BLOCKHAT SECURITY

Anu Initiative

Smart Contract Security Audit

Prepared by BlockHat

April 15th, 2023 – April 20th, 2023

BlockHat.io

contact@blockhat.io

Document Properties

Client	Anu Initiative CLG
Version	0.1
Classification	Public

Scope

The Anu Initiative Contract in the Anu Initiative Repository

Link	Address
`https://testnet.bscscan.com/token/0x6b3e4d70005d210e29a74283ab69ed52c6b5810b#code`	0x6b3e4d70005D210e29a74283aB69ed52c6b5810b

Files	MD5 Hash
/ReflectionTokenWithAntibot.sol	c98dc02881fde3ff55d5179ef3e937c4

Contacts

COMPANY	CONTACT
BlockHat	contact@blockhat.io

Contents

1 Introduction

Anu Initiative engaged BlockHat to conduct a security assessment on the Anu Initiative beginning on April 15th, 2023 and ending April 20th, 2023. In this report, we detail our methodical approach to evaluate potential security issues associated with the implementation of smart contracts, by exposing possible semantic discrepancies between the smart contract code and design document, and by recommending additional ideas to optimize the existing code. Our findings indicate that the current version of smart contracts can still be enhanced further due to the presence of many security and performance concerns.

This document summarizes the findings of our audit.

1.1 About Anu Initiative

Anu Initiative is a Company Limited by Guarantee (Not-For-Profit) incorporated in Dublin, Ireland. It was founded by nature lovers in an effort to harness the powers of blockchain technologies to create a positive impact on the planet, and to help improve the Web3 ecosystem. The Anu Initiative aims to help heal the damage that actions like oil spills or deforestation have on our environment and wildlife as well as offset the negative impact cryptocurrencies may have on the environment by providing resources to nature-focused charities around the world.

Issuer	Anu Initiative CLG
Website	`https://anuinitiative.org/`
Type	Solidity Smart Contract
Audit Method	Whitebox

1.2 Approach & Methodology

BlockHat used a combination of manual and automated security testing to achieve a balance between efficiency, timeliness, practicability, and correctness within the audit's scope. While manual testing is advised for identifying problems in logic, procedure, and implementation, automated testing techniques help to expand the coverage of smart contracts and can quickly detect code that does not comply with security best practices.

1.2.1 Risk Methodology

Vulnerabilities or bugs identified by BlockHat are ranked using a risk assessment technique that considers both the LIKELIHOOD and IMPACT of a security incident. This framework is effective at conveying the features and consequences of technological vulnerabilities.

Its quantitative paradigm enables repeatable and precise measurement, while also revealing the underlying susceptibility characteristics that were used to calculate the Risk scores. A risk level will be assigned to each vulnerability on a scale of 5 to 1, with 5 indicating the greatest possibility or impact.

- Likelihood quantifies the probability of a certain vulnerability being discovered and exploited in the untamed.
- Impact quantifies the technical and economic costs of a successful attack.
- Severity indicates the risk's overall criticality.

Probability and impact are classified into three categories: H, M, and L, which correspond to high, medium, and low, respectively. Severity is determined by probability and impact and is categorized into four levels, namely Critical, High, Medium, and Low.

Impact				
	High	Critical	High	Medium
	Medium	High	Medium	Low
	Low	Medium	Low	Low
		High	Medium	Low

Likelihood

2 Findings Overview

2.1 Summary

The following is a synopsis of our conclusions from our analysis of the Anu Initiative implementation. During the first part of our audit, we examine the smart contract source code and run the codebase via a static code analyzer. The objective here is to find known coding problems statically and then manually check (reject or confirm) issues highlighted by the tool. Additionally, we check business logics, system processes, and DeFi-related components manually to identify potential hazards and/or defects.

2.2 Key Findings

In general, these smart contracts are well-designed and constructed, but their implementation might be improved by addressing the discovered flaws, which include , 3 medium-severity, 4 low-severity vulnerabilities.

