

MARSWAP



SECURITY AUDIT REPORT FOR

HEEL (Good Dog) Audit

0xf941D3AAbf2EE0F5589E68Ba6047b8329592B366

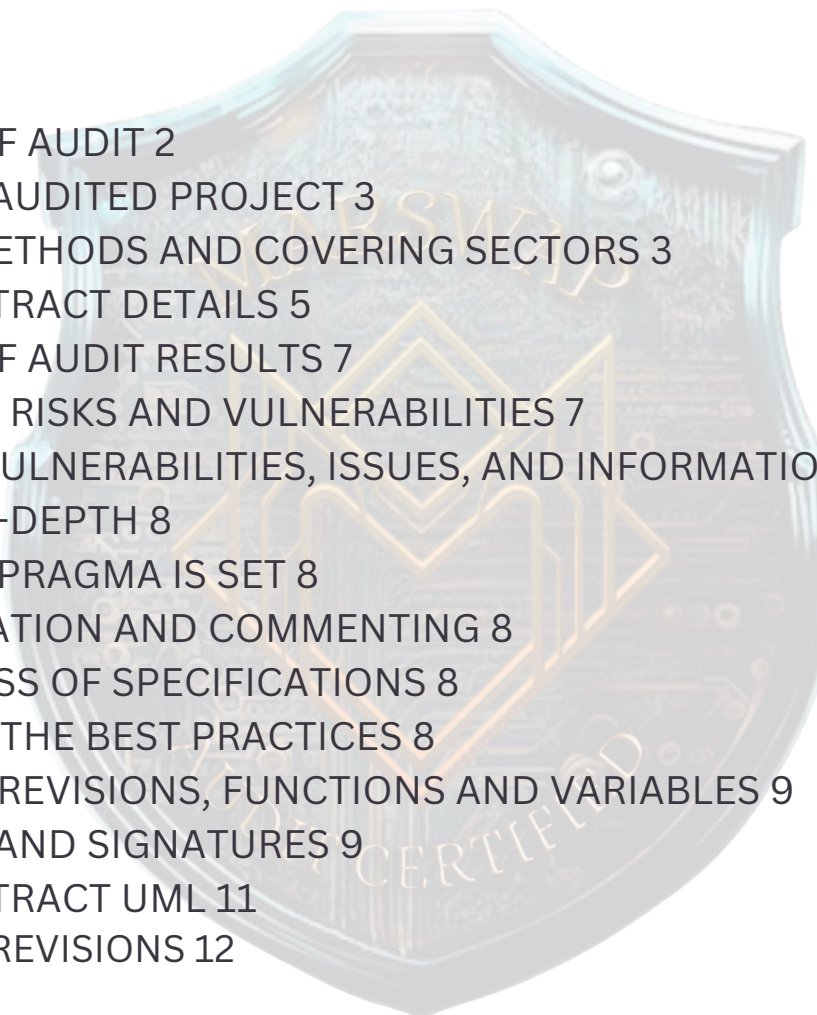
MARSWAP Audit provided by Fintech Global Services

Audit Tools used include

Manticore, Visual Code, Mythril, and Remix



Table of Contents



SUMMARY OF AUDIT	2
DETAILS OF AUDITED PROJECT	3
AUDITING METHODS AND COVERING SECTORS	3
SMART CONTRACT DETAILS	5
SUMMARY OF AUDIT RESULTS	7
SEVERITY OF RISKS AND VULNERABILITIES	7
REPORTED VULNERABILITIES, ISSUES, AND INFORMATIONAL NOTES	7
FINDINGS IN-DEPTH	8
A FLOATING PRAGMA IS SET	8
DOCUMENTATION AND COMMENTING	8
CORRECTNESS OF SPECIFICATIONS	8
FOLLOWING THE BEST PRACTICES	8
HISTORY OF REVISIONS, FUNCTIONS AND VARIABLES	9
FUNCTIONS AND SIGNATURES	9
SMART CONTRACT UML	11
HISTORY OF REVISIONS	12



DETAILS OF AUDITED PROJECT

Audited Project:	HEEL (Good Dog)
Source Code:	https://etherscan.io/token/0xf941d3aabf2ee0f5589e68ba6047b8329592b366#code (verified)
Solidity File:	GoodDog.sol
Security Audit Date:	Aug. 27 - 2023
Revisions:	Initial Audit Aug.27.2023
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 limits
 - 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 provided by 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

Contract Address:	0xf941D3AAbf2EE0F5589E68Ba6047b8329592B366 (verified)
Blockchain:	Ethereum
Language Used:	Solidity
Compiler Version:	v0.8.9+commit.e5eed63a
Etherscan Verification:	2022-08-31
Type of Smart Contract:	ERC20 Token – Reflection Token
Libraries Used:	SafeMath
Optimization Enabled:	No with 200 runs



