TOKEN AUDIT

SAFE GRAVITY

Smart Contract Security Audit

Audit Report

Sept, 2021

Table of contents

Project Introduction

SAFEGRAVITY (SAG) is a gravity-free upcoin that will continuously rise in price, uncorrelated to the crypto market as a whole. Implemented in the token is a 10% transaction tax for LP acquisition, token buyback with burning and marketing provision. Just over 100 days after launch SAG will reach its ceiling price of 1 USD. The question is then posed whether SAG inherent value will inherit its price action. Whether this happens or not depends purely on speculative forces and market psychology. We can assume that there may be some correlation to the pegged price returns, although as with any speculative asset it cannot be known ahead of time.

Name	SAFE GRAVITY
Total Supply	1,000,000,000,000,000
Type	BSC Token
Website	**https://www.safegravity.io/**
Platform	Binance Smart Chain
Deployed Contract	0x394491b6D8016C233435e1BDcb8B41a7a0c66408

Token Audit Team performed a security audit for SAFE GRAVITY smart contracts during the period of Sep 28, 2021 to Sep 29, 2021.

The code for the audit was taken from following the official link:

https://bscscan.com/address/0x394491b6D8016C233435e1BDcb8B41a7a0c66408#code

Auditing Methodologies applied:

- In this audit, we can review the code listed below.
- The overall quality of code.
- Whether the implementation of BEP 20 standards.
- Whether the code is secure.
- Gas Optimization
- Code is safe from reentrancy and other vulnerabilities

Manual Audit

- Manually analyzing the source code line-by-line in an attempt to identify security vulnerabilities.
- Gas Consumption and optimization
- Assessing the overall project structure, complexity & quality.
- Checking whether all the libraries used in the code of the latest version.

Automated Audit

- Projects can be Automated using these tools with Slither, Manticore, Sol Graph others.
- Performing Unit testing.
- Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.

➔ Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.

Static Analysis

Static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step a series of automated tools are used to test security of smart contracts.

Auditing Tools

Language: Solidity

Platform and tools: Slither, Manti-Core, VScode, Solhint, Solc-select, Solidity-coverage

Audit Aim: The focus of the audit was to verify whether the smart contract is secure, resilient, and working according to the standard specs. The audit activity can be categories into three types

- Security
- Sound Architecture
- Code Correctness and Quality

Tokenomics:

The SAFEGRAVITY Token is a BEP-20 token launched on the Binance Smart Chain. The token will be launched using the DxSale's DxLaunch decentralized launch platform. The SAFEGRAVITY token keeps all tokens safe through the active implementation of DxSale's DxLock wallet locking tool for liquidity pool locking as well as gnosis multisig protocol wallets for increased security.

Initial Token Distribution:

45% - PancakeSwap Initial Liquidity

40% - Public DxSales Pre-sale

5% - Giveaways / Airdrops

5% - Project Development Wallet (Lock for a minimun of 3 months) this will be distributed in 3 wallets:

- 2% for Dex Staking Partners

- 2% for Marketing Partners

- 1% for Team Distribution

5% - Emergency Use (lock for 100 days, our DEX will be up and running at this time and this funds will be used just in emergncy case)

Our top priority at launch will be having the SAFEGRAVITY token listed on centralized exchanges to help new users join us! If we reach our full DxSale goal a big % will be dedicated to exchange listings early in the project.

Issues Checking

We have scanned this smart contract code for commonly known and more specific Vulnerabilities that are below listed:

SN	Issue Description	Status
1	Re-entrancy	Verified
2	Compiler errors	Verified
3	Timestamp Dependence	Verified
4	Unsafe external calls	Verified
5	Gas Limit and Loops	Verified
6	DoS with Block Gas Limit	Verified
7	Private user data leaks	Verified
8	Code clones, functionality duplication	Verified
9	Style guide violation	Verified
10	Costly Loop	Verified
11	Balance equality	Verified
12	Unchecked math	Verified