Vulnerabilities	Severity	Status
Setting Maximum Wallet Size Too Low in Smart Contracts	MEDIUM	Not fixed
Changing Uniswap Router Address Mid-Swap in Smart Contract	MEDIUM	Not fixed
Change of BaseTokenForPair while swapping	MEDIUM	Not fixed
Unlimited Max Transaction Amount	LOW	Not fixed
Excluding Contract Owners from Max Transaction Amounts	LOW	Not fixed
Missing address verification	LOW	Not fixed
New Marketing Wallet Address in Update Function	LOW	Not fixed

3 Finding Details

A ReflectionTokenWithAntibot.sol

A.1 Setting Maximum Wallet Size Too Low in Smart Contracts [MEDIUM]

Description:

In the updateMaxWallet() function the require() statement checks whether _maxWallet is greater than zero. However, this may not be sufficient to ensure that the maximum wallet size is set to a reasonable value that allows legitimate users to participate in the system.

A potential issue with this implementation is that it does not prevent the maximum wallet size from being set too low. If the maxWallet value is initially set to a very low value, the require() statement in the updateMaxWallet() function would not catch this.

Code:

Listing 1: ReflectionTokenWithAntibot.sol

```
    function updateMaxWallet(uint256 _maxWallet) external onlyOwner {
        require(_maxWallet>0, "maxWallet > 0");
        emit UpdateMaxWallet(_maxWallet, maxWallet);
        maxWallet = _maxWallet;
    }
```

Risk Level:

Likelihood – 3
Impact - 2

Recommendation:

To prevent the maximum wallet size from being set too low, the require() statement in the updateMaxWallet() function should be modified to check that _maxWallet is greater than the current maxWallet value, rather than just greater than zero. Additionally, the value selected for the maximum wallet size should be carefully considered to ensure that it allows legitimate users to participate in the system while also reducing the risk of market manipulation.

Status - Not fixed

A.2 Changing Uniswap Router Address Mid-Swap in Smart Contract [MEDIUM]

Description:

It is important to use a fixed Uniswap router address in a smart contract that involves token swaps because the router is a critical component of the Uniswap protocol. The router is responsible for executing trades and calculating the correct exchange rate between tokens.

If the router address is changed during a swap, it could cause unexpected behavior or even result in loss of funds. For example, if a user starts a swap with one router address and the router address is changed midway through the swap, the trade may fail or the user may receive an incorrect amount of tokens.

Code:

Listing 2: ReflectionTokenWithAntibot.sol

```
    function updateUniswapV2Router(address newAddress) public onlyOwner
        ↪ {
        require(
            newAddress != address(mainRouter),
            "The router already has that address"
        );
        emit UpdateUniswapV2Router(newAddress, address(mainRouter));
        mainRouter = IUniswapV2Router02(newAddress);
        address _mainPair = IUniswapV2Factory(mainRouter.factory())
            .createPair(address(this), baseTokenForPair);
        mainPair = _mainPair;
        }
```

Risk Level:

Likelihood – 3
Impact - 2

Recommendation:

To prevent these issues, it is recommended to use a fixed Uniswap router address in a smart contract. This address should be declared as a constant variable in the contract, so that it cannot be modified at runtime. This will ensure that all swaps are executed using the correct router and that the exchange rate is calculated correctly. If there is a need to update the router address, it is important to do so carefully to minimize the risk of issues during a swap. It is recommended to only update the router address when the swap functionality is paused, and to inform users of the upcoming change to the router address to prevent them from initiating swaps during the update process. To further reduce the risk of issues during the update process, it is recommended to add a require statement in the updateUniswapV2Router() function to check if the swap functionality is paused before allowing the router address to be updated.

Status - Not fixed

A.3 Change of BaseTokenForPair while swapping [MEDIUM]

Description:

Changing the baseTokenForPair address during the swap process without notifying users beforehand can create unexpected behavior and pose a risk to users who are currently swapping with the token. This can lead to a loss of trust in the token and potentially harm the reputation of the token and its developers. Therefore, it's crucial to provide clear communication to token holders before making any changes to the token's core functionality or trading mechanism.

