skim(address)
sync	fff6cae9	sync()
initialize	c4d66de8	initialize(address)
factory	c45a0155	factory()
WETH	ad5c4648	WETH()
addLiquidity	e8e33700	addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256)
addLiquidityETH	f305d719	addLiquidityETH(address,uint256,uint256,uint256,address,uint256)
removeLiquidity	baa2abde	removeLiquidity(address,address,uint256,uint256,uint256,address,uint256)
removeLiquidityETH	02751cec	removeLiquidityETH(address,uint256,uint256,uint256,address,uint256)
removeLiquidityWithPermit	2195995c	removeLiquidityWithPermit(address,address,uint256,uint256,uint256,address,uint256,bool,uint8,bytes32,bytes32)
removeLiquidityETHWithPermit	ded9382a	removeLiquidityETHWithPermit(address,uint256,uint256,uint256,address,uint256,bool,uint8,bytes32,bytes32)
swapExactTokensForTokens	38ed1739	swapExactTokensForTokens(uint256,uint256,address[],address,uint256)
swapTokensForExactTokens	8803dbee	swapTokensForExactTokens(uint256,uint256,address[],address,uint256)
swapExactETHForTokens	7ff36ab5	swapExactETHForTokens(uint256,address[],address,uint256)
swapTokensForExactETH	4a25d94a	swapTokensForExactETH(uint256,uint256,address[],address,uint256)
swapExactTokensForETH	18cbafe5	swapExactTokensForETH(uint256,uint256,address[],address,uint256)
swapETHForExactTokens	fb3bdb41	swapETHForExactTokens(uint256,address[],address,uint256)
quote	ad615dec	quote(uint256,uint256,uint256)
getAmountOut	054d50d4	getAmountOut(uint256,uint256,uint256)
getAmountIn	85f8c259	getAmountIn(uint256,uint256,uint256)
getAmountsOut	d06ca61f	getAmountsOut(uint256,address[])
getAmountsIn	1f00ca74	getAmountsIn(uint256,address[])
removeLiquidityETHSupportingFeeOnTransferTokens	af2979eb	removeLiquidityETHSupportingFeeOnTransferTokens(address,uint256,uint256,uint256,address,uint256)
removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	5b0d5984	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens(address,uint256,uint256,uint256,address,uint256,bool,uint8,bytes32,bytes32)
swapExactETHForTokensSupportingFeeOnTransferTokens	b6f9de95	swapExactETHForTokensSupportingFeeOnTransferTokens(uint256,address[],address,uint256)
swapExactTokensForTokensSupportingFeeOnTransferTokens	5c11d795	swapExactTokensForTokensSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)
swapExactTokensForETHSupportingFeeOnTransferTokens	791ac947	swapExactTokensForETHSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)
totalSupply	18160ddd	totalSupply()
balanceOf	70a08231	balanceOf(address)
transfer	a9059cbb	transfer(address,uint256)
allowance	dd62ed3e	allowance(address,address)
approve	095ea7b3	approve(address,uint256)
transferFrom	23b872dd	transferFrom(address,address,uint256)
name	06fdde03	name()
decimals	313ce567	decimals()
symbol	95d89b41	symbol()
symbol	95d89b41	symbol()
name	06fdde03	name()
totalSupply	18160ddd	totalSupply()
balanceOf	70a08231	balanceOf(address)
decimals	313ce567	decimals()
allowance	dd62ed3e	allowance(address,address)
transfer	a9059cbb	transfer(address,uint256)
approve	095ea7b3	approve(address,uint256)
transferFrom	23b872dd	transferFrom(address,address,uint256)
increaseAllowance	39509351	increaseAllowance(address,uint256)
decreaseAllowance	a457c2d7	decreaseAllowance(address,uint256)
excludeFromFee	df8408fe	excludeFromFee(address,bool)

SMART CONTRACT UML



HISTORY OF REVISIONS

Initial audit was performed December 18 2023 and no need for further revisions of the
smart contract audit.

Team has been informed of a clean audit result and the smart contract follows all the golden
standards without actual vulnerabilities and very minimal informational notes. Notifications are
informative for future reference.

The team has been adviced to keep the privacy keys for the development wallet in safe location at all times as the development wallet is not changeable from the contract functions as its set as an static address directly on the contract which takes place during the deployment of the contract. Only way to change the development wallet of the project is to re-deploy the contract which is not efficient investor wise, or the project team wise.