13	Integer overflow/underflow	Verified
14	Cross-function Race Condition	Verified
15	Fallback function security	Verified
16	Data Consistency	Verified
17	Balance equality	Verified
18	ERC20 API violation	Verified
19	Deployment Consistency	Verified
20	Arithmetic accuracy	Verified
21	Transaction-Ordering Dependence (TOD) / Front Running	Verified
22	Address hardcoded	Verified
23	Scoping and Declarations	Verified
24	Implicit visibility level	Verified
25	Call Depth Attack (deprecated)	Verified

Severity Issue Categories

Every issue in this report was assigned a severity level from the following:

Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc.
High	The issue affects the ability of the contract to compile or operate in a significant way.
Medium	Issues on this level could potentially bring problems and should eventually be fixed.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that can't have a significant impact on execution.
Informational	The issue has no impact on the contract's ability to operate.

Issues Found

- Critical severity issues - No Critical severity issues found.
- High severity issues - No Critical severity issues found.
- Medium severity issues - No medium severity issues found.
- Low severity issues - No low severity issues were found.
- Informational - 5 Informational severity issues were found.

High	Medium	Low	Informational
0	0	0	5

Informational level severity issues

1. Missing zero address validation

   Severity: Informational

   Description:

   updating the marketing address, it should be checked for zero address.

```
ftrace | funcSig
function setMarketingWallet(address payable newWallet↑) external onlyOwner{
    marketingAddress = newWallet↑;
}
```

```
ftrace
constructor (address payable _marketingAddress↑) ERC20("SafeGravity", "SAG") {
    marketingAddress = _marketingAddress↑;

    IRouter _router = IRouter(0x10ED43C718714eb63d5aA57B78B54704E256024E);
```

Suggestion:

Use a require statement to check for zero address when updating the marketing address

2. Conformance-To-Solidity-Naming-Conventions

Severity: Informational

Description:

In the contract, many function names were found to be starting with capital letters. Functions other than constructors should use mixed Case

```
Function IRouter.WETH() (SafeGravity.sol#41) is not in mixedCase
Parameter SAG.setTaxes(uint256,uint256,uint256)._lpTax (SafeGravity.sol#539) is not in mixedCase
Parameter SAG.setTaxes(uint256,uint256,uint256)._marketTax (SafeGravity.sol#539) is not in mixedCase
Parameter SAG.setTaxes(uint256,uint256,uint256)._buybackTax (SafeGravity.sol#539) is not in mixedCase
Parameter SAG.setMaster(address)._master (SafeGravity.sol#550) is not in mixedCase
Parameter SAG.setLP(address)._lp (SafeGravity.sol#560) is not in mixedCase
Parameter SAG.setSwapAndLiquifyEnabled(bool)._enabled (SafeGravity.sol#577) is not in mixedCase
Parameter SAG.calculateFee(uint256)._amount (SafeGravity.sol#678) is not in mixedCase
Parameter SAG.enableTransfer(address)._addr (SafeGravity.sol#874) is not in mixedCase
Parameter SAG.excludeAddress(address)._addr (SafeGravity.sol#881) is not in mixedCase
Parameter SAG.setBuyBackEnabled(bool)._enabled (SafeGravity.sol#888) is not in mixedCase
Parameter SAG.setBuyBackLimit(uint256)._buybackLimit (SafeGravity.sol#892) is not in mixedCase
Parameter SAG.setBuyBackDivisor(uint256)._buybackDivisor (SafeGravity.sol#895) is not in mixedCase
Parameter SAG.setnumTokensSellDivisor(uint256)._numTokensSellDivisor (SafeGravity.sol#898) is not in mixedCase
Parameter SAG.setMaxTxDivisor(uint256)._maxTxDivisor (SafeGravity.sol#902) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

Rule exceptions:

Allow constant variable name/symbol/decimals to be lowercase (ERC20).

Allow _ at the beginning of the mixed_case match for private variables and unused parameters.

Suggestion:

Follow the Solidity: https://docs.soliditylang.org/en/v0.4.25/style-guide.html#naming-conventions

3. State variables that could be declared constant

Severity: Informational