Code:

Listing 3: ReflectionTokenWithAntibot.sol

```
function updateUniswapV2Pair(address _baseTokenForPair) external
{
    require(
        _baseTokenForPair != baseTokenForPair,
        "The baseTokenForPair already has that address"
    );
    baseTokenForPair=_baseTokenForPair;
    mainPair = IUniswapV2Factory(mainRouter.factory()).createPair(
        address(this),
        baseTokenForPair
    );
}
```

Risk Level:

Likelihood – 3
Impact - 2

Recommendation:

Use a fixed Uniswap pair address in a smart contract and declare this address as a constant variableor update the pair address only when the swap functionality is paused and after informing users of the upcoming change. To reduce the risk of issues during the update process, add a require statement in the updateUniswapV2Pair() function to check if the swap functionality is paused. It's also essential to communicate the reasons for the change, the expected impact on liquidity and trading volume, and any actions users need to take to ensure a smooth transition. In general, maintaining transparency and communicating changes to the community beforehand is essential to maintain trust and build a loyal user base.

Status - Not fixed

A.4 Unlimited Max Transaction Amount [LOW]

Description:

The current implementation of the smart contract does not set a limit on the maximum transaction amount, which could potentially lead to market manipulation and harm to other token holders. The max transaction amount is currently set by the contract owner during deployment, which could result in an unlimited maximum transaction amount.

Code:

Listing 4: ReflectionTokenWithAntibot.sol

```
        require(_maxTransactionAmount>0, "maxTransactionAmount > 0");
```

Listing 5: ReflectionTokenWithAntibot.sol

```
            function updateMaxTransactionAmount(uint256
                ↪ _maxTransactionAmount)
        external
        onlyOwner
    {
        require(_maxTransactionAmount>0, "maxTransactionAmount > 0");
        emit UpdateMaxTransactionAmount(_maxTransactionAmount,
            ↪ maxTransactionAmount);
        maxTransactionAmount = _maxTransactionAmount;
    }
```

Risk Level:

Likelihood – 2
Impact - 2

Recommendation:

To mitigate the risk of market manipulation and to ensure fair trading of the token, it is recommended to set a limit on the maximum transaction amount. This limit can be calculated as a percentage of the total supply of the token, and should be included in the contract's code. Additionally, it is recommended to set a reasonable limit for the maximum transaction amount during contract deployment, and to consider implementing a mechanism for adjusting this limit in the future if necessary.

A commonly used limit for the maximum transaction amount is 1% of the total supply of the token. This limit is often considered reasonable because it allows for small-to-medium-sized transactions while preventing large-scale market manipulation.

Status - Not fixed

A.5 Excluding Contract Owners from Max Transaction Amounts [LOW]

Description:

Excluding the owner from the max transaction amount in a token contract can potentially allow them to perform transactions that exceed the set limit. This could lead to market manipulation and harm other token holders. Large transactions by the owner could significantly impact the token price, making it vulnerable to malicious activities such as insider trading and pump and dump schemes.

For instance, suppose the maximum transaction amount for the token is set to 1% of the total supply. In that case, if the contract owner is excluded from this limit, they could perform a transaction that represents a significant portion of the total supply, which could cause the token price to fluctuate, affecting other token holders' investments.

To prevent these risks, it's recommended to include the contract owner in the max transaction amount limit. This will prevent centralization and reduce the potential for market manipulation. Additionally, it's essential to ensure that token contract developers provide clear communication to the community regarding any changes to the contract and its functionality to maintain transparency and build trust among the user base.

Code:

Listing 6: ReflectionTokenWithAntibot.sol

```
        isExcludedFromMaxTransactionAmount[_msgSender()]=true;
```

Risk Level:

Likelihood – 2
Impact - 2

Recommendation:

To prevent market manipulation and promote transparency, token contract developers should include the contract owner in the max transaction amount limit. This will prevent the owner from performing transactions that significantly impact the token price and harm other token holders. To maintain transparency and build trust.

