Automated findings report: 0x4473996394e1Da0c6E7a79dc32008432 on bsc (mainnet)

Report generated by auditbase.com

Summary

Medium Risk Issues

	Issue	Instances
[M001]	The owner is a single point of failure and a centralization risk	5
[M002]	Return values of `transfer()`/`transferFrom()` not checked	2

Total: 7 instances over 2 issues

Low Risk Issues

	Issue	Instances
[L001]	Division by zero not prevented	2
[L002]	Use `Ownable2Step` rather than `Ownable`	1
[L003]	Setters should have initial value check	4
[L004]	Empty `receive()`/`payable fallback()` function does not authorize requests	1
[L005]	Contracts are designed to receive ETH but do not implement function for withdrawal	1
[L006]	No limits when setting state variable amounts	9
[L007]	Allowed fees/rates should be capped by smart contracts	3

	Issue	Instances
[L008]	Consider implementing two-step procedure for updating protocol addresses	1
[L009]	Governance functions should be controlled by time locks	5
[L010]	Missing checks for address(0x0) when updating address state variables	2
[L011]	The setter does not set a state variable or call a fuction	1

Total: 30 instances over 11 issues

Non-critical Issues

	Issue	Instances
[NC001]	Constants should be defined rather than using magic numbers	11
[NC002]	Use scientific notation (e.g. 1e18) rather than exponentiation (e.g. 10**18)	2
[NC003]	Function ordering does not follow the Solidity style guide	1
[NC004]	Constants in comparisons should appear on the left side	1
[NC005]	Public functions not called by the contract should be declared external instead	11

	Issue	Instances
[NC006]	Multiple `address`/ID mappings can be combined into a single `mapping` of an `address`/ID to a `struct`, for readability	1
[NC007]	Interfaces should be indicated with an I prefix in the contract name	1
[NC008]	Variables need not be initialized to zero	3
[NC009]	Variable names for `constant`s are improperly named	4
[NC010]	Variable names for `immutable`s should use CONSTANT_CASE	1
[NC011]	Lines are too long	5
[NC012]	All @param values in a function's NatSpec comment should match the function's parameters	47
[NC013]	NatSpec @return argument is missing	34
[NC014]	File is missing NatSpec comments	1
[NC015]	Contract declarations should have NatSpec descriptions	8
[NC016]	Function declarations should have NatSpec descriptions	40
[NC017]	Contract declarations should have `@notice` tags	8
[NC018]	Contract declarations should have NatSpec @title annotations	8
[NC019]	State variable declarations should have NatSpec descriptions	29

	Issue	Instances
[NC020]	Event declarations should have NatSpec descriptions	2
[NC021]	Contract declarations should have NatSpec `@author` annotations	1
[NC022]	Function declarations should have `@notice` tags	1
[NC023]	Contract does not follow the Solidity style guide's suggested layout ordering	2
[NC024]	Non-external/public variable names should begin with an underscore	2
[NC025]	Consider disabling `renounceOwnership()`	1
[NC026]	Non-library/interface files should use fixed compiler versions, not floating ones	1
[NC027]	Contracts should have full test coverage	1
[NC028]	Large or complicated code bases should implement invariant tests	1
[NC029]	`address` shouldn't be hard-coded	5
[NC030]	Consider using `block.number` instead of `block.timestamp`	1
[NC031]	Large numeric literals should use underscores for readability	1
[NC032]	Variables should be named in mixedCase style	3
[NC033]	Function names should use lowerCamelCase	6
[NC034]	Consider adding a deny-list	2

	Issue	Instances
[NC035]	Custom errors should be used rather than `revert()`/`require()`	8
[NC036]	Top level declarations should be separated by two blank lines	7
[NC037]	Interfaces should be defined in separate files from their usage	4
[NC038]	Enum values should be used in place of constant array indexes	2
[NC039]	Zero as a function argument should have a descriptive meaning	13
[NC040]	Function names should differ to make the code more readable	18
[NC041]	It is standard for all external and public functions to be override from an interface	16
[NC042]	Consider adding formal verification proofs	1
[NC043]	Public state variables shouldn't have a preceding _ in their name	1
[NC044]	Large multiples of ten should use scientific notation (e.g. 1e6) rather than decimal literals (e.g. 1000000), for readability	1
[NC045]	Polymorphic functions make security audits more time-consuming and error-prone	9
[NC046]	Missing event and or timelock for critical parameter change	5
[NC047]	Setters should prevent re-setting of the same value	4

	Issue	Instances
[NC048]	High cyclomatic complexity	1
[NC049]	Use of override is unnecessary	6
[NC050]	Non-`external`/`public` function names should begin with an underscore	3
[NC051]	Contracts/libraries should each be defined in separate files	1
[NC052]	Consider adding emergency-stop functionality	2
[NC053]	Named imports of parent contracts are missing	2
[NC054]	NatSpec: Contract declarations should have @dev tags	8
[NC055]	NatSpec: Function @param tag is missing	37
[NC056]	Pure function accesses storage	12

Total: 407 instances over 56 issues

Gas Optimizations

	Issue	Instances	Total Gas Saved
[G001]	Don't Initialize Variables with Default Value	3	-
[G002]	Use != 0 instead of > 0 for Unsigned Integer Comparison	4	-
[G003]	Using `bool` for storage incurs overhead	2	34200
[G004]	Long Revert Strings	2	-

	Issue	Instances	Total Gas Saved
[G005]	Functions guaranteed to revert when called by normal users can be marked `payable`	5	-
[G006]	Use Custom Errors	8	232
[G007]	Use assembly to check for `address(0)`	4	24
[G008]	State variables should be cached in stack variables rather than re-reading them from storage	11	1067
[G009]	Multiple address/ID mappings can be combined into a single mapping of an address/ID to a struct, where appropriate	1	20042
[G010]	Multiple accesses of a mapping/array should use a local variable cache	1	42
[G011]	Internal functions only called once can be inlined to save gas	2	40
[G012]	Add unchecked {} for subtractions where the operands cannot underflow because of a previous require() or if-statement	1	85
[G013]	Optimize names to save gas	1	22

	Issue	Instances	Total Gas Saved
[G014]	Constructors can be marked payable	2	42
[G015]	Remove unused variables	5	-
[G016]	Use solidity version 0.8.20 or above to improve gas performance	1	1000
[G017]	Use assembly to emit events	3	-
[G018]	Don't use `_msgSender()` if not supporting EIP-2771	9	-
[G019]	Use `uint256(1)`/`uint256(2)` instead for `true` and `false` boolean states	4	68400
[G020]	Usage of `uints`/`ints` smaller than 32 bytes (256 bits) incurs overhead	1	10
[G021]	Consider activating `via-ir` for deploying	1	-
[G022]	`unchecked {}` can be used on the division of two `uints` in order to save gas	2	-
[G023]	Private functions used once can be inlined	11	-
[G024]	Unused named return variables without optimizer waste gas	27	-

	Issue	Instances	Total Gas Saved
[G025]	Avoid updating storage when the value hasn't changed	6	17400
[G026]	Do not calculate constants	1	-
[G027]	The use of a logical AND in place of double if is slightly less gas efficient in instances where there isn't a corresponding else statement for the given if statement	6	-

Total: 124 instances over 27 issues with **142606 gas** saved.

Gas totals use lower bounds of ranges and count two iterations of each for-loop. All values above are runtime, not deployment, values; deployment values are listed in the individual issue descriptions. The table above as well as its gas numbers do not include any of the excluded findings.

M001 - The owner is a single point of failure and a centralization risk:

Having a single EOA as the only owner of contracts is a large centralization risk and a single point of failure. A single private key may be taken in a hack, or the sole holder of the key may become unable to retrieve the key when necessary. Consider changing to a multi-signature setup, or having a role-based authorization model.

Click to show 5 findings

`File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol`

```
84          function renounceOwnership() public virtual onlyOwner {
```

```
208         function setIsExcludedFromFee(address account, bool newValue) public onlyOwner {
```

```
212         function setMaxWalletPer(uint256 newMaxWalletPer) public onlyOwner {
```

```
216         function setBuyFees(uint256 newBuyFee, uint256 newRedisBuyFee) public onlyOwner
```

```
221         function setSellFees(uint256 newSellFee, uint256 newRedisSellFee) public onlyOwn
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L221:221

M002 - Return values of `transfer()/transferFrom()` not checked:

Not all IERC20 implementations revert() when there's a failure in transfer()/transferFrom(). The function signature has a boolean return value and they indicate errors that way instead. By not checking the return value, operations that should have marked as failed, may potentially go through without actually making a payment

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
311             _twddev.transfer(dAmount);
```

```
312             _twdmkt.transfer(mAmount);
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L312:312

L001 - Division by zero not prevented:

The divisions below take an input parameter which does not have any zero-value checks, which may lead to the functions reverting when zero is passed.

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
50              require(c / a == b, "Multiplication overflow");
```

```
60              uint256 c = a / b;
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L60:60

L002 - Use `Ownable2Step` rather than `Ownable`:

`Ownable2Step` and `Ownable2StepUpgradeable` prevent the contract ownership from mistakenly being transferred to an address that cannot handle it (e.g. due to a typo in the address), by requiring that the recipient of the owner permissions actively accept via a contract call of its own.

`File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol`

```
113     contract SAVER is Context, IERC20, Ownable {
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L113:113

L003 - Setters should have initial value check:

Setters should have initial value check to prevent assigning wrong value to the variable. Assginment of wrong value can lead to unexpected behavior of the contract.

Click to show 4 findings

`File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol`

```
208         function setIsExcludedFromFee(address account, bool newValue) public onlyOwner {
209             _isExcludedFromFee[account] = newValue;
210         }


212         function setMaxWalletPer(uint256 newMaxWalletPer) public onlyOwner {
213             _maxWalletPer = newMaxWalletPer;
214         }


216         function setBuyFees(uint256 newBuyFee, uint256 newRedisBuyFee) public onlyOwner
217             _taxFeeOnBuy = newBuyFee;
218             _redisFeeOnBuy = newRedisBuyFee;
219         }


221         function setSellFees(uint256 newSellFee, uint256 newRedisSellFee) public onlyOwn
222             _taxFeeOnSell = newSellFee;
```

```
223             _redisFeeOnSell = newRedisSellFee;
224         }
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L221:224

L004 - Empty `receive()/payable fallback()` function does not authorize requests:

If the intention is for the Ether to be used, the function should call another function, otherwise it should revert (e.g. `require(msg.sender == address(weth))`). Having no access control on the function means that someone may send Ether to the contract, and have no way to get anything back out, which is a loss of funds. If the concern is having to spend a small amount of gas to check the sender against an immutable address, the code should at least have a function to rescue unused Ether.

`File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol`

```
339         receive() external payable {}
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L339:339

L005 - Contracts are designed to receive ETH but do not implement function for withdrawal:

The following contracts can receive ETH but can not withdraw, ETH is occasionally sent by users will be stuck in those contracts. This functionality also applies to baseTokens resulting in locked tokens and loss of funds.