Number of Interfaces: 3

Number of Contracts: 3

Solidity Versions: ^0.8.9

Total Lines: 585

Sell Tax: (0% Contract) 0%

Buy Tax: (0% Contract) 0%

Adjustable Taxes: Contract YES , Contract Renounced – Team cannot abuse taxes

Total Supply: 1,000,000,000

Circulating Supply: 98%

Contract is Proxy: NO

Blacklist Functions: NO

Can Mint: NO

Pausable: NO

Can Limit TX amount: Contract YES, Contract Renounced – Team cannot tx amount

SUMMARY OF AUDIT RESULTS

Marswap Security Scope smart contract audit for Good Dog (HEEL) is marked as a PASSED result without severe issues on the contract logic and functions of the contract. Review of the project and description of the project's use-case as a digital asset and the contract itself follows the line of good practices for the majority. Contract commenting could be improved to improve understanding of the contract functions from a third-party perspective. There are no severe- or lower-level vulnerability findings in the contract and the contract is renounced, so no abusable functions can be executed by the team.

CHANGEABLE VARIABLES AND SIDE NOTES

*Contract is Renounced and no abusable contract functions can be run by the team

*Owner can limit tax amount – Contract Renounced

*Owner can adjust taxes – Contract Renounced

SEVERITY OF RISKS AND VULNERABILITIES

LOW-SEVERITY

0

MEDIUM-SEVERITY

0

HIGH-SEVERITY

0

REPORTED VULNERABILITIES, ISSUES AND INFORMATIONAL NOTES

LEVEL OF SEVERITY	Description	FILE	CODELINES AFFECTED
INFORMATIONAL	A floating pragma is set.	GoodDog.sol	L: 14 C: 0

FINDINGS IN-DEPTH

A FLOATING PRAGMA IS SET

The current pragma Solidity directive is `^0.8.9`.

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 code has minimal to no amount of comments. Improving the stage of commenting on smart contracts, and cleaning the commenting, helps the user base to understand all the functionalities and use cases of the contract.

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 for a renounced contract.

FOLLOWING THE BEST PRACTICES

The contract follows the best practices in the majority and there are no concerns from the auditor of any malicious use of the contract's functions as the contract is renounced. Improving commenting on the contract improves readability and understanding of the contract from a third party perspective.

*Forks of this contract has by default adjustable taxes and adjustable transaction amounts



HISTORY OF REVISIONS, FUNCTIONS AND VARIABLES

FUNCTIONS AND SIGNATURES

Sighash | Function Signature

```
92704057 => _getTValues(uint256,uint256,uint256)
119df25f => _msgSender()
18160ddd => totalSupply()
70a08231 => balanceOf(address)
a9059cbb => transfer(address,uint256)
dd62ed3e => allowance(address,address)
095ea7b3 => approve(address,uint256)
23b872dd => transferFrom(address,address,uint256)
8da5cb5b => owner()
715018a6 => renounceOwnership()
f2fde38b => transferOwnership(address)
771602f7 => add(uint256,uint256)
b67d77c5 => sub(uint256,uint256)
e31bdc0a => sub(uint256,uint256,string)
c8a4ac9c => mul(uint256,uint256)
a391c15b => div(uint256,uint256)
b745d336 => div(uint256,uint256,string)
c9c65396 => createPair(address,address)
791ac947 =>
swapExactTokensForETHSupportingFeeOnTransferTokens(
uint256,uint256,address[],address,uint256)
c45a0155 => factory()
ad5c4648 => WETH()
f305d719 =>
addLiquidityETH(address,uint256,uint256,uint256,address,
uint256)
06fdde03 => name()
95d89b41 => symbol()
313ce567 => decimals()
2d838119 => tokenFromReflection(uint256)
301370af => removeAllFee()
e7e3e3a7 => restoreAllFee()
104e81ff => _approve(address,address,uint256)
30e0789e => _transfer(address,address,uint256)
b28805f4 => swapTokensForEth(uint256)
06b50197 => sendETHToFee(uint256)
8f70ccf7 => setTrading(bool)
c3c8cd80 => manualswap()
6fc3eaec => manualsend()
00b8cf2a => blockBots(address[])
6b999053 => unblockBot(address)
b09bbc79 =>
```