Status - Not fixed

A.6 Missing address verification [LOW]

Description:

Certain functions lack a safety check in the address, the address-type argument changeWhitelist , changeBlocklist and setNewOwner function should include a zero-address test for the address _user and address _newOwner

Code:

Listing 7: ReflectionTokenWithAntibot.sol

```
    function excludeFromReward(address account) public onlyOwner {
        require(!_isExcluded[account], "Account is already excluded");
        require(
            _excluded.length + 1 <= 50,
            "Cannot exclude more than 50 accounts. Include a previously
                ↪ excluded address."
        );
        if (_rOwned[account] > 0) {
            _tOwned[account] = tokenFromReflection(_rOwned[account]);
        }
        _isExcluded[account] = true;
        _excluded.push(account);
    }
```

Listing 8: ReflectionTokenWithAntibot.sol

```
    function excludeFromFee(address account, bool isEx) external
        ↪ onlyOwner {
        require(isExcludedFromFee[account] != isEx, "already");
        isExcludedFromFee[account] = isEx;
        emit ExcludedFromFee(account, isEx);
    }
```

Listing 9: ReflectionTokenWithAntibot.sol

```
    function includeInReward(address account) public onlyOwner {
        require(_isExcluded[account], "Account is not excluded");
        for (uint256 i = 0; i < _excluded.length; i++) {
            if (_excluded[i] == account) {
                uint256 prev_rOwned=_rOwned[account];
                _rOwned[account]=_tOwned[account]*_getRate();
                _rTotal=_rTotal+_rOwned[account]-prev_rOwned;
                _isExcluded[account] = false;
                _excluded[i] = _excluded[_excluded.length - 1];
                _excluded.pop();
                break;
            }
        }
    }
```

Risk Level:

Likelihood – 1
Impact - 2

Recommendation:

It is recommended to verify that the address provided in the arguments is different from the address(0) .

Status - Not fixed

A.7 New Marketing Wallet Address in Update Function [LOW]

Description:

The updateMarketingWalletfunction allows the contract owner to update the marketing wallet address, but it lacks a validation check to ensure the new address is not the same as the old one.

Code:

Listing 10: ReflectionTokenWithAntibot.sol

```
    function updateMarketingWallet(
        address _marketingWallet,
        bool _isMarketingFeeBaseToken
    ) external onlyOwner {
        require(_marketingWallet != address(0), "marketing wallet can't
            ↪ be 0");
        emit UpdateMarketingWallet(_marketingWallet,
            ↪ _isMarketingFeeBaseToken,
            marketingWallet, isMarketingFeeBaseToken);
        marketingWallet = _marketingWallet;
        isMarketingFeeBaseToken = _isMarketingFeeBaseToken;
        isExcludedFromFee[_marketingWallet] = true;
        }
```

Risk Level:

Likelihood – 1
Impact - 2

Recommendation:

Developers should add a validation check to the function to compare the new address with the current marketing wallet address. This will ensure that the function executes correctly and prevent duplicate transactions.

Status - Not fixed

4 Static Analysis (Slither)

Description:

Block Hat expanded the coverage of the specific contract areas using automated testing methodologies. Slither, a Solidity static analysis framework, was one of the tools used. Slither was run on all-scoped contracts in both text and binary formats. This tool can be used to test mathematical relationships between Solidity instances statically and variables that allow for the detection of errors or inconsistent usage of the contracts' APIs throughout the entire codebase.