`File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol`

```
339         receive() external payable {}
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L339:339

L006 - No limits when setting state variable amounts:

It is important to ensure state variables numbers are set to a reasonable value.

Click to show 9 findings

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
213             _maxWalletPer = newMaxWalletPer;
```

```
217             _taxFeeOnBuy = newBuyFee;
```

```
218             _redisFeeOnBuy = newRedisBuyFee;
```

```
222             _taxFeeOnSell = newSellFee;
```

```
223             _redisFeeOnSell = newRedisSellFee;
```

```
275                     _redisFee = _redisFeeOnBuy;
```

```
276                     _taxFee = _taxFeeOnBuy;
```

```
280                     _redisFee = _redisFeeOnSell;
```

```
281                     _taxFee = _taxFeeOnSell;
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L281:281

L007 - Allowed fees/rates should be capped by smart contracts:

Fees/rates should be required to be below 100%, preferably at a much lower limit, to ensure users don't have to monitor the blockchain for changes prior to using the protocol.

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
208         function setIsExcludedFromFee(address account, bool newValue) public onlyOwner {
209             _isExcludedFromFee[account] = newValue;
210         }
```

```
216         function setBuyFees(uint256 newBuyFee, uint256 newRedisBuyFee) public onlyOwner
217             _taxFeeOnBuy = newBuyFee;
218             _redisFeeOnBuy = newRedisBuyFee;
219         }
```

```
221         function setSellFees(uint256 newSellFee, uint256 newRedisSellFee) public onlyOwn
222             _taxFeeOnSell = newSellFee;
223             _redisFeeOnSell = newRedisSellFee;
224         }
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L221:224

L008 - Consider implementing two-step procedure for updating protocol addresses:

Lack of two-step procedure for critical operations leaves them error-prone. Consider adding two step procedure on the critical functions. See similar findings in previous Code4rena contests for reference: https://code4rena.com/reports/2022-06-illuminate/#2-critical-changes-should-use-two-step-procedure

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
208         function setIsExcludedFromFee(address account, bool newValue) public onlyOwner
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L208:208

L009 - Governance functions should be controlled by time locks:

Governance functions (such as upgrading contracts, setting critical parameters) should be controlled using time locks to introduce a delay between a proposal and its execution. This gives users time to exit before a potentially dangerous or malicious operation is applied.

Click to show 5 findings

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
216         function setBuyFees(uint256 newBuyFee, uint256 newRedisBuyFee) public onlyOwner
```

```
208         function setIsExcludedFromFee(address account, bool newValue) public onlyOwner {

84          function renounceOwnership() public virtual onlyOwner {

212         function setMaxWalletPer(uint256 newMaxWalletPer) public onlyOwner {

221         function setSellFees(uint256 newSellFee, uint256 newRedisSellFee) public onlyOwn
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L221:224

L010 - Missing checks for address(0x0) when updating address state variables:

issing checks for address(0x0) when updating address state variables

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
72              _owner = msgSender;


85              _owner = address(0);
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L85:85

L011 - The setter does not set a state variable or call a fuction:

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
208         function setIsExcludedFromFee(address account, bool newValue) public onlyOwner {
209             _isExcludedFromFee[account] = newValue;
210         }
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L208:210

NC001 - Constants should be defined rather than using magic numbers:

Even assembly can benefit from using readable constants instead of hex/numeric literals.

Click to show 11 findings

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
128         uint256 private _taxFeeOnBuy = 6;

131         uint256 private _taxFeeOnSell = 6;

139         address public immutable deadAddress = 0x000000000000000000000000000000000000dEa

140         address payable private _twddev = payable(0x55F84c660e27F630AF3112075e0D94c50F75

141         address payable private _twdmkt = payable(0x55F84c660e27F630AF3112075e0D94c50F75

161             IUniswapV2Router02 _uniswapV2Router = IUniswapV2Router02(0x10ED43C718714eb63

171             emit Transfer(address(0x0000000000000000000000000000000000000000), _msgSende

256                 uint256 maxWalletTokens = _maxWalletPer.mul(_tTotal).div(100);

309             uint256 dAmount = amount.mul(_devPer).div(100);

349             uint256 tFee = tAmount.mul(taxFee).div(100);

350             uint256 tTeam = tAmount.mul(teamFee).div(100);
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L350:350

NC002 - Use scientific notation (e.g. 1e18) rather than exponentiation (e.g. 10**18):

While the compiler knows to optimize away the exponentiation, it's still better coding practice to use idioms that do not require compiler optimization, if they exist.

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
123         uint256 private constant _tTotal = 1000000000 * 10**9;
```

```
123         uint256 private constant _tTotal = 1000000000 * 10**9;
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L123:123

NC003 - Function ordering does not follow the Solidity style guide:

According to the Solidity style guide, functions should be laid out in the following order :`constructor()`, `receive()`, `fallback()`, `external`, `public`, `internal`, `private`, but the cases below do not follow this pattern.

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
339         receive() external payable {}
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L339:339

NC004 - Constants in comparisons should appear on the left side:

This issue arises when constants in comparisons appear on the right side, which can lead to typo bugs.

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
46              if (a == 0) {
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L46:46

NC005 - Public functions not called by the contract should be declared external instead:

Contracts are allowed to override their parents' functions and change the visibility from `external` to `public`.

Click to show 11 findings

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
174         function name() public pure returns (string memory) {
175             return _name;
176         }


178         function symbol() public pure returns (string memory) {
179             return _symbol;
180         }


182         function decimals() public pure returns (uint8) {
183             return _decimals;
184         }


186         function totalSupply() public pure override returns (uint256) {
187             return _tTotal;
188         }


199         function allowance(address owner, address spender) public view override returns
200             return _allowances[owner][spender];
201         }


203         function approve(address spender, uint256 amount) public override returns (bool)
204             _approve(_msgSender(), spender, amount);
205             return true;
206         }


208         function setIsExcludedFromFee(address account, bool newValue) public onlyOwner {
209             _isExcludedFromFee[account] = newValue;
210         }


212         function setMaxWalletPer(uint256 newMaxWalletPer) public onlyOwner {
213             _maxWalletPer = newMaxWalletPer;
214         }
```

```
216         function setBuyFees(uint256 newBuyFee, uint256 newRedisBuyFee) public onlyOwner
217             _taxFeeOnBuy = newBuyFee;
218             _redisFeeOnBuy = newRedisBuyFee;
219         }


221         function setSellFees(uint256 newSellFee, uint256 newRedisSellFee) public onlyOwn
222             _taxFeeOnSell = newSellFee;
223             _redisFeeOnSell = newRedisSellFee;
224         }


226         function transferFrom(address sender, address recipient, uint256 amount) public
227             _transfer(sender, recipient, amount);
228             _approve(sender, _msgSender(), _allowances[sender][_msgSender()].sub(amount,
229             return true;
230         }
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L226:230

NC006 - Multiple `address`/ID mappings can be combined into a single `mapping` of an `address`/ID to a `struct`, for readability:

Well-organized data structures make code reviews easier, which may lead to fewer bugs. Consider combining related mappings into mappings to structs, so it's clear what data is related

`File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol`

```
116         mapping (address => uint256) private _rOwned;
117         mapping (address => uint256) private _tOwned;
118         mapping (address => mapping (address => uint256)) private _allowances;
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L116:118

NC007 - Interfaces should be indicated with an I prefix in the contract name:

`File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol`

```
23      interface Token {
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L23:23

NC008 - Variables need not be initialized to zero:

The default value for variables is zero, so initializing them to zero is superfluous.

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
127         uint256 private _redisFeeOnBuy = 0;
```

```
130         uint256 private _redisFeeOnSell = 0;
```

```
133         uint256 private _devPer = 0;
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L133:133

NC009 - Variable names for `constants` are improperly named:

According to the Style guide, for `constant` variable names, each word should use all capital letters, with underscores separating each word (CONSTANT_CASE).

Click to show 4 findings

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
123         uint256 private constant _tTotal = 1000000000 * 10**9;
```

```
143         string private constant _name = "Saver Protocol";
```

```
144         string private constant _symbol = "SVR";
```

```
145         uint8 private constant _decimals = 9;
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L145:145

NC010 - Variable names for `immutables` should use CONSTANT_CASE:

For `immutable` variable names, each word should use all capital letters, with underscores separating each word (CONSTANT_CASE).

`File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol`

```
139         address public immutable deadAddress = 0x000000000000000000000000000000000000dEa
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L139:139

NC011 - Lines are too long:

Usually lines in source code are limited to 80 characters. Today's screens are much larger so it's reasonable to stretch this in some cases. The solidity style guide recommends a maximumum line length of 120 characters, so the lines below should be split when they reach that length.