```
SAG.deadAddress (SafeGravity.sol#450) should be constant
SAG.privateSaleDropCompleted (SafeGravity.sol#459) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
```

Description:

The above constant state variables should be declared constant to save gas.

Suggestion:

Add the constant attributes to state variables that never change

4. Public function that could be declared external

Severity: Informational

Description:

The following public functions that are never called by the contract should be declared external to save gas

```
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (SafeGravity.sol#205-207)
        - SAG.allowance(address,address) (SafeGravity.sol#836-843)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (SafeGravity.sol#216-219)
        - SAG.approve(address,uint256) (SafeGravity.sol#819-827)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (SafeGravity.sol#234-242)
        - SAG.transferFrom(address,address,uint256) (SafeGravity.sol#599-607)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (SafeGravity.sol#256-259)
        - SAG.increaseAllowance(address,uint256) (SafeGravity.sol#788-796)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (SafeGravity.sol#275-281)
        - SAG.decreaseAllowance(address,uint256) (SafeGravity.sol#851-865)
setSwapAndLiquifyEnabled(bool) should be declared external:
        - SAG.setSwapAndLiquifyEnabled(bool) (SafeGravity.sol#577-580)
setBuyBackEnabled(bool) should be declared external:
        - SAG.setBuyBackEnabled(bool) (SafeGravity.sol#888-890)
setBuyBackLimit(uint256) should be declared external:
        - SAG.setBuyBackLimit(uint256) (SafeGravity.sol#892-893)
setBuyBackDivisor(uint256) should be declared external:
        - SAG.setBuyBackDivisor(uint256) (SafeGravity.sol#895-896)
setnumTokensSellDivisor(uint256) should be declared external:
        - SAG.setnumTokensSellDivisor(uint256) (SafeGravity.sol#898-900)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

Suggestion:

Use the external attribute for functions never called from the contract.

5. pragma solidity ^0.8.4

Severity: Informational

Description:

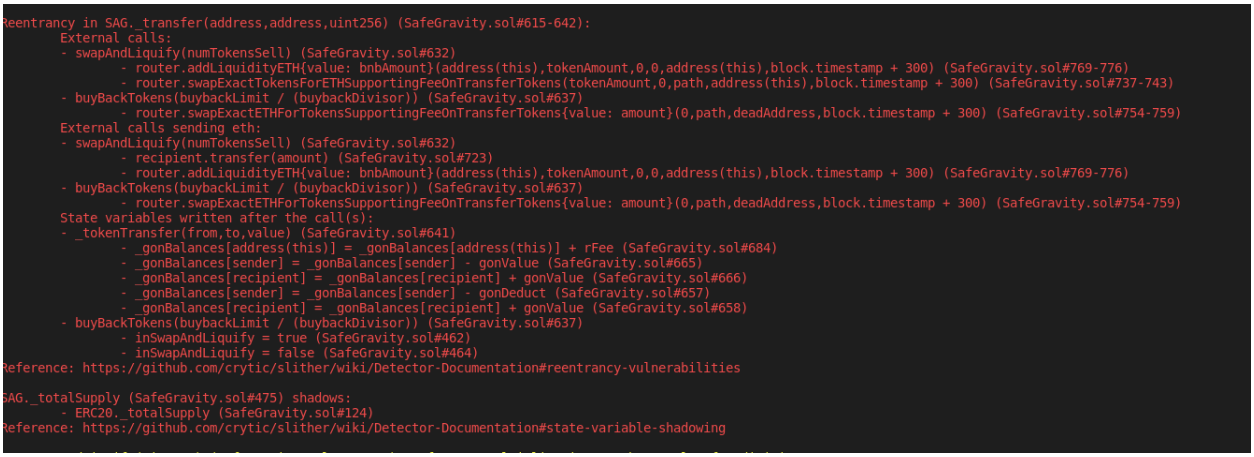
Contracts should be deployed using the same compiler version/flags with which they have been tested. Locking the pragma (for e.g., by not using ^ in pragma solidity 0.8.0) ensures that contracts do not accidentally get deployed using an older compiler version with unfixed bugs.