Results:

```
ReflectionTokenWithAntibot.takeFee() (ReflectionTokenWithAntibot.sol
    ↪ #1227-1279) sends eth to arbitrary user
        Dangerous calls:
        - (success) = address(marketingWallet).call{value:
            ↪ baseTokenForMarketing}() (ReflectionTokenWithAntibot.sol
            ↪ #1249)
ReflectionTokenWithAntibot.addLiquidity(uint256,uint256) (
    ↪ ReflectionTokenWithAntibot.sol#1306-1329) sends eth to arbitrary
    ↪ user
        Dangerous calls:
        - mainRouter.addLiquidityETH{value: baseTokenAmount}(address(this
            ↪ ),tokenAmount,0,0,address(0xdead),block.timestamp) (
            ↪ ReflectionTokenWithAntibot.sol#1310-1317)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #functions-that-send-ether-to-arbitrary-destinations

Reentrancy in ReflectionTokenWithAntibot._transfer(address,address,
    ↪ uint256) (ReflectionTokenWithAntibot.sol#1164-1225):
        External calls:
        - IGemAntiBot(gemAntiBot).onPreTransferCheck(from,to,amount) (
            ↪ ReflectionTokenWithAntibot.sol#1172)
        - takeFee() (ReflectionTokenWithAntibot.sol#1186)
               - returndata = address(token).functionCall(data,SafeERC20:
                  ↪  low-level call failed) (node_modules/@openzeppelin
                  ↪ /contracts/token/ERC20/utils/SafeERC20.sol#110)
               - IERC20(baseTokenForPair).approve(address(mainRouter),
                  ↪ baseTokenAmount) (ReflectionTokenWithAntibot.sol
                  ↪ #1308)
               - mainRouter.addLiquidityETH{value: baseTokenAmount}(
                  ↪ address(this),tokenAmount,0,0,address(0xdead),block
                  ↪ .timestamp) (ReflectionTokenWithAntibot.sol
                  ↪ #1310-1317)
               - IGemAntiBot(gemAntiBot).onPreTransferCheck(from,to,
                  ↪ amount) (ReflectionTokenWithAntibot.sol#1172)
               - (success,returndata) = target.call{value: value}(data) (
                  ↪ node_modules/@openzeppelin/contracts/utils/Address.
                  ↪ sol#137)
               - mainRouter.addLiquidity(address(this),baseTokenForPair,
                  ↪ tokenAmount,baseTokenAmount,0,0,address(0xdead),
                  ↪ block.timestamp) (ReflectionTokenWithAntibot.sol
                  ↪ #1319-1328)
               - mainRouter.
                  ↪ swapExactTokensForETHSupportingFeeOnTransferTokens(
                  ↪ tokenAmount,0,path,address(this),block.timestamp) (
                  ↪ ReflectionTokenWithAntibot.sol#1287-1293)
               - uniswapV2Caller.
                  ↪ swapExactTokensForTokensSupportingFeeOnTransferTokens
                  ↪ (address(mainRouter),tokenAmount,0,path,block.
                  ↪ timestamp) (ReflectionTokenWithAntibot.sol
                  ↪ #1296-1302)
               - (success) = address(marketingWallet).call{value:
                  ↪ baseTokenForMarketing}() (
                  ↪ ReflectionTokenWithAntibot.sol#1249)
               - IERC20(baseTokenForPair).safeTransfer(marketingWallet,
                  ↪ baseTokenForMarketing) (ReflectionTokenWithAntibot.
```

```
            ↪ sol#1255)
    External calls sending eth:
    - takeFee() (ReflectionTokenWithAntibot.sol#1186)
            - mainRouter.addLiquidityETH{value: baseTokenAmount}(
               ↪ address(this),tokenAmount,0,0,address(0xdead),block
               ↪ .timestamp) (ReflectionTokenWithAntibot.sol
               ↪ #1310-1317)
            - (success,returndata) = target.call{value: value}(data) (
               ↪ node_modules/@openzeppelin/contracts/utils/Address.
               ↪ sol#137)
            - (success) = address(marketingWallet).call{value:
               ↪ baseTokenForMarketing}() (