Click to show 5 findings

`File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol`

```
_approve(sender, _msgSender(), _allowances[sender][_msgSender()].sub(amount, "Transfer amoun

if ((_isExcludedFromFee[from] || _isExcludedFromFee[to]) || (from != uniswapV2Pair && to !=

(uint256 rAmount, uint256 rTransferAmount, uint256 rFee, uint256 tTransferAmount, uint256 tF

function _getTValues(uint256 tAmount, uint256 taxFee, uint256 teamFee) private pure returns

function _getRValues(uint256 tAmount, uint256 tFee, uint256 tTeam, uint256 currentRate) priv
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L355:355

NC012 - All @param values in a function's NatSpec comment should match the function's parameters:

A function's NatSpec comment should accurately describe all of the function's parameters.

Click to show 47 findings

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
7           function _msgSender() internal view virtual returns (address) {

13          function totalSupply() external view returns (uint256);

14          function balanceOf(address account) external view returns (uint256);

15          function transfer(address recipient, uint256 amount) external returns (bool);

16          function allowance(address owner, address spender) external view returns (uint25

17          function approve(address spender, uint256 amount) external returns (bool);

18          function transferFrom(address sender, address recipient, uint256 amount) externa

24          function transferFrom(address, address, uint) external returns (bool);

25          function transfer(address, uint) external returns (bool);

29          function add(uint256 a, uint256 b) internal pure returns (uint256) {

35          function sub(uint256 a, uint256 b) internal pure returns (uint256) {

39          function sub(uint256 a, uint256 b, string memory errorMessage) internal pure ret

45          function mul(uint256 a, uint256 b) internal pure returns (uint256) {

54          function div(uint256 a, uint256 b) internal pure returns (uint256) {
```

```
58          function div(uint256 a, uint256 b, string memory errorMessage) internal pure ret

75          function owner() public view returns (address) {

84          function renounceOwnership() public virtual onlyOwner {

90          function createPair(address tokenA, address tokenB) external returns (address pa

101         function factory() external pure returns (address);

102         function WETH() external pure returns (address);

174         function name() public pure returns (string memory) {

178         function symbol() public pure returns (string memory) {

182         function decimals() public pure returns (uint8) {

186         function totalSupply() public pure override returns (uint256) {

190         function balanceOf(address account) public view override returns (uint256) {

194         function transfer(address recipient, uint256 amount) public override returns (bo

199         function allowance(address owner, address spender) public view override returns

203         function approve(address spender, uint256 amount) public override returns (bool)

208         function setIsExcludedFromFee(address account, bool newValue) public onlyOwner {
```

```
212         function setMaxWalletPer(uint256 newMaxWalletPer) public onlyOwner {

216         function setBuyFees(uint256 newBuyFee, uint256 newRedisBuyFee) public onlyOwner

221         function setSellFees(uint256 newSellFee, uint256 newRedisSellFee) public onlyOwn

226         function transferFrom(address sender, address recipient, uint256 amount) public

232         function tokenFromReflection(uint256 rAmount) private view returns(uint256) {

238         function _approve(address owner, address spender, uint256 amount) private {

245         function _transfer(address from, address to, uint256 amount) private {

294         function swapTokensForEth(uint256 tokenAmount) private lockTheSwap {

308         function sendETHToFee(uint256 amount) private {

315         function _tokenTransfer(address sender, address recipient, uint256 amount) priva

319         function _transferStandard(address sender, address recipient, uint256 tAmount) p

328         function _takeTeam(uint256 tTeam) private {

334         function _reflectFee(uint256 rFee, uint256 tFee) private {

341         function _getValues(uint256 tAmount) private view returns (uint256, uint256, ui

348         function _getTValues(uint256 tAmount, uint256 taxFee, uint256 teamFee) private p

355         function _getRValues(uint256 tAmount, uint256 tFee, uint256 tTeam, uint256 curre
```

```
363         function _getRate() private view returns(uint256) {
```

```
368         function _getCurrentSupply() private view returns(uint256, uint256) {
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L368:368

NC013 - NatSpec @return argument is missing:

A function's NatSpec @return statement should accurately describe the function's return value.

Click to show 34 findings

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
7           function _msgSender() internal view virtual returns (address) {
```

```
13          function totalSupply() external view returns (uint256);
```

```
14          function balanceOf(address account) external view returns (uint256);
```

```
15          function transfer(address recipient, uint256 amount) external returns (bool);
```

```
16          function allowance(address owner, address spender) external view returns (uint25
```

```
17          function approve(address spender, uint256 amount) external returns (bool);
```

```
18          function transferFrom(address sender, address recipient, uint256 amount) externa
```

```
24          function transferFrom(address, address, uint) external returns (bool);
```

```
25          function transfer(address, uint) external returns (bool);
```

```
29          function add(uint256 a, uint256 b) internal pure returns (uint256) {

35          function sub(uint256 a, uint256 b) internal pure returns (uint256) {

39          function sub(uint256 a, uint256 b, string memory errorMessage) internal pure ret

45          function mul(uint256 a, uint256 b) internal pure returns (uint256) {

54          function div(uint256 a, uint256 b) internal pure returns (uint256) {

58          function div(uint256 a, uint256 b, string memory errorMessage) internal pure ret

75          function owner() public view returns (address) {

90          function createPair(address tokenA, address tokenB) external returns (address pa

101         function factory() external pure returns (address);

102         function WETH() external pure returns (address);

174         function name() public pure returns (string memory) {

178         function symbol() public pure returns (string memory) {

182         function decimals() public pure returns (uint8) {

186         function totalSupply() public pure override returns (uint256) {

190         function balanceOf(address account) public view override returns (uint256) {
```

```
194         function transfer(address recipient, uint256 amount) public override returns (bo

199         function allowance(address owner, address spender) public view override returns

203         function approve(address spender, uint256 amount) public override returns (bool)

226         function transferFrom(address sender, address recipient, uint256 amount) public

232         function tokenFromReflection(uint256 rAmount) private view returns(uint256) {

341         function _getValues(uint256 tAmount) private view returns (uint256, uint256, uin

348         function _getTValues(uint256 tAmount, uint256 taxFee, uint256 teamFee) private p

355         function _getRValues(uint256 tAmount, uint256 tFee, uint256 tTeam, uint256 curre

363         function _getRate() private view returns(uint256) {

368         function _getCurrentSupply() private view returns(uint256, uint256) {
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L368:368

NC014 - File is missing NatSpec comments:

The file does not contain any of the NatSpec comments (@inheritdoc, @param, @return, @notice), which are important for documentation and user confirmation.

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
// SPDX-License-Identifier: MIT

pragma solidity ^0.8.19;
```

```
abstract contract Context {
    function _msgSender() internal view virtual returns (address) {
        return msg.sender;
    }
}

interface IERC20 {
    function totalSupply() external view returns (uint256);
    function balanceOf(address account) external view returns (uint256);
    function transfer(address recipient, uint256 amount) external returns (bool);
    function allowance(address owner, address spender) external view returns (uint256);
    function approve(address spender, uint256 amount) external returns (bool);
    function transferFrom(address sender, address recipient, uint256 amount) external returr
    event Transfer(address indexed from, address indexed to, uint256 value);
    event Approval(address indexed owner, address indexed spender, uint256 value);
}

interface Token {
    function transferFrom(address, address, uint) external returns (bool);
    function transfer(address, uint) external returns (bool);
}

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

    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
        return sub(a, b, "Subtraction overflow");
    }

    function sub(uint256 a, uint256 b, string memory errorMessage) internal pure returns (ui
        require(b <= a, errorMessage);
        uint256 c = a - b;
        return c;
    }

    function mul(uint256 a, uint256 b) internal pure returns (uint256) {
        if (a == 0) {
            return 0;
        }
        uint256 c = a * b;
        require(c / a == b, "Multiplication overflow");
        return c;
```

```
    }

    function div(uint256 a, uint256 b) internal pure returns (uint256) {
        return div(a, b, "Division by zero");
    }

    function div(uint256 a, uint256 b, string memory errorMessage) internal pure returns (ui
        require(b > 0, errorMessage);
        uint256 c = a / b;
        return c;
    }

}

contract Ownable is Context {
    address private _owner;
    address private _previousOwner;

    constructor () {
        address msgSender = _msgSender();
        _owner = msgSender;
    }

    function owner() public view returns (address) {
        return _owner;
    }

    modifier onlyOwner() {
        require(_owner == _msgSender(), "Caller is not the owner");
        _;
    }

    function renounceOwnership() public virtual onlyOwner {
        _owner = address(0);
    }
}

interface IUniswapV2Factory {
    function createPair(address tokenA, address tokenB) external returns (address pair);
}

interface IUniswapV2Router02 {
    function swapExactTokensForETHSupportingFeeOnTransferTokens(
        uint amountIn,
        uint amountOutMin,
        address[] calldata path,
```

```
        address to,
        uint deadline
    ) external;
    function factory() external pure returns (address);
    function WETH() external pure returns (address);
    function addLiquidityETH(
        address token,
        uint amountTokenDesired,
        uint amountTokenMin,
        uint amountETHMin,
        address to,
        uint deadline
    ) external payable returns (uint amountToken, uint amountETH, uint liquidity);
}

contract SAVER is Context, IERC20, Ownable {

    using SafeMath for uint256;
    mapping (address => uint256) private _rOwned;
    mapping (address => uint256) private _tOwned;
    mapping (address => mapping (address => uint256)) private _allowances;
    mapping (address => bool) private _isExcludedFromFee;
    mapping (address => bool) private _rewardAddress;

    uint256 private constant MAX = ~uint256(0);
    uint256 private constant _tTotal = 1000000000 * 10**9;
    uint256 private _rTotal = (MAX - (MAX % _tTotal));
    uint256 private _tFeeTotal;

    uint256 private _redisFeeOnBuy = 0;
    uint256 private _taxFeeOnBuy = 6;

    uint256 private _redisFeeOnSell = 0;
    uint256 private _taxFeeOnSell = 6;

    uint256 private _devPer = 0;
    uint256 public _maxWalletPer = 2;

    uint256 private _redisFee;
    uint256 private _taxFee;

    address public immutable deadAddress = 0x000000000000000000000000000000000000dEaD;
    address payable private _twddev = payable(0x55F84c660e27F630AF3112075e0D94c50F75d1AB);
    address payable private _twdmkt = payable(0x55F84c660e27F630AF3112075e0D94c50F75d1AB);

    string private constant _name = "Saver Protocol";
```

```
string private constant _symbol = "SVR";
uint8 private constant _decimals = 9;

IUniswapV2Router02 public uniswapV2Router;
address public uniswapV2Pair;

bool private inSwap = false;
bool private swapEnabled = true;

modifier lockTheSwap {
    inSwap = true;
    _;
    inSwap = false;
}
constructor () {
    _rOwned[_msgSender()] = _rTotal;

    IUniswapV2Router02 _uniswapV2Router = IUniswapV2Router02(0x10ED43C718714eb63d5aA57B7
    uniswapV2Router = _uniswapV2Router;
    uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory())
        .createPair(address(this), _uniswapV2Router.WETH());

    _isExcludedFromFee[owner()] = true;
    _isExcludedFromFee[address(this)] = true;
    _isExcludedFromFee[_twddev] = true;
    _isExcludedFromFee[_twdmkt] = true;

    emit Transfer(address(0x0000000000000000000000000000000000000000), _msgSender(), _t
}

function name() public pure returns (string memory) {
    return _name;
}

function symbol() public pure returns (string memory) {
    return _symbol;
}

function decimals() public pure returns (uint8) {
    return _decimals;
}

function totalSupply() public pure override returns (uint256) {
    return _tTotal;
}
```

```
function balanceOf(address account) public view override returns (uint256) {
    return tokenFromReflection(_rOwned[account]);
}

function transfer(address recipient, uint256 amount) public override returns (bool) {
    _transfer(_msgSender(), recipient, amount);
    return true;
}

function allowance(address owner, address spender) public view override returns (uint256
    return _allowances[owner][spender];
}

function approve(address spender, uint256 amount) public override returns (bool) {
    _approve(_msgSender(), spender, amount);
    return true;
}

function setIsExcludedFromFee(address account, bool newValue) public onlyOwner {
    _isExcludedFromFee[account] = newValue;
}

function setMaxWalletPer(uint256 newMaxWalletPer) public onlyOwner {
    _maxWalletPer = newMaxWalletPer;
}

function setBuyFees(uint256 newBuyFee, uint256 newRedisBuyFee) public onlyOwner {
    _taxFeeOnBuy = newBuyFee;
    _redisFeeOnBuy = newRedisBuyFee;
}

function setSellFees(uint256 newSellFee, uint256 newRedisSellFee) public onlyOwner {
    _taxFeeOnSell = newSellFee;
    _redisFeeOnSell = newRedisSellFee;
}

function transferFrom(address sender, address recipient, uint256 amount) public override
    _transfer(sender, recipient, amount);
    _approve(sender, _msgSender(), _allowances[sender][_msgSender()].sub(amount, "Transf
    return true;
}

function tokenFromReflection(uint256 rAmount) private view returns(uint256) {
    require(rAmount <= _rTotal, "Amount must be less than total reflections");
    uint256 currentRate =  _getRate();
    return rAmount.div(currentRate);
```

```
}

function _approve(address owner, address spender, uint256 amount) private {
    require(owner != address(0), "Approve from the zero address");
    require(spender != address(0), "Approve to the zero address");
    _allowances[owner][spender] = amount;
    emit Approval(owner, spender, amount);
}

function _transfer(address from, address to, uint256 amount) private {
    require(from != address(0), "Transfer from the zero address");
    require(to != address(0), "Transfer to the zero address");
    require(amount > 0, "Transfer amount must be greater than zero");
    if (to != deadAddress &&
        to != address(this) &&
        to != uniswapV2Pair &&
        !_isExcludedFromFee[from] &&
        !_isExcludedFromFee[to]
    ) {
        uint256 heldTokens = balanceOf(to);
        uint256 maxWalletTokens = _maxWalletPer.mul(_tTotal).div(100);
        require((heldTokens + amount) <= maxWalletTokens, "Over wallet limit.");
    }

    _redisFee = 0;
    _taxFee = 0;

    if (from != owner() && to != owner()) {

        uint256 contractTokenBalance = balanceOf(address(this));
        if (!inSwap && from != uniswapV2Pair && swapEnabled && contractTokenBalance > 0)
            swapTokensForEth(contractTokenBalance);
            uint256 contractETHBalance = address(this).balance;
            if(contractETHBalance > 0) {
                sendETHToFee(address(this).balance);
            }
        }

        if(from == uniswapV2Pair && to != address(uniswapV2Router)) {
            _redisFee = _redisFeeOnBuy;
            _taxFee = _taxFeeOnBuy;
        }

        if (to == uniswapV2Pair && from != address(uniswapV2Router)) {
            _redisFee = _redisFeeOnSell;
            _taxFee = _taxFeeOnSell;
```

```
        }

        if ((_isExcludedFromFee[from] || _isExcludedFromFee[to]) || (from != uniswapV2Pa
            _redisFee = 0;
            _taxFee = 0;
        }

    }

    _tokenTransfer(from,to,amount);
}

function swapTokensForEth(uint256 tokenAmount) private lockTheSwap {
    address[] memory path = new address[](2);
    path[0] = address(this);
    path[1] = uniswapV2Router.WETH();
    _approve(address(this), address(uniswapV2Router), tokenAmount);
    uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
        tokenAmount,
        0,
        path,
        address(this),
        block.timestamp
    );
}

function sendETHToFee(uint256 amount) private {
    uint256 dAmount = amount.mul(_devPer).div(100);
    uint256 mAmount = amount.sub(dAmount);
    _twddev.transfer(dAmount);
    _twdmkt.transfer(mAmount);
}

function _tokenTransfer(address sender, address recipient, uint256 amount) private {
    _transferStandard(sender, recipient, amount);
}

function _transferStandard(address sender, address recipient, uint256 tAmount) private
    (uint256 rAmount, uint256 rTransferAmount, uint256 rFee, uint256 tTransferAmount, ui
    _rOwned[sender] = _rOwned[sender].sub(rAmount);
    _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount);
    _takeTeam(tTeam);
    _reflectFee(rFee, tFee);
    emit Transfer(sender, recipient, tTransferAmount);
}
```

```
function _takeTeam(uint256 tTeam) private {
    uint256 currentRate =  _getRate();
    uint256 rTeam = tTeam.mul(currentRate);
    _rOwned[address(this)] = _rOwned[address(this)].add(rTeam);
}

function _reflectFee(uint256 rFee, uint256 tFee) private {
    _rTotal = _rTotal.sub(rFee);
    _tFeeTotal = _tFeeTotal.add(tFee);
}

receive() external payable {}

function _getValues(uint256 tAmount) private view returns (uint256, uint256, uint256, ui
    (uint256 tTransferAmount, uint256 tFee, uint256 tTeam) = _getTValues(tAmount, _redis
    uint256 currentRate =  _getRate();
    (uint256 rAmount, uint256 rTransferAmount, uint256 rFee) = _getRValues(tAmount, tFee
    return (rAmount, rTransferAmount, rFee, tTransferAmount, tFee, tTeam);
}

function _getTValues(uint256 tAmount, uint256 taxFee, uint256 teamFee) private pure retu
    uint256 tFee = tAmount.mul(taxFee).div(100);
    uint256 tTeam = tAmount.mul(teamFee).div(100);
    uint256 tTransferAmount = tAmount.sub(tFee).sub(tTeam);
    return (tTransferAmount, tFee, tTeam);
}

function _getRValues(uint256 tAmount, uint256 tFee, uint256 tTeam, uint256 currentRate)
    uint256 rAmount = tAmount.mul(currentRate);
    uint256 rFee = tFee.mul(currentRate);
    uint256 rTeam = tTeam.mul(currentRate);
    uint256 rTransferAmount = rAmount.sub(rFee).sub(rTeam);
    return (rAmount, rTransferAmount, rFee);
}

function _getRate() private view returns(uint256) {
    (uint256 rSupply, uint256 tSupply) = _getCurrentSupply();
    return rSupply.div(tSupply);
}

function _getCurrentSupply() private view returns(uint256, uint256) {
    uint256 rSupply = _rTotal;
    uint256 tSupply = _tTotal;
    if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
    return (rSupply, tSupply);
}
```

```
}
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L1:1

NC015 - Contract declarations should have NatSpec descriptions:

e.g.0 `@dev` or `@notice`, and it must appear above the contract definition braces in order to be identified by the compiler as NatSpec

Click to show 8 findings

`File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol`

```
23      interface Token {
```

```
28      library SafeMath {
```

```
12      interface IERC20 {
```

```
89      interface IUniswapV2Factory {
```

```
93      interface IUniswapV2Router02 {
```

```
66      contract Ownable is Context {
```

```
6       abstract contract Context {
```

```
113     contract SAVER is Context, IERC20, Ownable {
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L113:113

NC016 - Function declarations should have NatSpec descriptions:

Function declarations should be preceded by a NatSpec comment.