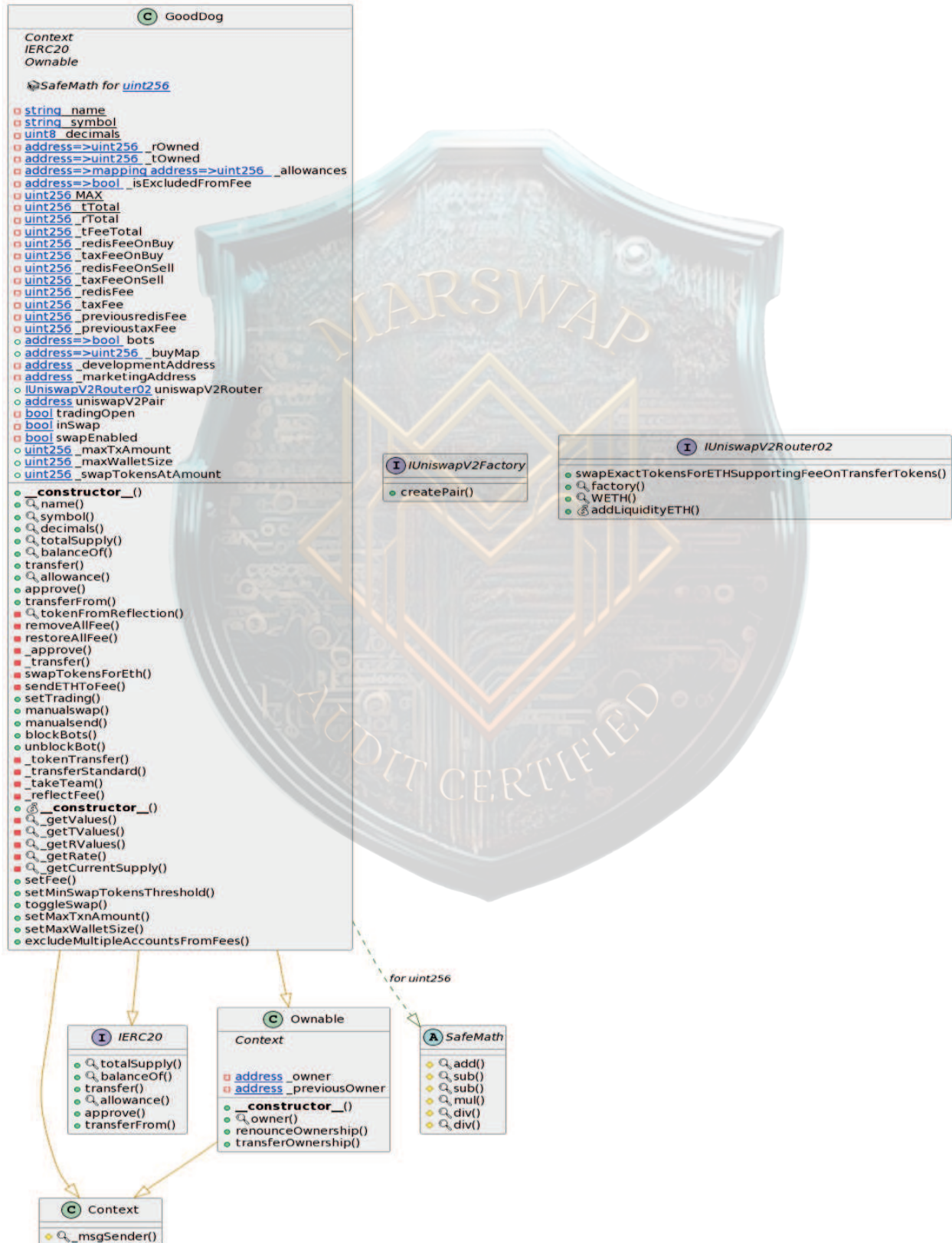
MARSWAP
SAFETY & SECURITY

```
_tokenTransfer(address,address,uint256,bool)
2852df65 => _transferStandard(address,address,uint256)
05f3c41f => _takeTeam(uint256)
184d894e => _reflectFee(uint256,uint256)
d4780e36 => _getValues(uint256)
1d5671e4 => _getRValues(uint256,uint256,uint256,uint256)
94e10784 => _getRate()
97a9d560 => _getCurrentSupply()
a2a957bb => setFee(uint256,uint256,uint256,uint256)
98a5c315 => setMinSwapTokensThreshold(uint256)
6d8aa8f8 => toggleSwap(bool)
74010ece => setMaxTxnAmount(uint256)
ea1644d5 => setMaxWalletSize(uint256)
c492f046 => excludeMultipleAccountsFromFees(address[],bool)
```





SMART CONTRACT UML





MARSWAP
SAFETY & SECURITY

HISTORY OF REVISIONS

The initial audit was performed in August. 27-2023 and no need for further revisions of the smart contract audit.

The team has been informed of a clean audit result and in the majority, the smart contract follows all the golden standards without actual vulnerabilities. Notifications are informative for future reference.

Contract commenting could be improved to improve understanding of the contract functions from a third-party perspective.