               ↪ ReflectionTokenWithAntibot.sol#1249)
    State variables written after the call(s):
    - removeAllFee() (ReflectionTokenWithAntibot.sol#1188)
            - _liquidityFee = 0 (ReflectionTokenWithAntibot.sol#819)
    - _liquidityFee = buyLiquidityFee (ReflectionTokenWithAntibot.sol
       ↪ #1199)
    - _liquidityFee = sellLiquidityFee (ReflectionTokenWithAntibot.
       ↪ sol#1205)
    - restoreAllFee() (ReflectionTokenWithAntibot.sol#1210)
            - _liquidityFee = _previousLiquidityFee (
               ↪ ReflectionTokenWithAntibot.sol#824)
    - _tokenTransfer(from,to,amount) (ReflectionTokenWithAntibot.sol
       ↪ #1209)
            - _liquidityFeeTokens = _liquidityFeeTokens + (tLiquidity)
               ↪  (ReflectionTokenWithAntibot.sol#853)
    - removeAllFee() (ReflectionTokenWithAntibot.sol#1188)
            - _marketingFee = 0 (ReflectionTokenWithAntibot.sol#817)
    - _marketingFee = buyMarketingFee (ReflectionTokenWithAntibot.sol
       ↪ #1200)
    - _marketingFee = sellMarketingFee (ReflectionTokenWithAntibot.
       ↪ sol#1206)
    - restoreAllFee() (ReflectionTokenWithAntibot.sol#1210)
```

```
            - _marketingFee = _previousMarketingFee (
               ↪ ReflectionTokenWithAntibot.sol#825)
      - _tokenTransfer(from,to,amount) (ReflectionTokenWithAntibot.sol
         ↪ #1209)
            - _marketingFeeTokens = _marketingFeeTokens + (tMarketing)
               ↪  (ReflectionTokenWithAntibot.sol#854)
      - removeAllFee() (ReflectionTokenWithAntibot.sol#1188)
            - _previousLiquidityFee = _liquidityFee (
               ↪ ReflectionTokenWithAntibot.sol#814)
      - removeAllFee() (ReflectionTokenWithAntibot.sol#1188)
            - _previousMarketingFee = _marketingFee (
               ↪ ReflectionTokenWithAntibot.sol#815)
      - removeAllFee() (ReflectionTokenWithAntibot.sol#1188)
            - _previousRewardFee = _rewardFee (
               ↪ ReflectionTokenWithAntibot.sol#813)
      - _tokenTransfer(from,to,amount) (ReflectionTokenWithAntibot.sol
         ↪ #1209)
            - _rOwned[address(this)] = _rOwned[address(this)] + (
               ↪ rLiquidity) + (rMarketing) (
               ↪ ReflectionTokenWithAntibot.sol#858-860)
            - _rOwned[sender] = _rOwned[sender] - (rAmount) (
               ↪ ReflectionTokenWithAntibot.sol#630)
            - _rOwned[sender] = _rOwned[sender] - (rAmount) (
               ↪ ReflectionTokenWithAntibot.sol#651)
            - _rOwned[sender] = _rOwned[sender] - (rAmount) (
               ↪ ReflectionTokenWithAntibot.sol#696)
            - _rOwned[recipient] = _rOwned[recipient] + (
               ↪ rTransferAmount) (ReflectionTokenWithAntibot.sol
               ↪ #631)
            - _rOwned[sender] = _rOwned[sender] - (rAmount) (
               ↪ ReflectionTokenWithAntibot.sol#674)
            - _rOwned[recipient] = _rOwned[recipient] + (
               ↪ rTransferAmount) (ReflectionTokenWithAntibot.sol
               ↪ #653)
```

```
        - _rOwned[recipient] = _rOwned[recipient] + (
            ↪ rTransferAmount) (ReflectionTokenWithAntibot.sol
            ↪ #675)
        - _rOwned[recipient] = _rOwned[recipient] + (
            ↪ rTransferAmount) (ReflectionTokenWithAntibot.sol
            ↪ #698)
- _tokenTransfer(from,to,amount) (ReflectionTokenWithAntibot.sol
    ↪ #1209)
        - _rTotal = _rTotal - (rFee) (ReflectionTokenWithAntibot.
            ↪ sol#705)
- removeAllFee() (ReflectionTokenWithAntibot.sol#1188)
        - _rewardFee = 0 (ReflectionTokenWithAntibot.sol#818)
- _rewardFee = buyRewardFee (ReflectionTokenWithAntibot.sol#1198)
- _rewardFee = sellRewardFee (ReflectionTokenWithAntibot.sol
    ↪ #1204)
- restoreAllFee() (ReflectionTokenWithAntibot.sol#1210)