Click to show 40 findings

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
13          function totalSupply() external view returns (uint256);

14          function balanceOf(address account) external view returns (uint256);

15          function transfer(address recipient, uint256 amount) external returns (bool);

16          function allowance(address owner, address spender) external view returns (uint25

17          function approve(address spender, uint256 amount) external returns (bool);

18          function transferFrom(address sender, address recipient, uint256 amount) externa

24          function transferFrom(address, address, uint) external returns (bool);

25          function transfer(address, uint) external returns (bool);

29          function add(uint256 a, uint256 b) internal pure returns (uint256) {

35          function sub(uint256 a, uint256 b) internal pure returns (uint256) {

39          function sub(uint256 a, uint256 b, string memory errorMessage) internal pure ret

45          function mul(uint256 a, uint256 b) internal pure returns (uint256) {

54          function div(uint256 a, uint256 b) internal pure returns (uint256) {

58          function div(uint256 a, uint256 b, string memory errorMessage) internal pure ret
```

```
70          constructor () {

75          function owner() public view returns (address) {

84          function renounceOwnership() public virtual onlyOwner {

90          function createPair(address tokenA, address tokenB) external returns (address pa

94          function swapExactTokensForETHSupportingFeeOnTransferTokens(

101         function factory() external pure returns (address);

102         function WETH() external pure returns (address);

103         function addLiquidityETH(

158         constructor () {

174         function name() public pure returns (string memory) {

178         function symbol() public pure returns (string memory) {

182         function decimals() public pure returns (uint8) {

186         function totalSupply() public pure override returns (uint256) {

190         function balanceOf(address account) public view override returns (uint256) {

194         function transfer(address recipient, uint256 amount) public override returns (bo
```

```
199         function allowance(address owner, address spender) public view override returns

203         function approve(address spender, uint256 amount) public override returns (bool)

208         function setIsExcludedFromFee(address account, bool newValue) public onlyOwner t

212         function setMaxWalletPer(uint256 newMaxWalletPer) public onlyOwner {

216         function setBuyFees(uint256 newBuyFee, uint256 newRedisBuyFee) public onlyOwner

221         function setSellFees(uint256 newSellFee, uint256 newRedisSellFee) public onlyOwn

226         function transferFrom(address sender, address recipient, uint256 amount) public

232         function tokenFromReflection(uint256 rAmount) private view returns(uint256) {

294         function swapTokensForEth(uint256 tokenAmount) private lockTheSwap {

308         function sendETHToFee(uint256 amount) private {

339         receive() external payable {}
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L339:339

NC017 - Contract declarations should have `@notice` tags:

`@notice` is used to explain to end users what the contract does, and the compiler interprets `///` or `/**` comments as this tag if one wasn't explicitly provided.

Click to show 8 findings

`File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol`

```
6       abstract contract Context {
```

```
12      interface IERC20 {
```

```
23      interface Token {
```

```
28      library SafeMath {
```

```
66      contract Ownable is Context {
```

```
89      interface IUniswapV2Factory {
```

```
93      interface IUniswapV2Router02 {
```

```
113     contract SAVER is Context, IERC20, Ownable {
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L113:113

NC018 - Contract declarations should have NatSpec @title annotations:

Click to show 8 findings

`File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol`

```
6       abstract contract Context {
```

```
12      interface IERC20 {
```

```
23      interface Token {
```

```
28      library SafeMath {
```

```
66      contract Ownable is Context {
```

```
89      interface IUniswapV2Factory {
```

```
93      interface IUniswapV2Router02 {
```

```
113     contract SAVER is Context, IERC20, Ownable {
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L113:113

NC019 - State variable declarations should have NatSpec descriptions:

e.g. `@notice` for public state variables, and `@dev` for non-public ones.

Click to show 29 findings

`File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol`

```
67          address private _owner;
```

```
68          address private _previousOwner;
```

```
116         mapping (address => uint256) private _rOwned;
```

```
117         mapping (address => uint256) private _tOwned;
```

```
118         mapping (address => mapping (address => uint256)) private _allowances;
```

```
119         mapping (address => bool) private _isExcludedFromFee;
```

```
120         mapping (address => bool) private _rewardAddress;
```

```
122         uint256 private constant MAX = ~uint256(0);
```

```
123         uint256 private constant _tTotal = 1000000000 * 10**9;

124         uint256 private _rTotal = (MAX - (MAX % _tTotal));

125         uint256 private _tFeeTotal;

127         uint256 private _redisFeeOnBuy = 0;

128         uint256 private _taxFeeOnBuy = 6;

130         uint256 private _redisFeeOnSell = 0;

131         uint256 private _taxFeeOnSell = 6;

133         uint256 private _devPer = 0;

134         uint256 public _maxWalletPer = 2;

136         uint256 private _redisFee;

137         uint256 private _taxFee;

139         address public immutable deadAddress = 0x000000000000000000000000000000000000dEa

140         address payable private _twddev = payable(0x55F84c660e27F630AF3112075e0D94c50F75

141         address payable private _twdmkt = payable(0x55F84c660e27F630AF3112075e0D94c50F75

143         string private constant _name = "Saver Protocol";
```

```
144         string private constant _symbol = "SVR";

145         uint8 private constant _decimals = 9;

147         IUniswapV2Router02 public uniswapV2Router;

148         address public uniswapV2Pair;

150         bool private inSwap = false;

151         bool private swapEnabled = true;
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L151:151

NC020 - Event declarations should have NatSpec descriptions:

`File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol`

```
19          event Transfer(address indexed from, address indexed to, uint256 value);

20          event Approval(address indexed owner, address indexed spender, uint256 value);
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L20:20

NC021 - Contract declarations should have NatSpec `@author` annotations:

`File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol`

```
// SPDX-License-Identifier: MIT

pragma solidity ^0.8.19;
```

```
abstract contract Context {
    function _msgSender() internal view virtual returns (address) {
        return msg.sender;
    }
}

interface IERC20 {
    function totalSupply() external view returns (uint256);
    function balanceOf(address account) external view returns (uint256);
    function transfer(address recipient, uint256 amount) external returns (bool);
    function allowance(address owner, address spender) external view returns (uint256);
    function approve(address spender, uint256 amount) external returns (bool);
    function transferFrom(address sender, address recipient, uint256 amount) external returr
    event Transfer(address indexed from, address indexed to, uint256 value);
    event Approval(address indexed owner, address indexed spender, uint256 value);
}

interface Token {
    function transferFrom(address, address, uint) external returns (bool);
    function transfer(address, uint) external returns (bool);
}

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

    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
        return sub(a, b, "Subtraction overflow");
    }

    function sub(uint256 a, uint256 b, string memory errorMessage) internal pure returns (ui
        require(b <= a, errorMessage);
        uint256 c = a - b;
        return c;
    }

    function mul(uint256 a, uint256 b) internal pure returns (uint256) {
        if (a == 0) {
            return 0;
        }
        uint256 c = a * b;
        require(c / a == b, "Multiplication overflow");
        return c;
```

```
    }

    function div(uint256 a, uint256 b) internal pure returns (uint256) {
        return div(a, b, "Division by zero");
    }

    function div(uint256 a, uint256 b, string memory errorMessage) internal pure returns (ui
        require(b > 0, errorMessage);
        uint256 c = a / b;
        return c;
    }

}

contract Ownable is Context {
    address private _owner;
    address private _previousOwner;

    constructor () {
        address msgSender = _msgSender();
        _owner = msgSender;
    }

    function owner() public view returns (address) {
        return _owner;
    }

    modifier onlyOwner() {
        require(_owner == _msgSender(), "Caller is not the owner");
        _;
    }

    function renounceOwnership() public virtual onlyOwner {
        _owner = address(0);
    }
}

interface IUniswapV2Factory {
    function createPair(address tokenA, address tokenB) external returns (address pair);
}

interface IUniswapV2Router02 {
    function swapExactTokensForETHSupportingFeeOnTransferTokens(
        uint amountIn,
        uint amountOutMin,
        address[] calldata path,
```

```
        address to,
        uint deadline
    ) external;
    function factory() external pure returns (address);
    function WETH() external pure returns (address);
    function addLiquidityETH(
        address token,
        uint amountTokenDesired,
        uint amountTokenMin,
        uint amountETHMin,
        address to,
        uint deadline
    ) external payable returns (uint amountToken, uint amountETH, uint liquidity);
}

contract SAVER is Context, IERC20, Ownable {

    using SafeMath for uint256;
    mapping (address => uint256) private _rOwned;
    mapping (address => uint256) private _tOwned;
    mapping (address => mapping (address => uint256)) private _allowances;
    mapping (address => bool) private _isExcludedFromFee;
    mapping (address => bool) private _rewardAddress;

    uint256 private constant MAX = ~uint256(0);
    uint256 private constant _tTotal = 1000000000 * 10**9;
    uint256 private _rTotal = (MAX - (MAX % _tTotal));
    uint256 private _tFeeTotal;

    uint256 private _redisFeeOnBuy = 0;
    uint256 private _taxFeeOnBuy = 6;

    uint256 private _redisFeeOnSell = 0;
    uint256 private _taxFeeOnSell = 6;

    uint256 private _devPer = 0;
    uint256 public _maxWalletPer = 2;

    uint256 private _redisFee;
    uint256 private _taxFee;

    address public immutable deadAddress = 0x000000000000000000000000000000000000dEaD;
    address payable private _twddev = payable(0x55F84c660e27F630AF3112075e0D94c50F75d1AB);
    address payable private _twdmkt = payable(0x55F84c660e27F630AF3112075e0D94c50F75d1AB);

    string private constant _name = "Saver Protocol";
```

```
string private constant _symbol = "SVR";
uint8 private constant _decimals = 9;

IUniswapV2Router02 public uniswapV2Router;
address public uniswapV2Pair;

bool private inSwap = false;
bool private swapEnabled = true;

modifier lockTheSwap {
    inSwap = true;
    _;
    inSwap = false;
}
constructor () {
    _rOwned[_msgSender()] = _rTotal;

    IUniswapV2Router02 _uniswapV2Router = IUniswapV2Router02(0x10ED43C718714eb63d5aA57B7
    uniswapV2Router = _uniswapV2Router;
    uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory())
        .createPair(address(this), _uniswapV2Router.WETH());

    _isExcludedFromFee[owner()] = true;
    _isExcludedFromFee[address(this)] = true;
    _isExcludedFromFee[_twddev] = true;
    _isExcludedFromFee[_twdmkt] = true;

    emit Transfer(address(0x0000000000000000000000000000000000000000), _msgSender(), _t
}

function name() public pure returns (string memory) {
    return _name;
}

function symbol() public pure returns (string memory) {
    return _symbol;
}

function decimals() public pure returns (uint8) {
    return _decimals;
}

function totalSupply() public pure override returns (uint256) {
    return _tTotal;
}
```

```
function balanceOf(address account) public view override returns (uint256) {
    return tokenFromReflection(_rOwned[account]);
}

function transfer(address recipient, uint256 amount) public override returns (bool) {
    _transfer(_msgSender(), recipient, amount);
    return true;
}

function allowance(address owner, address spender) public view override returns (uint256
    return _allowances[owner][spender];
}

function approve(address spender, uint256 amount) public override returns (bool) {
    _approve(_msgSender(), spender, amount);
    return true;
}

function setIsExcludedFromFee(address account, bool newValue) public onlyOwner {
    _isExcludedFromFee[account] = newValue;
}

function setMaxWalletPer(uint256 newMaxWalletPer) public onlyOwner {
    _maxWalletPer = newMaxWalletPer;
}

function setBuyFees(uint256 newBuyFee, uint256 newRedisBuyFee) public onlyOwner {
    _taxFeeOnBuy = newBuyFee;
    _redisFeeOnBuy = newRedisBuyFee;
}

function setSellFees(uint256 newSellFee, uint256 newRedisSellFee) public onlyOwner {
    _taxFeeOnSell = newSellFee;
    _redisFeeOnSell = newRedisSellFee;
}

function transferFrom(address sender, address recipient, uint256 amount) public override
    _transfer(sender, recipient, amount);
    _approve(sender, _msgSender(), _allowances[sender][_msgSender()].sub(amount, "Transf
    return true;
}

function tokenFromReflection(uint256 rAmount) private view returns(uint256) {
    require(rAmount <= _rTotal, "Amount must be less than total reflections");
    uint256 currentRate =  _getRate();
    return rAmount.div(currentRate);
```

```
}

function _approve(address owner, address spender, uint256 amount) private {
    require(owner != address(0), "Approve from the zero address");
    require(spender != address(0), "Approve to the zero address");
    _allowances[owner][spender] = amount;
    emit Approval(owner, spender, amount);
}

function _transfer(address from, address to, uint256 amount) private {
    require(from != address(0), "Transfer from the zero address");
    require(to != address(0), "Transfer to the zero address");
    require(amount > 0, "Transfer amount must be greater than zero");
    if (to != deadAddress &&
        to != address(this) &&
        to != uniswapV2Pair &&
        !_isExcludedFromFee[from] &&
        !_isExcludedFromFee[to]
    ) {
        uint256 heldTokens = balanceOf(to);
        uint256 maxWalletTokens = _maxWalletPer.mul(_tTotal).div(100);
        require((heldTokens + amount) <= maxWalletTokens, "Over wallet limit.");
    }

    _redisFee = 0;
    _taxFee = 0;

    if (from != owner() && to != owner()) {

        uint256 contractTokenBalance = balanceOf(address(this));
        if (!inSwap && from != uniswapV2Pair && swapEnabled && contractTokenBalance > 0)
            swapTokensForEth(contractTokenBalance);
            uint256 contractETHBalance = address(this).balance;
            if(contractETHBalance > 0) {
                sendETHToFee(address(this).balance);
            }
        }

        if(from == uniswapV2Pair && to != address(uniswapV2Router)) {
            _redisFee = _redisFeeOnBuy;
            _taxFee = _taxFeeOnBuy;
        }

        if (to == uniswapV2Pair && from != address(uniswapV2Router)) {
            _redisFee = _redisFeeOnSell;
            _taxFee = _taxFeeOnSell;
```

```
        }

        if ((_isExcludedFromFee[from] || _isExcludedFromFee[to]) || (from != uniswapV2Pa
            _redisFee = 0;
            _taxFee = 0;
        }

    }

    _tokenTransfer(from,to,amount);
}

function swapTokensForEth(uint256 tokenAmount) private lockTheSwap {
    address[] memory path = new address[](2);
    path[0] = address(this);
    path[1] = uniswapV2Router.WETH();
    _approve(address(this), address(uniswapV2Router), tokenAmount);
    uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
        tokenAmount,
        0,
        path,
        address(this),
        block.timestamp
    );
}

function sendETHToFee(uint256 amount) private {
    uint256 dAmount = amount.mul(_devPer).div(100);
    uint256 mAmount = amount.sub(dAmount);
    _twddev.transfer(dAmount);
    _twdmkt.transfer(mAmount);
}

function _tokenTransfer(address sender, address recipient, uint256 amount) private {
    _transferStandard(sender, recipient, amount);
}

function _transferStandard(address sender, address recipient, uint256 tAmount) private
    (uint256 rAmount, uint256 rTransferAmount, uint256 rFee, uint256 tTransferAmount, ui
    _rOwned[sender] = _rOwned[sender].sub(rAmount);
    _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount);
    _takeTeam(tTeam);
    _reflectFee(rFee, tFee);
    emit Transfer(sender, recipient, tTransferAmount);
}
```

```
function _takeTeam(uint256 tTeam) private {
    uint256 currentRate =  _getRate();
    uint256 rTeam = tTeam.mul(currentRate);
    _rOwned[address(this)] = _rOwned[address(this)].add(rTeam);
}

function _reflectFee(uint256 rFee, uint256 tFee) private {
    _rTotal = _rTotal.sub(rFee);
    _tFeeTotal = _tFeeTotal.add(tFee);
}

receive() external payable {}

function _getValues(uint256 tAmount) private view returns (uint256, uint256, uint256, ui
    (uint256 tTransferAmount, uint256 tFee, uint256 tTeam) = _getTValues(tAmount, _redis
    uint256 currentRate =  _getRate();
    (uint256 rAmount, uint256 rTransferAmount, uint256 rFee) = _getRValues(tAmount, tFee
    return (rAmount, rTransferAmount, rFee, tTransferAmount, tFee, tTeam);
}

function _getTValues(uint256 tAmount, uint256 taxFee, uint256 teamFee) private pure retu
    uint256 tFee = tAmount.mul(taxFee).div(100);
    uint256 tTeam = tAmount.mul(teamFee).div(100);
    uint256 tTransferAmount = tAmount.sub(tFee).sub(tTeam);
    return (tTransferAmount, tFee, tTeam);
}

function _getRValues(uint256 tAmount, uint256 tFee, uint256 tTeam, uint256 currentRate)
    uint256 rAmount = tAmount.mul(currentRate);
    uint256 rFee = tFee.mul(currentRate);
    uint256 rTeam = tTeam.mul(currentRate);
    uint256 rTransferAmount = rAmount.sub(rFee).sub(rTeam);
    return (rAmount, rTransferAmount, rFee);
}

function _getRate() private view returns(uint256) {
    (uint256 rSupply, uint256 tSupply) = _getCurrentSupply();
    return rSupply.div(tSupply);
}

function _getCurrentSupply() private view returns(uint256, uint256) {
    uint256 rSupply = _rTotal;
    uint256 tSupply = _tTotal;
    if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
    return (rSupply, tSupply);
}
```

```
}
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L1:1

NC022 - Function declarations should have `@notice` tags:

`@notice` is used to explain to end users what the function does, and the compiler interprets `///` or `/**` comments as this tag if one wasn't explicitly provided.

`File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol`