Suggestion:

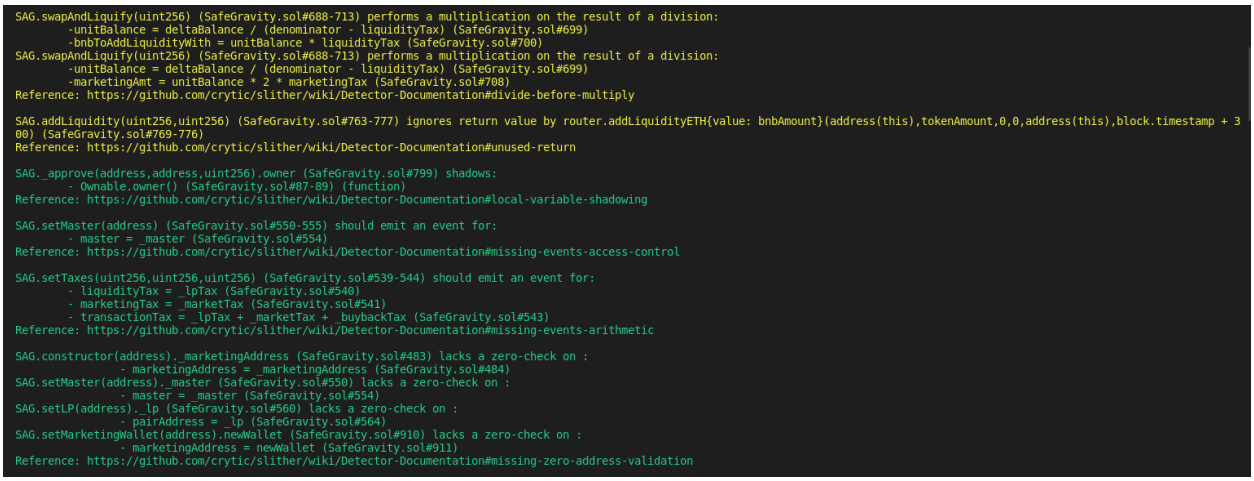
Lock the pragma version

Automated Testing

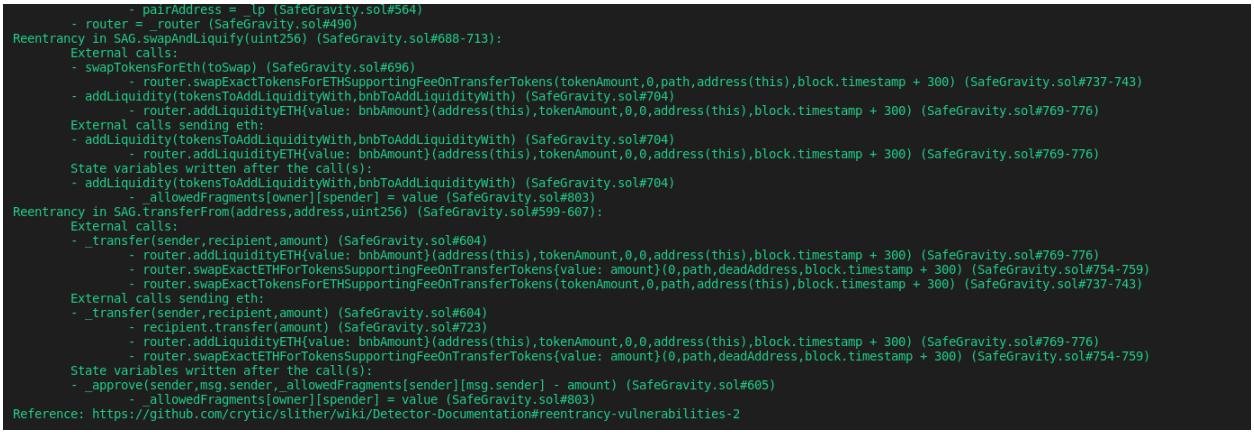
We have to use Automated testing Slither. It is an Automated Analysis Tool in Smart Contract.



```
eentrancy in SAG._transfer(address,address,uint256) (SafeGravity.sol#615-642):
       External calls:
       - swapAndLiquify(numTokensSell) (SafeGravity.sol#632)
               - router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,address(this),block.timestamp + 300) (SafeGravity.sol#769-776)
               - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp + 300) (SafeGravity.sol#737-743)
       - buyBackTokens(buybackLimit / (buybackDivisor)) (SafeGravity.sol#637)
               - router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: amount}(0,path,deadAddress,block.timestamp + 300) (SafeGravity.sol#754-759)
       External calls sending eth:
       - swapAndLiquify(numTokensSell) (SafeGravity.sol#632)
               - recipient.transfer(amount) (SafeGravity.sol#723)
               - router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,address(this),block.timestamp + 300) (SafeGravity.sol#769-776)
       - buyBackTokens(buybackLimit / (buybackDivisor)) (SafeGravity.sol#637)
               - router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: amount}(0,path,deadAddress,block.timestamp + 300) (SafeGravity.sol#754-759)
       State variables written after the call(s):
       - _tokenTransfer(from,to,value) (SafeGravity.sol#641)
               - _gonBalances[address(this)] = _gonBalances[address(this)] + rFee (SafeGravity.sol#684)
               - _gonBalances[sender] = _gonBalances[sender] - gonValue (SafeGravity.sol#665)
               - _gonBalances[recipient] = _gonBalances[recipient] + gonValue (SafeGravity.sol#666)
               - _gonBalances[sender] = _gonBalances[sender] - gonDeduct (SafeGravity.sol#657)
               - _gonBalances[recipient] = _gonBalances[recipient] + gonValue (SafeGravity.sol#658)
       - buyBackTokens(buybackLimit / (buybackDivisor)) (SafeGravity.sol#637)
               - inSwapAndLiquify = true (SafeGravity.sol#462)
               - inSwapAndLiquify = false (SafeGravity.sol#464)
eference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities

AG._totalSupply (SafeGravity.sol#475) shadows:
       - ERC20._totalSupply (SafeGravity.sol#124)
eference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing
```



```
SAG.swapAndLiquify(uint256) (SafeGravity.sol#688-713) performs a multiplication on the result of a division:
        -unitBalance = deltaBalance / (denominator - liquidityTax) (SafeGravity.sol#699)
        -bnbToAddLiquidityWith = unitBalance * liquidityTax (SafeGravity.sol#700)
SAG.swapAndLiquify(uint256) (SafeGravity.sol#688-713) performs a multiplication on the result of a division:
        -unitBalance = deltaBalance / (denominator - liquidityTax) (SafeGravity.sol#699)
        -marketingAmt = unitBalance * 2 * marketingTax (SafeGravity.sol#708)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

SAG.addLiquidity(uint256,uint256) (SafeGravity.sol#763-777) ignores return value by router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,address(this),block.timestamp + 3
00) (SafeGravity.sol#769-776)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

SAG._approve(address,address,uint256).owner (SafeGravity.sol#799) shadows:
        - Ownable.owner() (SafeGravity.sol#87-89) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

SAG.setMaster(address) (SafeGravity.sol#550-555) should emit an event for:
        - master = _master (SafeGravity.sol#554)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control

SAG.setTaxes(uint256,uint256,uint256) (SafeGravity.sol#539-544) should emit an event for:
        - liquidityTax = _lpTax (SafeGravity.sol#540)
        - marketingTax = _marketTax (SafeGravity.sol#541)
        - transactionTax = _lpTax + _marketTax + _buybackTax (SafeGravity.sol#543)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

SAG.constructor(address)._marketingAddress (SafeGravity.sol#483) lacks a zero-check on :
                - marketingAddress = _marketingAddress (SafeGravity.sol#484)
SAG.setMaster(address)._master (SafeGravity.sol#550) lacks a zero-check on :
                - master = _master (SafeGravity.sol#554)
SAG.setLP(address)._lp (SafeGravity.sol#560) lacks a zero-check on :
                - pairAddress = _lp (SafeGravity.sol#564)
SAG.setMarketingWallet(address).newWallet (SafeGravity.sol#910) lacks a zero-check on :
                - marketingAddress = newWallet (SafeGravity.sol#911)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```