        - _rewardFee = _previousRewardFee (
            ↪ ReflectionTokenWithAntibot.sol#823)
- _tokenTransfer(from,to,amount) (ReflectionTokenWithAntibot.sol
    ↪ #1209)
        - _tFeeTotal = _tFeeTotal + (tFee) (
            ↪ ReflectionTokenWithAntibot.sol#706)
- _tokenTransfer(from,to,amount) (ReflectionTokenWithAntibot.sol
    ↪ #1209)
        - _tOwned[address(this)] = _tOwned[address(this)] + (
            ↪ tLiquidity) + (tMarketing) (
            ↪ ReflectionTokenWithAntibot.sol#862-864)
        - _tOwned[sender] = _tOwned[sender] - (tAmount) (
            ↪ ReflectionTokenWithAntibot.sol#695)
        - _tOwned[sender] = _tOwned[sender] - (tAmount) (
            ↪ ReflectionTokenWithAntibot.sol#673)
        - _tOwned[recipient] = _tOwned[recipient] + (
            ↪ tTransferAmount) (ReflectionTokenWithAntibot.sol
            ↪ #652)
```

```
                - _tOwned[recipient] = _tOwned[recipient] + (
                    ↪ tTransferAmount) (ReflectionTokenWithAntibot.sol
                    ↪ #697)
Reentrancy in ReflectionTokenWithAntibot.takeFee() (
    ↪ ReflectionTokenWithAntibot.sol#1227-1279):
        External calls:
        - swapTokensForBaseToken(tokensForSwap) (
            ↪ ReflectionTokenWithAntibot.sol#1242)
                - mainRouter.
                    ↪ swapExactTokensForETHSupportingFeeOnTransferTokens(
                    ↪ tokenAmount,0,path,address(this),block.timestamp) (
                    ↪ ReflectionTokenWithAntibot.sol#1287-1293)
                - uniswapV2Caller.
                    ↪ swapExactTokensForTokensSupportingFeeOnTransferTokens
                    ↪ (address(mainRouter),tokenAmount,0,path,block.
                    ↪ timestamp) (ReflectionTokenWithAntibot.sol
                    ↪ #1296-1302)
        - (success) = address(marketingWallet).call{value:
            ↪ baseTokenForMarketing}() (ReflectionTokenWithAntibot.sol
            ↪ #1249)
        External calls sending eth:
        - (success) = address(marketingWallet).call{value:
            ↪ baseTokenForMarketing}() (ReflectionTokenWithAntibot.sol
            ↪ #1249)
        State variables written after the call(s):
        - _marketingFeeTokens = 0 (ReflectionTokenWithAntibot.sol#1251)
Reentrancy in ReflectionTokenWithAntibot.takeFee() (
    ↪ ReflectionTokenWithAntibot.sol#1227-1279):
        External calls:
        - swapTokensForBaseToken(tokensForLiquidity) (
            ↪ ReflectionTokenWithAntibot.sol#1262)
                - mainRouter.
                    ↪ swapExactTokensForETHSupportingFeeOnTransferTokens(
                    ↪ tokenAmount,0,path,address(this),block.timestamp) (
                    ↪ ReflectionTokenWithAntibot.sol#1287-1293)
                - uniswapV2Caller.
                    ↪ swapExactTokensForTokensSupportingFeeOnTransferTokens
                    ↪ (address(mainRouter),tokenAmount,0,path,block.
                    ↪ timestamp) (ReflectionTokenWithAntibot.sol
                    ↪ #1296-1302)
        - _transfer(address(this),marketingWallet,_marketingFeeTokens) (
            ↪ ReflectionTokenWithAntibot.sol#1266)
                - returndata = address(token).functionCall(data,SafeERC20:
                    ↪  low-level call failed) (node_modules/@openzeppelin
                    ↪ /contracts/token/ERC20/utils/SafeERC20.sol#110)
                - IERC20(baseTokenForPair).approve(address(mainRouter),
                    ↪ baseTokenAmount) (ReflectionTokenWithAntibot.sol
                    ↪ #1308)
                - mainRouter.addLiquidityETH{value: baseTokenAmount}(
                    ↪ address(this),tokenAmount,0,0,address(0xdead),block
                    ↪ .timestamp) (ReflectionTokenWithAntibot.sol