```
339         receive() external payable {}
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L339:339

NC023 - Contract does not follow the Solidity style guide's suggested layout ordering:

The style guide says that, within a contract, the ordering should be 1) Type declarations, 2) State variables, 3) Events, 4) Modifiers, and 5) Functions, but the contract(s) below do not follow this ordering.

`File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol`

```
19          event Transfer(address indexed from, address indexed to, uint256 value);
```

```
79          modifier onlyOwner() {
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L79:79

NC024 - Non-external/public variable names should begin with an underscore:

According to the Solidity Style Guide, non-external/public variable names should begin with an underscore

`File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol`

```
150         bool private inSwap = false;
```

```
151         bool private swapEnabled = true;
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L151:151

NC025 - Consider disabling `renounceOwnership()`:

If the plan for your project does not include eventually giving up all ownership control, consider overwriting OpenZeppelin's Ownable's `renounceOwnership()` function in order to disable it.

`File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol`

```
113     contract SAVER is Context, IERC20, Ownable {
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L113:113

NC026 - Non-library/interface files should use fixed compiler versions, not floating ones:

Using a floating compiler version like ^0.8.16 or >=0.8.16 can lead to unexpected behavior if the compiler version used differs from the one intended. It's recommended to specify a fixed compiler version for non-library/interface files.

`File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol`

```
3       pragma solidity ^0.8.19;
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L3:3

NC027 - Contracts should have full test coverage:

While 100% code coverage does not guarantee that there are no bugs, it often will catch easy-to-find bugs, and will ensure that there are fewer regressions when the code invariably has to be modified. Furthermore, in order to get full coverage, code authors will often have to re-organize their code so that it is more modular, so that each component can be tested separately, which reduces interdependencies between modules and layers, and makes for code that is easier to reason about and audit.

File: Various Files

None

NC028 - Large or complicated code bases should implement invariant tests:

Large code bases, or code with lots of inline-assembly, complicated math, or complicated interactions between multiple contracts, should implement invariant fuzzing tests. Invariant fuzzers such as Echidna require the test writer to come up with invariants which should not be violated under any circumstances, and the fuzzer tests various inputs and function calls to ensure that the invariants always hold. Even code with 100% code coverage can still have bugs due to the order of the operations a user performs, and invariant fuzzers, with properly and extensively-written invariants, can close this testing gap significantly.

File: Various Files

None

NC029 - `address` shouldn't be hard-coded:

It is often better to declare addresses as `immutable`, and assign them via constructor arguments. This allows the code to remain the same across deployments on different networks and avoids recompilation when addresses need to change.

Click to show 5 findings

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
139         address public immutable deadAddress = 0x000000000000000000000000000000000000dEa

140         address payable private _twddev = payable(0x55F84c660e27F630AF3112075e0D94c50F75

141         address payable private _twdmkt = payable(0x55F84c660e27F630AF3112075e0D94c50F75

161             IUniswapV2Router02 _uniswapV2Router = IUniswapV2Router02(0x10ED43C718714eb63

171             emit Transfer(address(0x0000000000000000000000000000000000000000), _msgSende
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L171:171

NC030 - Consider using `block.number` instead of `block.timestamp`:

`block.timestamp` is vulnerable to miner manipulation and creates a potential front-running vulnerability. Consider using `block.number` instead.

`File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol`

```
304                 block.timestamp
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L304:304

NC031 - Large numeric literals should use underscores for readability:

At a glance, it's quite difficult to understand how big this number is. Use underscores to make values more clear.

`File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol`

```
123         uint256 private constant _tTotal = 1000000000 * 10**9;
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L123:123

NC032 - Variables should be named in mixedCase style:

As the Solidity Style Guide suggests: arguments, local variables and mutable state variables should be named in mixedCase style.

`File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol`

```
268                     uint256 contractETHBalance = address(this).balance;
```

```
107             uint amountETHMin,
```

```
90          function createPair(address tokenA, address tokenB) external returns (address pa
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L90:90

NC033 - Function names should use lowerCamelCase:

According to the Solidity style guide function names should be in `mixedCase` (lowerCamelCase).

Click to show 6 findings

`File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol`

```
94          function swapExactTokensForETHSupportingFeeOnTransferTokens(
```

```
102         function WETH() external pure returns (address);
```

```
103         function addLiquidityETH(
```

```
308         function sendETHToFee(uint256 amount) private {
```

```
348         function _getTValues(uint256 tAmount, uint256 taxFee, uint256 teamFee) private p
```

```
355         function _getRValues(uint256 tAmount, uint256 tFee, uint256 tTeam, uint256 curre
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L355:355

NC034 - Consider adding a deny-list:

Doing so will significantly increase centralization, but will help to prevent hackers from using stolen tokens.

`File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol`

```
66      contract Ownable is Context {
```

```
113     contract SAVER is Context, IERC20, Ownable {
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L113:113

NC035 - Custom errors should be used rather than `revert()/require()`:

Custom errors are available from solidity version 0.8.4. Custom errors are more easily processed in `try-catch` blocks, and are easier to re-use and maintain.

Click to show 8 findings

`File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol`

```
80              require(_owner == _msgSender(), "Caller is not the owner");
```

```
233             require(rAmount <= _rTotal, "Amount must be less than total reflections");
```

```
239             require(owner != address(0), "Approve from the zero address");
```

```
240             require(spender != address(0), "Approve to the zero address");
```

```
246             require(from != address(0), "Transfer from the zero address");
```

```
247             require(to != address(0), "Transfer to the zero address");
```

```
248             require(amount > 0, "Transfer amount must be greater than zero");
```

```
257                 require((heldTokens + amount) <= maxWalletTokens, "Over wallet limit.");
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L257:257

NC036 - Top level declarations should be separated by two blank lines:

Click to show 7 findings

`File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol`

27

65

112

92

22

11

88

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L88:88

NC037 - Interfaces should be defined in separate files from their usage:

This issue arises when the interfaces are defined in the same files where they are used. They should be separated into different files for better readability and reusability.

Click to show 4 findings

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
12      interface IERC20 {
13          function totalSupply() external view returns (uint256);
14          function balanceOf(address account) external view returns (uint256);
15          function transfer(address recipient, uint256 amount) external returns (bool);
16          function allowance(address owner, address spender) external view returns (uint25
17          function approve(address spender, uint256 amount) external returns (bool);
18          function transferFrom(address sender, address recipient, uint256 amount) externa
19          event Transfer(address indexed from, address indexed to, uint256 value);
20          event Approval(address indexed owner, address indexed spender, uint256 value);
21      }
```

```
23      interface Token {
24          function transferFrom(address, address, uint) external returns (bool);
25          function transfer(address, uint) external returns (bool);
26      }


89      interface IUniswapV2Factory {
90          function createPair(address tokenA, address tokenB) external returns (address pa
91      }


93      interface IUniswapV2Router02 {
94          function swapExactTokensForETHSupportingFeeOnTransferTokens(
95              uint amountIn,
96              uint amountOutMin,
97              address[] calldata path,
98              address to,
99              uint deadline
100         ) external;
101         function factory() external pure returns (address);
102         function WETH() external pure returns (address);
103         function addLiquidityETH(
104             address token,
105             uint amountTokenDesired,
106             uint amountTokenMin,
107             uint amountETHMin,
108             address to,
109             uint deadline
110         ) external payable returns (uint amountToken, uint amountETH, uint liquidity);
111     }
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L93:111

NC038 - Enum values should be used in place of constant array indexes:

Create a commented enum value to use in place of constant array indexes, this makes the code far easier to understand.

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
296             path[0] = address(this);
```

```
297             path[1] = uniswapV2Router.WETH();
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L297:297

NC039 - Zero as a function argument should have a descriptive meaning:

Consider using descriptive constants or an enum instead of passing zero directly on function calls, as that might be error-prone, to fully describe the caller's intention.