```
                - pairAddress = _lp (SafeGravity.sol#564)
        - router = _router (SafeGravity.sol#490)
Reentrancy in SAG.swapAndLiquify(uint256) (SafeGravity.sol#688-713):
        External calls:
        - swapTokensForEth(toSwap) (SafeGravity.sol#696)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp + 300) (SafeGravity.sol#737-743)
        - addLiquidity(tokensToAddLiquidityWith,bnbToAddLiquidityWith) (SafeGravity.sol#704)
                - router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,address(this),block.timestamp + 300) (SafeGravity.sol#769-776)
        External calls sending eth:
        - addLiquidity(tokensToAddLiquidityWith,bnbToAddLiquidityWith) (SafeGravity.sol#704)
                - router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,address(this),block.timestamp + 300) (SafeGravity.sol#769-776)
        State variables written after the call(s):
        - addLiquidity(tokensToAddLiquidityWith,bnbToAddLiquidityWith) (SafeGravity.sol#704)
                - _allowedFragments[owner][spender] = value (SafeGravity.sol#803)
Reentrancy in SAG.transferFrom(address,address,uint256) (SafeGravity.sol#599-607):
        External calls:
        - _transfer(sender,recipient,amount) (SafeGravity.sol#604)
                - router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,address(this),block.timestamp + 300) (SafeGravity.sol#769-776)
                - router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: amount}(0,path,deadAddress,block.timestamp + 300) (SafeGravity.sol#754-759)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp + 300) (SafeGravity.sol#737-743)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (SafeGravity.sol#604)
                - recipient.transfer(amount) (SafeGravity.sol#723)
                - router.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,address(this),block.timestamp + 300) (SafeGravity.sol#769-776)
                - router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: amount}(0,path,deadAddress,block.timestamp + 300) (SafeGravity.sol#754-759)
        State variables written after the call(s):
        - _approve(sender,msg.sender,_allowedFragments[sender][msg.sender] - amount) (SafeGravity.sol#605)
                - _allowedFragments[owner][spender] = value (SafeGravity.sol#803)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

```
Context._msgData() (SafeGravity.sol#71-74) is never used and should be removed
ERC20._approve(address,address,uint256) (SafeGravity.sol#367-373) is never used and should be removed
ERC20._beforeTokenTransfer(address,address,uint256) (SafeGravity.sol#389) is never used and should be removed
ERC20._burn(address,uint256) (SafeGravity.sol#341-352) is never used and should be removed
ERC20._mint(address,uint256) (SafeGravity.sol#320-328) is never used and should be removed
ERC20._transfer(address,address,uint256) (SafeGravity.sol#297-309) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.6 (SafeGravity.sol#7) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.6 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Function IRouter.WETH() (SafeGravity.sol#41) is not in mixedCase
Parameter SAG.setTaxes(uint256,uint256,uint256)._lpTax (SafeGravity.sol#539) is not in mixedCase
Parameter SAG.setTaxes(uint256,uint256,uint256)._marketTax (SafeGravity.sol#539) is not in mixedCase
Parameter SAG.setTaxes(uint256,uint256,uint256)._buybackTax (SafeGravity.sol#539) is not in mixedCase
