MARSWAP

SECURITY AUDIT REPORT FOR

Sigma Token Audit

0xBD1EfB3643C9F3c679eFf55eEa332ECd7427EF74

MARSWAP Audit provided by Fintech Global Services

Audit Tools used include

Manticore, Visual Code, Mythril, and Remix

MARSWAP
SAFETY & SECURITY

SUMMARY OF AUDIT

Details of Audit

Audit Project	**Sigma Token - (SGMA)**
Source Code	https://etherscan.io/token/0xBD1EfB3643C9F3c679eFf55eEa332ECd7427EF74#code (verified)
Solidity File	Sigma.sol
Security Audit Date	August 12, 2023
Revisions	Initial Audit: August 12, 2023
Auditing Methods	Automatic review - Manual review

MARSWAP
SAFETY & SECURITY

Table of Contents

MARSWAP
SAFETY & SECURITY

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:

- Review of the specifications, documentation and commenting provided by the project owners regarding the functionality of the smart contract
- Automated analysis of the smart contract followed by manual, line-by-line analysis of the smart contract
- 3Assessment of smart contracts correctness regarding the documentation and commenting compared to functionality
- Assessment of following the best practices
- Recommendations for better practises in case severe or medium vulnerabilities

MARSWAP
SAFETY & SECURITY

Smart Contract Details

Contract Adress	0xBD1EfB3643C9F3c679eFf55eEa332ECd7427EF74 (verified)
Blockchain	Ethereum
Language Used	Solidity
Compiler Version	v0.8.11+commit.d7f03943
Etherscan Verification	2023-08-06
Type of Smart Contract	ERC20 Token – Standard Token
Libraries Used	
Optimization Enabled	No with 200runs

MARSWAP
SAFETY & SECURITY

Smart Contract Details

Number of Interfaces: 4
Number of Contracts: 5
Solidity Versions: ^0.8.8
Total Lines: 721
Sell Tax: *(4% Contract)* 4%
Buy Tax: *(4% Contract)* 4%
Adjustable Taxes: *YES* , Max 12%
Total Supply: 1,000,000,000
Circulating Supply: 100%
Contract is Proxy: NO
Blacklist Functions: NO
Can Mint: NO
Pausable: NO
Can Limit TX amount: NO

MARSWAP
SAFETY & SECURITY

Summary of Audit Results

Marswap Security Scope smart contract audit for Sigma Tokenis 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 for majority. Mostly and default commented contract. There are no severe vulnerability findings in the contract and no concerns about malicious use of the contract functions.

CHANGEABLE VARIABLES AND SIDENOTES

- Owner can whitelist addresses from fees
- Owner can change contract taxes in reasonable rates
- Contract has recovery functions for stuck natives and tokens

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	Code Lindes Affected
INFORMATIONAL	A floating pragma is set.	sigma.sol	L: 3 C: 0
INFORMATIONAL	Potential use of "block.number" as source of randonmness.	sigma.sol	L: 507 C: 12
INFORMATIONAL	Potential use of "block.number" as source of randonmness.	sigma.sol	L: 661 C: 24

Findings in Depth

A FLOATING PRAGMA IS SET

The current pragma Solidity directive is ^0.8.8.
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 with 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 an severity of the finding, there is no adjustments needed and contract has been successfully deployed and marked as informational

POTENTIAL USE OF “BLOCK.NUMBER” AS SOURCE OF RANDOMNESS

The environment variable "block.number" is used as a source of randomness. Note that the values of variables like gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Popular use-cases for usage of “block.number” variable are executing Gambling and Raffle DApps to pick an winner randomly using pseudo-random generator.

Contract does not perform lotteries or gambling, so no adjustments are needed in the smart contract and stated as informational sidenote.

DOCUMENTATION AND COMMENTING

The code has a decent amount of comments and documentation. Improving stage of commenting on smart contracts, and cleaning the commenting, helps userbase to understand all the functionalities and use-case of the contract. Functions are marked in understandable way and commenting on the contract has been made human readable to understand.

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 what is described about the project.

FOLLOWING THE BEST PRACTICES

The contract follows the best practices in majority and there isno concerns from the auditor of any malicious use of the contracts functions and range of adjustable taxes are within reasonable rates.