Click to show 13 findings

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
122         uint256 private constant MAX = ~uint256(0);
```

```
122         uint256 private constant MAX = ~uint256(0);
```

```
140         address payable private _twddev = payable(0x55F84c660e27F630AF3112075e0D94c50F75
```

```
140         address payable private _twddev = payable(0x55F84c660e27F630AF3112075e0D94c50F75
```

```
141         address payable private _twdmkt = payable(0x55F84c660e27F630AF3112075e0D94c50F75
```

```
141         address payable private _twdmkt = payable(0x55F84c660e27F630AF3112075e0D94c50F75
```

```
161             IUniswapV2Router02 _uniswapV2Router = IUniswapV2Router02(0x10ED43C718714eb63
```

```
256                 uint256 maxWalletTokens = _maxWalletPer.mul(_tTotal).div(100);
```

```
295             address[] memory path = new address[](2);
```

```
299             uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
300                 tokenAmount,
301                 0,
302                 path,
303                 address(this),
304                 block.timestamp
305             );


309             uint256 dAmount = amount.mul(_devPer).div(100);


349             uint256 tFee = tAmount.mul(taxFee).div(100);


350             uint256 tTeam = tAmount.mul(teamFee).div(100);
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L350:350

NC040 - Function names should differ to make the code more readable:

In Solidity, while function overriding allows for functions with the same name to coexist, it is advisable to avoid this practice to enhance code readability and maintainability. Having multiple functions with the same name, even with different parameters or in inherited contracts, can cause confusion and increase the likelihood of errors during development, testing, and debugging. Using distinct and descriptive function names not only clarifies the purpose and behavior of each function, but also helps prevent unintended function calls or incorrect overriding. By adopting a clear and consistent naming convention, developers can create more comprehensible and maintainable smart contracts.

Click to show 18 findings

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
    function totalSupply() external view returns (uint256);

    function totalSupply() public pure override returns (uint256) {
        return _tTotal;
    }

    function balanceOf(address account) external view returns (uint256);
```

```
function balanceOf(address account) public view override returns (uint256) {
    return tokenFromReflection(_rOwned[account]);
}

function transfer(address recipient, uint256 amount) external returns (bool);

function transfer(address, uint) external returns (bool);

function transfer(address recipient, uint256 amount) public override returns (bool) {
    _transfer(_msgSender(), recipient, amount);
    return true;
}

function allowance(address owner, address spender) external view returns (uint256);

function allowance(address owner, address spender) public view override returns (uint256
    return _allowances[owner][spender];
}

function approve(address spender, uint256 amount) external returns (bool);

function approve(address spender, uint256 amount) public override returns (bool) {
    _approve(_msgSender(), spender, amount);
    return true;
}

function transferFrom(address sender, address recipient, uint256 amount) external return

function transferFrom(address, address, uint) external returns (bool);

function transferFrom(address sender, address recipient, uint256 amount) public override
    _transfer(sender, recipient, amount);
    _approve(sender, _msgSender(), _allowances[sender][_msgSender()].sub(amount, "Transf
    return true;
}

function sub(uint256 a, uint256 b) internal pure returns (uint256) {
    return sub(a, b, "Subtraction overflow");
}

function sub(uint256 a, uint256 b, string memory errorMessage) internal pure returns (ui
    require(b <= a, errorMessage);
    uint256 c = a - b;
    return c;
}
```

```
    function div(uint256 a, uint256 b) internal pure returns (uint256) {
        return div(a, b, "Division by zero");
    }

    function div(uint256 a, uint256 b, string memory errorMessage) internal pure returns (ui
        require(b > 0, errorMessage);
        uint256 c = a / b;
        return c;
    }
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L58:62

NC041 - It is standard for all external and public functions to be override from an interface:

This is to ensure the whole API is extracted in an interface

Click to show 16 findings

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
75          function owner() public view returns (address) {
76              return _owner;
77          }


84          function renounceOwnership() public virtual onlyOwner {
85              _owner = address(0);
86          }


174         function name() public pure returns (string memory) {
175             return _name;
176         }


178         function symbol() public pure returns (string memory) {
179             return _symbol;
180         }


182         function decimals() public pure returns (uint8) {
183             return _decimals;
184         }
```

```
186         function totalSupply() public pure override returns (uint256) {
187             return _tTotal;
188         }


190         function balanceOf(address account) public view override returns (uint256) {
191             return tokenFromReflection(_rOwned[account]);
192         }


194         function transfer(address recipient, uint256 amount) public override returns (bo
195             _transfer(_msgSender(), recipient, amount);
196             return true;
197         }


199         function allowance(address owner, address spender) public view override returns
200             return _allowances[owner][spender];
201         }


203         function approve(address spender, uint256 amount) public override returns (bool)
204             _approve(_msgSender(), spender, amount);
205             return true;
206         }


208         function setIsExcludedFromFee(address account, bool newValue) public onlyOwner {
209             _isExcludedFromFee[account] = newValue;
210         }


212         function setMaxWalletPer(uint256 newMaxWalletPer) public onlyOwner {
213             _maxWalletPer = newMaxWalletPer;
214         }


216         function setBuyFees(uint256 newBuyFee, uint256 newRedisBuyFee) public onlyOwner
217             _taxFeeOnBuy = newBuyFee;
218             _redisFeeOnBuy = newRedisBuyFee;
219         }


221         function setSellFees(uint256 newSellFee, uint256 newRedisSellFee) public onlyOwn
222             _taxFeeOnSell = newSellFee;
```

```
223             _redisFeeOnSell = newRedisSellFee;
224         }


226         function transferFrom(address sender, address recipient, uint256 amount) public
227             _transfer(sender, recipient, amount);
228             _approve(sender, _msgSender(), _allowances[sender][_msgSender()].sub(amount,
229             return true;
230         }


339         receive() external payable {}
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L339:339

NC042 - Consider adding formal verification proofs:

Consider using formal verification to mathematically prove that your code does what is intended, and does not have any edge cases with unexpected behavior. The solidity compiler itself has this functionality built in based off of SMTChecker.

`File: Various Files`

```
None
```

NC043 - Public state variables shouldn't have a preceding __ in their name:

Remove the _ from the state variable name, ensure you also refactor where these state variables are internally called.

`File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol`

```
134         uint256 public _maxWalletPer = 2;
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L134:134

NC044 - Large multiples of ten should use scientific notation (e.g. 1e6) rather than decimal literals (e.g. 1000000), for readability:

Using scientific notation for large multiples of ten improves code readability. Instead of writing large decimal literals, consider using scientific notation.

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
123         uint256 private constant _tTotal = 1000000000 * 10**9;
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L123:123

NC045 - Polymorphic functions make security audits more time-consuming and error-prone:

The instances below point to one of two functions with the same name. Consider naming each function differently, in order to make code navigation and analysis easier.

Click to show 9 findings

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
186         function totalSupply() public pure override returns (uint256) {

190         function balanceOf(address account) public view override returns (uint256) {

25          function transfer(address, uint) external returns (bool);

199         function allowance(address owner, address spender) public view override returns

203         function approve(address spender, uint256 amount) public override returns (bool)

24          function transferFrom(address, address, uint) external returns (bool);

39          function sub(uint256 a, uint256 b, string memory errorMessage) internal pure ret
```

```
58          function div(uint256 a, uint256 b, string memory errorMessage) internal pure ret
```

```
158         constructor () {
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L158:158

NC046 - Missing event and or timelock for critical parameter change:

Events help non-contract tools to track changes, and timelocks prevent users from being surprised by changes.

Click to show 5 findings

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
213             _maxWalletPer = newMaxWalletPer;
```

```
217             _taxFeeOnBuy = newBuyFee;
```

```
218             _redisFeeOnBuy = newRedisBuyFee;
```

```
222             _taxFeeOnSell = newSellFee;
```

```
223             _redisFeeOnSell = newRedisSellFee;
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L223:223

NC047 - Setters should prevent re-setting of the same value:

This especially problematic when the setter also emits the same value, which may be confusing to offline parsers.

Click to show 4 findings

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
208         function setIsExcludedFromFee(address account, bool newValue) public onlyOwner {
209             _isExcludedFromFee[account] = newValue;
210         }


212         function setMaxWalletPer(uint256 newMaxWalletPer) public onlyOwner {
213             _maxWalletPer = newMaxWalletPer;
214         }


216         function setBuyFees(uint256 newBuyFee, uint256 newRedisBuyFee) public onlyOwner
217             _taxFeeOnBuy = newBuyFee;
218             _redisFeeOnBuy = newRedisBuyFee;
219         }


221         function setSellFees(uint256 newSellFee, uint256 newRedisSellFee) public onlyOwn
222             _taxFeeOnSell = newSellFee;
223             _redisFeeOnSell = newRedisSellFee;
224         }
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L221:224

NC048 - High cyclomatic complexity:

Consider breaking down these blocks into more manageable units, by splitting things into utility functions, by reducing nesting, and by using early returns.

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
245         function _transfer(address from, address to, uint256 amount) private {
246             require(from != address(0), "Transfer from the zero address");
247             require(to != address(0), "Transfer to the zero address");
248             require(amount > 0, "Transfer amount must be greater than zero");
249             if (to != deadAddress &&
250                 to != address(this) &&
251                 to != uniswapV2Pair &&
252                 !_isExcludedFromFee[from] &&
253                 !_isExcludedFromFee[to]
254             ) {
255                 uint256 heldTokens = balanceOf(to);
256                 uint256 maxWalletTokens = _maxWalletPer.mul(_tTotal).div(100);
257                 require((heldTokens + amount) <= maxWalletTokens, "Over wallet limit.");
```

```
258             }
259
260             _redisFee = 0;
261             _taxFee = 0;
262
263             if (from != owner() && to != owner()) {
264
265                 uint256 contractTokenBalance = balanceOf(address(this));
266                 if (!inSwap && from != uniswapV2Pair && swapEnabled && contractTokenBala
267                     swapTokensForEth(contractTokenBalance);
268                     uint256 contractETHBalance = address(this).balance;
269                     if(contractETHBalance > 0) {
270                         sendETHToFee(address(this).balance);
271                     }
272                 }
273
274                 if(from == uniswapV2Pair && to != address(uniswapV2Router)) {
275                     _redisFee = _redisFeeOnBuy;
276                     _taxFee = _taxFeeOnBuy;
277                 }
278
279                 if (to == uniswapV2Pair && from != address(uniswapV2Router)) {
280                     _redisFee = _redisFeeOnSell;
281                     _taxFee = _taxFeeOnSell;
282                 }
283
284                 if ((_isExcludedFromFee[from] || _isExcludedFromFee[to]) || (from != uni
285                     _redisFee = 0;
286                     _taxFee = 0;
287                 }
288
289             }
290
291             _tokenTransfer(from,to,amount);
292         }
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L245:292

NC049 - Use of override is unnecessary:

Starting with Solidity version 0.8.8, using the override keyword when the function solely overrides an interface function, and the function doesn't exist in multiple base contracts, is unnecessary.

Click to show 6 findings

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
    function totalSupply() public pure override returns (uint256) {
        return _tTotal;
    }

    function balanceOf(address account) public view override returns (uint256) {
        return tokenFromReflection(_rOwned[account]);
    }

    function transfer(address recipient, uint256 amount) public override returns (bool) {
        _transfer(_msgSender(), recipient, amount);
        return true;
    }

    function allowance(address owner, address spender) public view override returns (uint256
        return _allowances[owner][spender];
    }

    function approve(address spender, uint256 amount) public override returns (bool) {
        _approve(_msgSender(), spender, amount);
        return true;
    }

    function transferFrom(address sender, address recipient, uint256 amount) public override
        _transfer(sender, recipient, amount);
        _approve(sender, _msgSender(), _allowances[sender][_msgSender()].sub(amount, "Transf
        return true;
    }
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L226:230

NC050 - Non-`external`/`public` function names should begin with an underscore:

According to the Solidity Style Guide, non-`external`/`public` function names should begin with an underscore

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
232         function tokenFromReflection(uint256 rAmount) private view returns(uint256) {
```

```
294         function swapTokensForEth(uint256 tokenAmount) private lockTheSwap {
```

```
308         function sendETHToFee(uint256 amount) private {
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L308:308

NC051 - Contracts/libraries should each be defined in separate files:

This helps to make tracking changes across commits easier, among other reasons. The instances below are the second+ contract/library within each file.

`File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol`

```
66      contract Ownable is Context {
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L66:66

NC052 - Consider adding emergency-stop functionality:

Adding a way to quickly halt protocol functionality in an emergency, rather than having to pause individual contracts one-by-one, will make in-progress hack mitigation faster and much less stressful.