Parameter SAG.setMaster(address)._master (SafeGravity.sol#550) is not in mixedCase
Parameter SAG.setLP(address)._lp (SafeGravity.sol#560) is not in mixedCase
Parameter SAG.setSwapAndLiquifyEnabled(bool)._enabled (SafeGravity.sol#577) is not in mixedCase
Parameter SAG.calculateFee(uint256)._amount (SafeGravity.sol#678) is not in mixedCase
Parameter SAG.enableTransfer(address)._addr (SafeGravity.sol#874) is not in mixedCase
Parameter SAG.excludeAddress(address)._addr (SafeGravity.sol#881) is not in mixedCase
Parameter SAG.setBuyBackEnabled(bool)._enabled (SafeGravity.sol#888) is not in mixedCase
Parameter SAG.setBuyBackLimit(uint256)._buybackLimit (SafeGravity.sol#892) is not in mixedCase
Parameter SAG.setBuyBackDivisor(uint256)._buybackDivisor (SafeGravity.sol#895) is not in mixedCase
Parameter SAG.setnumTokensSellDivisor(uint256)._numTokensSellDivisor (SafeGravity.sol#898) is not in mixedCase
Parameter SAG.setMaxTxDivisor(uint256)._maxTxDivisor (SafeGravity.sol#902) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (SafeGravity.sol#72)" inContext (SafeGravity.sol#66-75)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
```

```
SAG.slitherConstructorVariables() (SafeGravity.sol#392-915) uses literals with too many digits:
        - deadAddress = 0x000000000000000000000000000000000000dEaD (SafeGravity.sol#450)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

ERC20._allowances (SafeGravity.sol#122) is never used in SAG (SafeGravity.sol#392-915)
SAG.privateSaleDropCompleted (SafeGravity.sol#459) is never used in SAG (SafeGravity.sol#392-915)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

SAG.deadAddress (SafeGravity.sol#450) should be constant
SAG.privateSaleDropCompleted (SafeGravity.sol#459) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
```

```
        - SAG.transfer(address,uint256) (SafeGravity.sol#591-595)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (SafeGravity.sol#205-207)
        - SAG.allowance(address,address) (SafeGravity.sol#836-843)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (SafeGravity.sol#216-219)
        - SAG.approve(address,uint256) (SafeGravity.sol#819-827)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (SafeGravity.sol#234-242)
        - SAG.transferFrom(address,address,uint256) (SafeGravity.sol#599-607)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (SafeGravity.sol#256-259)
        - SAG.increaseAllowance(address,uint256) (SafeGravity.sol#788-796)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (SafeGravity.sol#275-281)
        - SAG.decreaseAllowance(address,uint256) (SafeGravity.sol#851-865)
setSwapAndLiquifyEnabled(bool) should be declared external:
        - SAG.setSwapAndLiquifyEnabled(bool) (SafeGravity.sol#577-580)
setBuyBackEnabled(bool) should be declared external:
        - SAG.setBuyBackEnabled(bool) (SafeGravity.sol#888-890)
setBuyBackLimit(uint256) should be declared external:
        - SAG.setBuyBackLimit(uint256) (SafeGravity.sol#892-893)
setBuyBackDivisor(uint256) should be declared external:
        - SAG.setBuyBackDivisor(uint256) (SafeGravity.sol#895-896)
setnumTokensSellDivisor(uint256) should be declared external:
        - SAG.setnumTokensSellDivisor(uint256) (SafeGravity.sol#898-900)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
./SafeGravity.sol analyzed (9 contracts with 75 detectors), 67 result(s) found
```