HISTORY OF REVISIONS, FUNCTIONS AND VARIABLES

FUNCTIONS AND SIGNATURES

Sighash | Function Signature

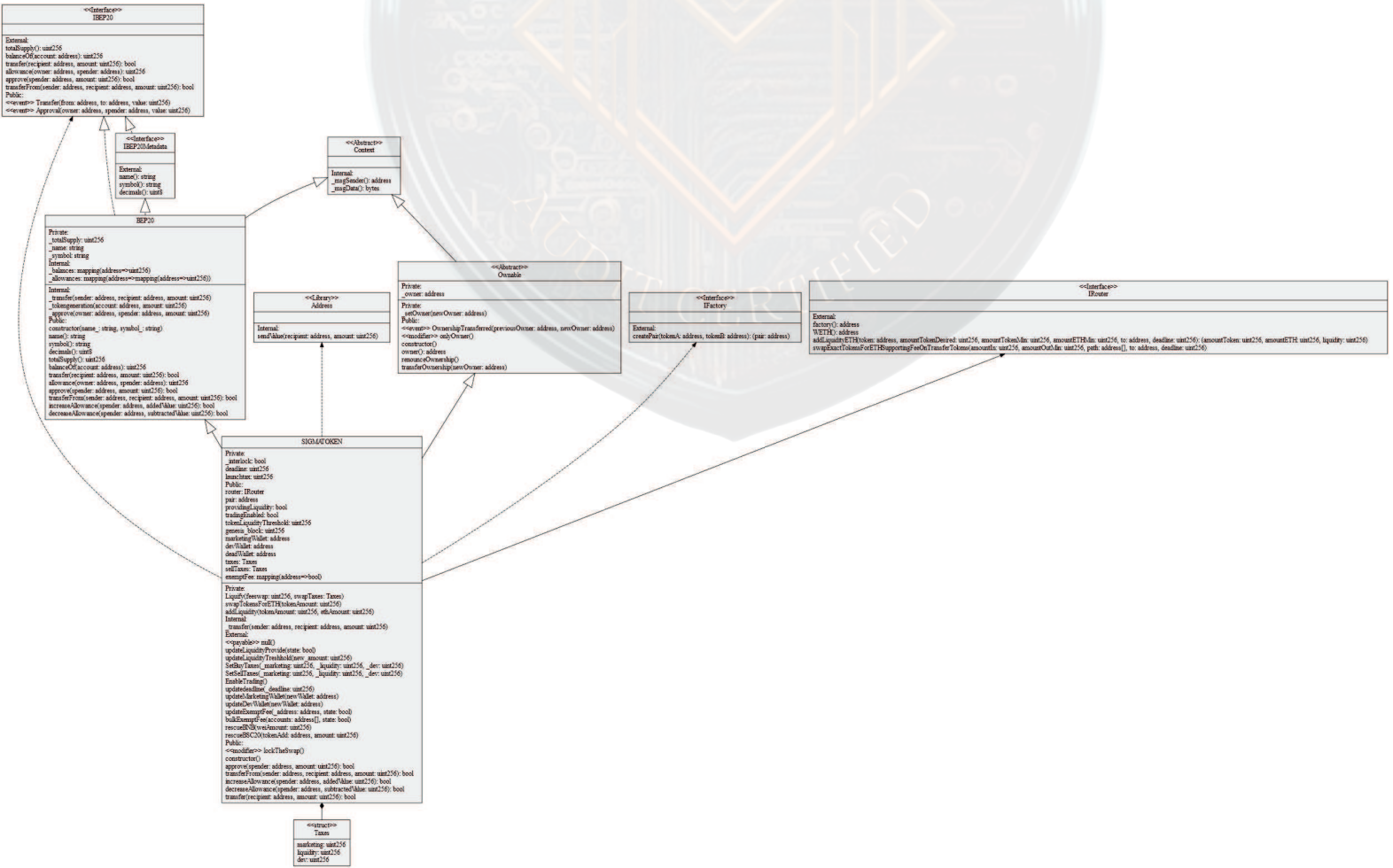
39509351 => increaseAllowance(address,uint256)
119df25f => _msgSender()
8b49d47e => _msgData()
18160ddd => totalSupply()
70a08231 => balanceOf(address)
a9059cbb => transfer(address,uint256)
dd62ed3e => allowance(address,address)
095ea7b3 => approve(address,uint256)
23b872dd => transferFrom(address,address,uint256)
06fdde03 => name()
95d89b41 => symbol()
313ce567 => decimals()
a457c2d7 => decreaseAllowance(address,uint256)
30e0789e => _transfer(address,address,uint256)
c143c0de => _tokengeneration(address,uint256)
104e81ff => _approve(address,address,uint256)
24a084df => sendValue(address,uint256)
8da5cb5b => owner()
715018a6 => renounceOwnership()
f2fde38b => transferOwnership(address)
fc201122 => _setOwner(address)
c9c65396 => createPair(address,address)
c45a0155 => factory()
ad5c4648 => WETH()
f305d719 => addLiquidityETH(address,uint256,uint256,uint256,address,uint256)

Sighash | Function Signature

791ac947 => swapExactTokensForETHSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)

464244d9 => Liquify(uint256,Taxes)
e56a645e => swapTokensForETH(uint256)
9cd441da => addLiquidity(uint256,uint256)
1340538f => updateLiquidityProvide(bool)
42b6fa11 => updateLiquidityTreshhold(uint256)
4a2b3b13 => SetBuyTaxes(uint256,uint256,uint256)
4e736f22 => SetSellTaxes(uint256,uint256,uint256)
1d97b7cd => EnableTrading()
edaa1168 => updatedeadline(uint256)
aacebbe3 => updateMarketingWallet(address)
1816467f => updateDevWallet(address)
355496ca => updateExemptFee(address,bool)
0e375a5c => bulkExemptFee(address[],bool)
441b1d30 => rescueBNB(uint256)
c9300ed5 => rescueBSC20(address,uint256)

SMART CONTRACT UML



History of Revisions

Initial audit was performed August 12, 2023 and no need for further revisions of the smart contract audit. Team has been informed of a clean audit result and in majority the smart contract follows all the golden standards without actual vulnerabilities. Notifications are informative for future reference.