`File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol`

```
113     contract SAVER is Context, IERC20, Ownable {
```

```
66      contract Ownable is Context {
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L66:66

NC053 - Named imports of parent contracts are missing:

`File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol`

```
113     contract SAVER is Context, IERC20, Ownable {
```

```
66      contract Ownable is Context {
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L66:66

NC054 - NatSpec: Contract declarations should have @dev tags:

@dev is used to explain extra details to developers

Click to show 8 findings

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
6       abstract contract Context {
```

```
12      interface IERC20 {
```

```
23      interface Token {
```

```
28      library SafeMath {
```

```
66      contract Ownable is Context {
```

```
89      interface IUniswapV2Factory {
```

```
93      interface IUniswapV2Router02 {
```

```
113     contract SAVER is Context, IERC20, Ownable {
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L113:113

NC055 - NatSpec: Function @param tag is missing:

Click to show 37 findings

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
14          function balanceOf(address account) external view returns (uint256);

15          function transfer(address recipient, uint256 amount) external returns (bool);

16          function allowance(address owner, address spender) external view returns (uint25

17          function approve(address spender, uint256 amount) external returns (bool);

18          function transferFrom(address sender, address recipient, uint256 amount) externa

24          function transferFrom(address, address, uint) external returns (bool);

25          function transfer(address, uint) external returns (bool);

29          function add(uint256 a, uint256 b) internal pure returns (uint256) {

35          function sub(uint256 a, uint256 b) internal pure returns (uint256) {

39          function sub(uint256 a, uint256 b, string memory errorMessage) internal pure ret

45          function mul(uint256 a, uint256 b) internal pure returns (uint256) {

54          function div(uint256 a, uint256 b) internal pure returns (uint256) {

58          function div(uint256 a, uint256 b, string memory errorMessage) internal pure ret

90          function createPair(address tokenA, address tokenB) external returns (address pa

94          function swapExactTokensForETHSupportingFeeOnTransferTokens(
```

```
103         function addLiquidityETH(

190         function balanceOf(address account) public view override returns (uint256) {

194         function transfer(address recipient, uint256 amount) public override returns (bo

199         function allowance(address owner, address spender) public view override returns

203         function approve(address spender, uint256 amount) public override returns (bool)

208         function setIsExcludedFromFee(address account, bool newValue) public onlyOwner {

212         function setMaxWalletPer(uint256 newMaxWalletPer) public onlyOwner {

216         function setBuyFees(uint256 newBuyFee, uint256 newRedisBuyFee) public onlyOwner

221         function setSellFees(uint256 newSellFee, uint256 newRedisSellFee) public onlyOwn

226         function transferFrom(address sender, address recipient, uint256 amount) public

232         function tokenFromReflection(uint256 rAmount) private view returns(uint256) {

238         function _approve(address owner, address spender, uint256 amount) private {

245         function _transfer(address from, address to, uint256 amount) private {

294         function swapTokensForEth(uint256 tokenAmount) private lockTheSwap {

308         function sendETHToFee(uint256 amount) private {
```

```
315         function _tokenTransfer(address sender, address recipient, uint256 amount) priva

319         function _transferStandard(address sender, address recipient, uint256 tAmount) p

328         function _takeTeam(uint256 tTeam) private {

334         function _reflectFee(uint256 rFee, uint256 tFee) private {

341         function _getValues(uint256 tAmount) private view returns (uint256, uint256, uin

348         function _getTValues(uint256 tAmount, uint256 taxFee, uint256 teamFee) private p

355         function _getRValues(uint256 tAmount, uint256 tFee, uint256 tTeam, uint256 curre
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L355:355

NC056 - Pure function accesses storage:

While the compiler currently flags functions like these as being pure, this is a bug which will be fixed in a future version, so it's best to not use pure visibility, in order to not break when this bug is fixed.

Click to show 12 findings

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
29          function add(uint256 a, uint256 b) internal pure returns (uint256) {
30              uint256 c = a + b;
31              require(c >= a, "Addition overflow");
32              return c;
33          }

35          function sub(uint256 a, uint256 b) internal pure returns (uint256) {
36              return sub(a, b, "Subtraction overflow");
37          }
```

```
39          function sub(uint256 a, uint256 b, string memory errorMessage) internal pure ret
40              require(b <= a, errorMessage);
41              uint256 c = a - b;
42              return c;
43          }


45          function mul(uint256 a, uint256 b) internal pure returns (uint256) {
46              if (a == 0) {
47                  return 0;
48              }
49              uint256 c = a * b;
50              require(c / a == b, "Multiplication overflow");
51              return c;
52          }


54          function div(uint256 a, uint256 b) internal pure returns (uint256) {
55              return div(a, b, "Division by zero");
56          }


58          function div(uint256 a, uint256 b, string memory errorMessage) internal pure ret
59              require(b > 0, errorMessage);
60              uint256 c = a / b;
61              return c;
62          }


174         function name() public pure returns (string memory) {
175             return _name;
176         }


178         function symbol() public pure returns (string memory) {
179             return _symbol;
180         }


182         function decimals() public pure returns (uint8) {
183             return _decimals;
184         }


186         function totalSupply() public pure override returns (uint256) {
```

```
187             return _tTotal;
188         }


348         function _getTValues(uint256 tAmount, uint256 taxFee, uint256 teamFee) private p
349             uint256 tFee = tAmount.mul(taxFee).div(100);
350             uint256 tTeam = tAmount.mul(teamFee).div(100);
351             uint256 tTransferAmount = tAmount.sub(tFee).sub(tTeam);
352             return (tTransferAmount, tFee, tTeam);
353         }


355         function _getRValues(uint256 tAmount, uint256 tFee, uint256 tTeam, uint256 curre
356             uint256 rAmount = tAmount.mul(currentRate);
357             uint256 rFee = tFee.mul(currentRate);
358             uint256 rTeam = tTeam.mul(currentRate);
359             uint256 rTransferAmount = rAmount.sub(rFee).sub(rTeam);
360             return (rAmount, rTransferAmount, rFee);
361         }
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L355:361

G001 - Don't Initialize Variables with Default Value:

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
uint256 private _redisFeeOnBuy = 0;

uint256 private _redisFeeOnSell = 0;

uint256 private _devPer = 0;
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L133:133

G002 - Use != 0 instead of > 0 for Unsigned Integer Comparison:

Click to show 4 findings

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
require(b > 0, errorMessage);
```

```
require(amount > 0, "Transfer amount must be greater than zero");

if (!inSwap && from != uniswapV2Pair && swapEnabled && contractTokenBalance > 0) {

if(contractETHBalance > 0) {
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L269:269

G003 - Using `bool` for storage incurs overhead:

Use uint256(1) and uint256(2) for true/false to avoid a Gwarmaccess (100 gas), and to avoid Gsset (20000 gas) when changing from 'false' to 'true', after having been 'true' in the past. See source.

```
File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol


150         bool private inSwap = false;


151         bool private swapEnabled = true;
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L151:151

G004 - Long Revert Strings:

```
File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol


require(rAmount <= _rTotal, "Amount must be less than total reflections");

require(amount > 0, "Transfer amount must be greater than zero");
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L248:248

G005 - Functions guaranteed to revert when called by normal users can be marked `payable`:

If a function modifier such as `onlyOwner` is used, the function will revert if a normal user tries to pay the function. Marking the function as `payable` will lower the gas cost for legitimate callers because the compiler will not include checks for whether a payment was provided.

Click to show 5 findings

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
function renounceOwnership() public virtual onlyOwner {

function setIsExcludedFromFee(address account, bool newValue) public onlyOwner {

function setMaxWalletPer(uint256 newMaxWalletPer) public onlyOwner {

function setBuyFees(uint256 newBuyFee, uint256 newRedisBuyFee) public onlyOwner {

function setSellFees(uint256 newSellFee, uint256 newRedisSellFee) public onlyOwner {
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L221:221

G006 - Use Custom Errors:

Instead of using error strings, to reduce deployment and runtime cost, you should use Custom Errors. This would save both deployment and runtime cost.

Source: https://consensys.net/diligence/blog/2019/09/stop-using-string-error-messages/

Click to show 8 findings

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
80              require(_owner == _msgSender(), "Caller is not the owner");

233             require(rAmount <= _rTotal, "Amount must be less than total reflections");

239             require(owner != address(0), "Approve from the zero address");

240             require(spender != address(0), "Approve to the zero address");

246             require(from != address(0), "Transfer from the zero address");

247             require(to != address(0), "Transfer to the zero address");
```

```
248             require(amount > 0, "Transfer amount must be greater than zero");
```

```
257                 require((heldTokens + amount) <= maxWalletTokens, "Over wallet limit.");
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L257:257

G007 - Use assembly to check for `address(0)`:

Saves 6 gas per instance

Click to show 4 findings

`File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol`

```
239             require(owner != address(0), "Approve from the zero address");
```

```
240             require(spender != address(0), "Approve to the zero address");
```

```
246             require(from != address(0), "Transfer from the zero address");
```

```
247             require(to != address(0), "Transfer to the zero address");
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L247:247

G008 - State variables should be cached in stack variables rather than re-reading them from storage:

The instances below point to the second+ access of a state variable within a function. Caching of a state variable replaces each Gwarmaccess (100 gas) with a much cheaper stack read. Other less obvious fixes/optimizations include having local memory caches of state variable structs, or having local caches of state variable contracts/addresses.

Saves 100 gas per instance

Click to show 11 findings

`File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol`

```
169             _isExcludedFromFee[_twdmkt] = true;

279                 if (to == uniswapV2Pair && from != address(uniswapV2Router)) {

284                 if ((_isExcludedFromFee[from] || _isExcludedFromFee[to]) || (from != uni

285                     _redisFee = 0;

286                     _taxFee = 0;

299             uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(

322             _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount);

331             _rOwned[address(this)] = _rOwned[address(this)].add(rTeam);

335             _rTotal = _rTotal.sub(rFee);

336             _tFeeTotal = _tFeeTotal.add(tFee);

371             if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L371:371

G009 - Multiple address/ID mappings can be combined into a single mapping of an address/ID to a struct, where appropriate:

Saves a storage slot for the mapping. Depending on the circumstances and sizes of types, can avoid a Gsset (20000 gas) per mapping combined. Reads and subsequent writes can also be cheaper when a function requires both values and they both fit in the same storage slot. Finally, if both fields are accessed in the same function, can save ~42 gas per access due to not having to recalculate the

key's keccak256 hash (Gkeccak256 - 30 gas) and that calculation's associated stack operations.

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
116         mapping (address => uint256) private _rOwned;
117         mapping (address => uint256) private _tOwned;
118         mapping (address => mapping (address => uint256)) private _allowances;
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L116:118

G010 - Multiple accesses of a mapping/array should use a local variable cache:

The instances below point to the second+ access of a value inside a mapping/array, within a function. Caching a mapping's value in a local storage or calldata variable when the value is accessed multiple times, saves ~42 gas per access due to not having to recalculate the key's keccak256 hash (Gkeccak256 - 30 gas) and that calculation's associated stack operations. Caching an array's struct avoids recalculating the array offsets into memory/calldata

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
322             _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount);
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L322:322

G011 - Internal functions only called once can be inlined to save gas:

Not inlining costs 20 to 40 gas because of two extra JUMP instructions and additional stack operations needed for function calls.

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
39          function sub(uint256 a, uint256 b, string memory errorMessage) internal pure ret
```

```
58          function div(uint256 a, uint256 b, string memory errorMessage) internal pure ret
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L58:58

G012 - Add unchecked {} for subtractions where the operands cannot underflow because of a previous require() or if-statement:

`require(a <= b); x = b - a => require(a <= b); unchecked { x = b - a }`

`File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol`

```
41              uint256 c = a - b;
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L41:41

G013 - Optimize names to save gas:

public/external function names and public member variable names can be optimized to save gas.

`File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol`

```
6       abstract contract Context {
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L6:6

G014 - Constructors can be marked payable:

Payable functions cost less gas to execute, since the compiler does not have to add extra checks to ensure that a payment wasn't provided. A constructor can safely be marked as payable, since only the deployer would be able to pass funds, and the project itself would not pass any funds.

`File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol`

```
70          constructor () {
71              address msgSender = _msgSender();
72              _owner = msgSender;
73          }
```

```
158         constructor () {
159             _rOwned[_msgSender()] = _rTotal;
160 
161             IUniswapV2Router02 _uniswapV2Router = IUniswapV2Router02(0x10ED43C718714eb63
162             uniswapV2Router = _uniswapV2Router;
163             uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory())
164                 .createPair(address(this), _uniswapV2Router.WETH());
165 
166             _isExcludedFromFee[owner()] = true;
167             _isExcludedFromFee[address(this)] = true;
168             _isExcludedFromFee[_twddev] = true;
169             _isExcludedFromFee[_twdmkt] = true;
170 
171             emit Transfer(address(0x0000000000000000000000000000000000000000), _msgSende
172         }
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L158:172

G015 - Remove unused variables:

Removing unused variables saves gas.

Click to show 5 findings

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
19          event Transfer(address indexed from, address indexed to, uint256 value);
```

```
20          event Approval(address indexed owner, address indexed spender, uint256 value);
```

```
68          address private _previousOwner;
```

```
117         mapping (address => uint256) private _tOwned;
```

```
120         mapping (address => bool) private _rewardAddress;
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L120:120

G016 - Use solidity version 0.8.20 or above to improve gas performance:

Upgrade to the latest solidity version 0.8.20 to get additional gas savings. See the latest release for reference: https://blog.soliditylang.org/2023/05/10/solidity-0.8.20-release-announcement/

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
3       pragma solidity ^0.8.19;
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L3:3

G017 - Use assembly to emit events:

````
We can use assembly to emit events efficiently by utilizing `scratch space` and the `fre
Note: In order to do this optimization safely, we will need to cache and restore the fre

For example, for a generic `emit` event for `eventSentAmountExample`:
```solidity
// uint256 id, uint256 value, uint256 amount
emit eventSentAmountExample(id, value, amount);
```

We can use the following assembly emit events:

```solidity
        assembly {
            let memptr := mload(0x40)
            mstore(0x00, calldataload(0x44))
            mstore(0x20, calldataload(0xa4))
            mstore(0x40, amount)
            log1(
                0x00,
                0x60,
                // keccak256("eventSentAmountExample(uint256,uint256,uint256)")
                0xa622cf392588fbf2cd020ff96b2f4ebd9c76d7a4bc7f3e6b2f18012312e76bc3
            )
            mstore(0x40, memptr)
        }
```
````

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
171             emit Transfer(address(0x0000000000000000000000000000000000000000), _msgSende
```

```
242             emit Approval(owner, spender, amount);
```

```
325             emit Transfer(sender, recipient, tTransferAmount);
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L325:325

G018 - Don't use `_msgSender()` if not supporting EIP-2771:

```
Use `msg.sender` if the code does not implement [EIP-2771 trusted forwarder](https://eip
```

Click to show 9 findings

`File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol`

```
7           function _msgSender() internal view virtual returns (address) {
8               return msg.sender;
9           }
```

```
71              address msgSender = _msgSender();
```

```
80              require(_owner == _msgSender(), "Caller is not the owner");
```

```
159             _rOwned[_msgSender()] = _rTotal;
```

```
171             emit Transfer(address(0x0000000000000000000000000000000000000000), _msgSende
```

```
195             _transfer(_msgSender(), recipient, amount);
```

```
204             _approve(_msgSender(), spender, amount);
```

```
228             _approve(sender, _msgSender(), _allowances[sender][_msgSender()].sub(amount,
```

```
228             _approve(sender, _msgSender(), _allowances[sender][_msgSender()].sub(amount,
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L228:228

G019 - Use `uint256(1)/uint256(2)` instead for `true` and `false` boolean states:

If you don't use boolean for storage you will avoid Gwarmaccess 100 gas. In addition, state changes of boolean from `true` to `false` can cost up to ~20000 gas rather than `uint256(2)` to `uint256(1)` that would cost significantly less.