Sol-hint Tool:

A linter for Solidity that provides both Security and Style Guide validations.

Coding style issues influence code readability and, in some cases, may lead to bugs in future. Smart Contracts have a naming convention, indentation and code layout issues. It's recommended to use Solidity Style Guide to fix all the issues. Consider following the Solidity guidelines on formatting the code and commenting for all the files. It can improve the overall code quality and readability

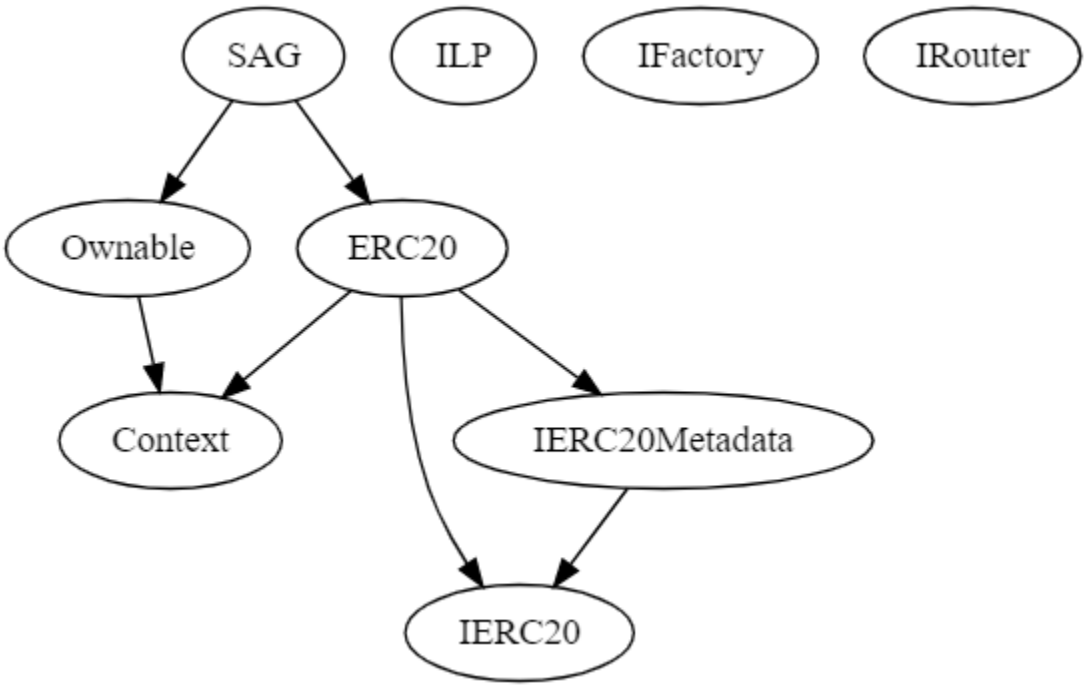
```
safegravity.sol
   72:2  error  Line length must be no more than 120 but current length is 132  max-line-length
  591:2  error  Line length must be no more than 120 but current length is 137  max-line-length
  615:2  error  Line length must be no more than 120 but current length is 126  max-line-length

✖3 problems (3 errors, 0 warnings)
```

Functional View

Function	Return Type	Test Result
name	Public	Verified
symbol	Public	Verified
decimals	Public	Verified
totalSupply	Public	Verified
balanceOf	Public	Verified
transfer	Public	Verified
allowance	Public	Verified
approve	Public	Verified
TransferFrom	Public	Verified
increaseAllowance	Public	Verified
decreaseAllowance	Public	Verified
isExcludedFromReward	Public	Verified
totalFees	Public	Verified
deliver	Public	Verified
reflectionFromToken	Public	Verified

tokenFromReflection	Public	Verified
excludeFromReward	Public	Verified
includeInReward	Public	Verified
transferBothExcluded	Public	Verified

Inheritance Chart



Audit Findings Results

There were 5 Informational found during the audit. All the mentioned findings may have an effect only in case of specific conditions performed by the contract owner. None of the critical issues were resolved.

Generally, the contracts are well written and structured. The findings during the audit have some impact on contract performance or security

Disclaimer

This audit does not provide a security or correctness guarantee of audited smart contracts You agree that your access and/or use, including but not limited to any services, products, platforms, content, will be at your Own risk. Smart contract remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security.


http://tokenaudit.net/


https://t.me/TokenAuditt


https://twitter.com/AuditToken