Click to show 4 findings

`File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol`

```
119         mapping (address => bool) private _isExcludedFromFee;
```

```
120         mapping (address => bool) private _rewardAddress;
```

```
150         bool private inSwap = false;
```

```
151         bool private swapEnabled = true;
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L151:151

G020 - Usage of `uints/ints` smaller than 32 bytes (256 bits) incurs overhead:

When using elements that are smaller than 32 bytes, your contract's gas usage may be higher. This is because the EVM operates on 32 bytes at a time. Therefore, if the element is smaller than that, the EVM must use more operations in order to reduce the size of the element from 32 bytes to the desired size.https://docs.soliditylang.org/en/v0.8.11/internals/layout_in_storage.htmlEach operation involving a `uint8` costs an extra **22-28 gas** (depending on whether the other operand is also a variable of type `uint8`) as compared to ones involving `uint256`, due to the compiler having to

clear the higher bits of the memory word before operating on the `uint8`, as well as the associated stack operations of doing so. Use a larger size then downcast where needed.

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
145         uint8 private constant _decimals = 9;
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L145:145

G021 - Consider activating `via-ir` for deploying:

```
The IR-based code generator was introduced with an aim to not only allow code generation

You can enable it on the command line using `--via-ir` or with the option `{"viaIR": tru

This will take longer to compile, but you can just simple test it before deploying and i

More on: https://docs.soliditylang.org/en/v0.8.17/ir-breaking-changes.html
```

File: Various Files

None

G022 - `unchecked {}` can be used on the division of two `uints` in order to save gas:

The division cannot overflow, since both the numerator and the denominator are non-negative.

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
50              require(c / a == b, "Multiplication overflow");
```

```
60              uint256 c = a / b;
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L60:60

G023 - Private functions used once can be inlined:

Private functions used once can be inlined to save GAS

Click to show 11 findings

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
232         function tokenFromReflection(uint256 rAmount) private view returns(uint256) {
233             require(rAmount <= _rTotal, "Amount must be less than total reflections");
234             uint256 currentRate =  _getRate();
235             return rAmount.div(currentRate);
236         }


294         function swapTokensForEth(uint256 tokenAmount) private lockTheSwap {
295             address[] memory path = new address[](2);
296             path[0] = address(this);
297             path[1] = uniswapV2Router.WETH();
298             _approve(address(this), address(uniswapV2Router), tokenAmount);
299             uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
300                 tokenAmount,
301                 0,
302                 path,
303                 address(this),
304                 block.timestamp
305             );
306         }


308         function sendETHToFee(uint256 amount) private {
309             uint256 dAmount = amount.mul(_devPer).div(100);
310             uint256 mAmount = amount.sub(dAmount);
311             _twddev.transfer(dAmount);
312             _twdmkt.transfer(mAmount);
313         }


315         function _tokenTransfer(address sender, address recipient, uint256 amount) priva
316             _transferStandard(sender, recipient, amount);
317         }


319         function _transferStandard(address sender, address recipient, uint256 tAmount) p
320             (uint256 rAmount, uint256 rTransferAmount, uint256 rFee, uint256 tTransferAm
321             _rOwned[sender] = _rOwned[sender].sub(rAmount);
```

```
322             _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount);
323             _takeTeam(tTeam);
324             _reflectFee(rFee, tFee);
325             emit Transfer(sender, recipient, tTransferAmount);
326         }


328         function _takeTeam(uint256 tTeam) private {
329             uint256 currentRate =  _getRate();
330             uint256 rTeam = tTeam.mul(currentRate);
331             _rOwned[address(this)] = _rOwned[address(this)].add(rTeam);
332         }


334         function _reflectFee(uint256 rFee, uint256 tFee) private {
335             _rTotal = _rTotal.sub(rFee);
336             _tFeeTotal = _tFeeTotal.add(tFee);
337         }


341         function _getValues(uint256 tAmount) private view returns (uint256, uint256, uin
342             (uint256 tTransferAmount, uint256 tFee, uint256 tTeam) = _getTValues(tAmount
343             uint256 currentRate =  _getRate();
344             (uint256 rAmount, uint256 rTransferAmount, uint256 rFee) = _getRValues(tAmou
345             return (rAmount, rTransferAmount, rFee, tTransferAmount, tFee, tTeam);
346         }


348         function _getTValues(uint256 tAmount, uint256 taxFee, uint256 teamFee) private p
349             uint256 tFee = tAmount.mul(taxFee).div(100);
350             uint256 tTeam = tAmount.mul(teamFee).div(100);
351             uint256 tTransferAmount = tAmount.sub(tFee).sub(tTeam);
352             return (tTransferAmount, tFee, tTeam);
353         }


355         function _getRValues(uint256 tAmount, uint256 tFee, uint256 tTeam, uint256 curre
356             uint256 rAmount = tAmount.mul(currentRate);
357             uint256 rFee = tFee.mul(currentRate);
358             uint256 rTeam = tTeam.mul(currentRate);
359             uint256 rTransferAmount = rAmount.sub(rFee).sub(rTeam);
360             return (rAmount, rTransferAmount, rFee);
361         }


368         function _getCurrentSupply() private view returns(uint256, uint256) {
```

```
369             uint256 rSupply = _rTotal;
370             uint256 tSupply = _tTotal;
371             if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
372             return (rSupply, tSupply);
373         }
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L368:373

G024 - Unused named return variables without optimizer waste gas:

Consider changing the variable to be an unnamed one, since the variable is never assigned, nor is it returned by name. If the optimizer is not turned on, leaving the code as it is will also waste gas for the stack variable.

Click to show 27 findings

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
    function _msgSender() internal view virtual returns (address) {

    function totalSupply() external view returns (uint256);

    function balanceOf(address account) external view returns (uint256);

    function transfer(address recipient, uint256 amount) external returns (bool);

    function allowance(address owner, address spender) external view returns (uint256);

    function approve(address spender, uint256 amount) external returns (bool);

    function transferFrom(address sender, address recipient, uint256 amount) external returr

    function transferFrom(address, address, uint) external returns (bool);

    function transfer(address, uint) external returns (bool);

    function sub(uint256 a, uint256 b) internal pure returns (uint256) {

    function div(uint256 a, uint256 b) internal pure returns (uint256) {

    function owner() public view returns (address) {

    function createPair(address tokenA, address tokenB) external returns (address pair);
```

```
    function factory() external pure returns (address);

    function WETH() external pure returns (address);

    ) external payable returns (uint amountToken, uint amountETH, uint liquidity);

    ) external payable returns (uint amountToken, uint amountETH, uint liquidity);

    ) external payable returns (uint amountToken, uint amountETH, uint liquidity);

    function name() public pure returns (string memory) {

    function symbol() public pure returns (string memory) {

    function decimals() public pure returns (uint8) {

    function totalSupply() public pure override returns (uint256) {

    function balanceOf(address account) public view override returns (uint256) {

    function transfer(address recipient, uint256 amount) public override returns (bool) {

    function allowance(address owner, address spender) public view override returns (uint256

    function approve(address spender, uint256 amount) public override returns (bool) {

    function transferFrom(address sender, address recipient, uint256 amount) public override
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L226:226

G025 - Avoid updating storage when the value hasn't changed:

If the old value is equal to the new value, not re-storing the value will avoid a Gsreset (**2900 gas**), potentially at the expense of a Gcoldsload (**2100 gas**) or a Gwarmaccess (**100 gas**).

Click to show 6 findings

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
216         function setBuyFees(uint256 newBuyFee, uint256 newRedisBuyFee) public onlyOwner
```

```
84          function renounceOwnership() public virtual onlyOwner {
```

```
334         function _reflectFee(uint256 rFee, uint256 tFee) private {
```

```
245         function _transfer(address from, address to, uint256 amount) private {
```

```
221         function setSellFees(uint256 newSellFee, uint256 newRedisSellFee) public onlyOwn
```

```
212         function setMaxWalletPer(uint256 newMaxWalletPer) public onlyOwner {
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L212:212

G026 - Do not calculate constants:

Due to how constant variables are implemented (replacements at compile-time), an expression assigned to a constant variable is recomputed each time that the variable is used, which wastes some gas.

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
123         uint256 private constant _tTotal = 1000000000 * 10**9;
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L123:123

G027 - The use of a logical AND in place of double if is slightly less gas efficient in instances where there isn't a corresponding else statement for the given if statement:

Using a double if statement instead of logical AND (&&) can provide similar short-circuiting behavior whereas double if is slightly more efficient.

Click to show 6 findings

File: tmp/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol

```
249             if (to != deadAddress &&
250                 to != address(this) &&
251                 to != uniswapV2Pair &&
```

```
252                 !_isExcludedFromFee[from] &&
253                 !_isExcludedFromFee[to]
254             ) {
255                 uint256 heldTokens = balanceOf(to);
256                 uint256 maxWalletTokens = _maxWalletPer.mul(_tTotal).div(100);
257                 require((heldTokens + amount) <= maxWalletTokens, "Over wallet limit.");
258             }


263             if (from != owner() && to != owner()) {
264
265                 uint256 contractTokenBalance = balanceOf(address(this));
266                 if (!inSwap && from != uniswapV2Pair && swapEnabled && contractTokenBala
267                     swapTokensForEth(contractTokenBalance);
268                     uint256 contractETHBalance = address(this).balance;
269                     if(contractETHBalance > 0) {
270                         sendETHToFee(address(this).balance);
271                     }
272                 }
273
274                 if(from == uniswapV2Pair && to != address(uniswapV2Router)) {
275                     _redisFee = _redisFeeOnBuy;
276                     _taxFee = _taxFeeOnBuy;
277                 }
278
279                 if (to == uniswapV2Pair && from != address(uniswapV2Router)) {
280                     _redisFee = _redisFeeOnSell;
281                     _taxFee = _taxFeeOnSell;
282                 }
283
284                 if ((_isExcludedFromFee[from] || _isExcludedFromFee[to]) || (from != uni
285                     _redisFee = 0;
286                     _taxFee = 0;
287                 }
288
289             }


266                 if (!inSwap && from != uniswapV2Pair && swapEnabled && contractTokenBala
267                     swapTokensForEth(contractTokenBalance);
268                     uint256 contractETHBalance = address(this).balance;
269                     if(contractETHBalance > 0) {
270                         sendETHToFee(address(this).balance);
271                     }
272                 }
```

```
274                 if(from == uniswapV2Pair && to != address(uniswapV2Router)) {
275                     _redisFee = _redisFeeOnBuy;
276                     _taxFee = _taxFeeOnBuy;
277                 }


279                 if (to == uniswapV2Pair && from != address(uniswapV2Router)) {
280                     _redisFee = _redisFeeOnSell;
281                     _taxFee = _taxFeeOnSell;
282                 }


284                 if ((_isExcludedFromFee[from] || _isExcludedFromFee[to]) || (from != uni
285                     _redisFee = 0;
286                     _taxFee = 0;
287                 }
```

0x4473996394e1Da0c6E7a79dc320084328920040A on bsc (mainnet)/tree/main/92026900-78eb-4676-a19f-8506daabc4bd/contract.sol#L284:287