                    ↪ #1310-1317)
                - IGemAntiBot(gemAntiBot).onPreTransferCheck(from,to,
                    ↪ amount) (ReflectionTokenWithAntibot.sol#1172)
                - (success,returndata) = target.call{value: value}(data) (
                    ↪ node_modules/@openzeppelin/contracts/utils/Address.
                    ↪ sol#137)
                - mainRouter.addLiquidity(address(this),baseTokenForPair,
                    ↪ tokenAmount,baseTokenAmount,0,0,address(0xdead),
                    ↪ block.timestamp) (ReflectionTokenWithAntibot.sol
                    ↪ #1319-1328)
                - mainRouter.
                    ↪ swapExactTokensForETHSupportingFeeOnTransferTokens(
                    ↪ tokenAmount,0,path,address(this),block.timestamp) (
                    ↪ ReflectionTokenWithAntibot.sol#1287-1293)
                - uniswapV2Caller.
                    ↪ swapExactTokensForTokensSupportingFeeOnTransferTokens
                    ↪ (address(mainRouter),tokenAmount,0,path,block.
                    ↪ timestamp) (ReflectionTokenWithAntibot.sol
                    ↪ #1296-1302)
                - (success) = address(marketingWallet).call{value:
                    ↪ baseTokenForMarketing}() (
                    ↪ ReflectionTokenWithAntibot.sol#1249)
                - IERC20(baseTokenForPair).safeTransfer(marketingWallet,
                    ↪ baseTokenForMarketing) (ReflectionTokenWithAntibot.
                    ↪ sol#1255)
         External calls sending eth:
         - _transfer(address(this),marketingWallet,_marketingFeeTokens) (
             ↪ ReflectionTokenWithAntibot.sol#1266)
                - mainRouter.addLiquidityETH{value: baseTokenAmount}(
                    ↪ address(this),tokenAmount,0,0,address(0xdead),block
                    ↪ .timestamp) (ReflectionTokenWithAntibot.sol
                    ↪ #1310-1317)
                - (success,returndata) = target.call{value: value}(data) (
                    ↪ node_modules/@openzeppelin/contracts/utils/Address.
                    ↪ sol#137)
                - (success) = address(marketingWallet).call{value:
                    ↪ baseTokenForMarketing}() (
                    ↪ ReflectionTokenWithAntibot.sol#1249)
         State variables written after the call(s):
         - _marketingFeeTokens = 0 (ReflectionTokenWithAntibot.sol#1268)
  Reentrancy in ReflectionTokenWithAntibot.takeFee() (
      ↪ ReflectionTokenWithAntibot.sol#1227-1279):
         External calls:
         - swapTokensForBaseToken(tokensForSwap) (
             ↪ ReflectionTokenWithAntibot.sol#1242)
                - mainRouter.
                    ↪ swapExactTokensForETHSupportingFeeOnTransferTokens(
                    ↪ tokenAmount,0,path,address(this),block.timestamp) (
                    ↪ ReflectionTokenWithAntibot.sol#1287-1293)
                - uniswapV2Caller.
                    ↪ swapExactTokensForTokensSupportingFeeOnTransferTokens
                ↪ (address(mainRouter),tokenAmount,0,path,block.
                ↪ timestamp) (ReflectionTokenWithAntibot.sol
                ↪ #1296-1302)
        - (success) = address(marketingWallet).call{value:
            ↪ baseTokenForMarketing}() (ReflectionTokenWithAntibot.sol
            ↪ #1249)
        - IERC20(baseTokenForPair).safeTransfer(marketingWallet,
            ↪ baseTokenForMarketing) (ReflectionTokenWithAntibot.sol
            ↪ #1255)
        - swapTokensForBaseToken(tokensForLiquidity) (
            ↪ ReflectionTokenWithAntibot.sol#1262)
                - mainRouter.
                    ↪ swapExactTokensForETHSupportingFeeOnTransferTokens(
                    ↪ tokenAmount,0,path,address(this),block.timestamp) (
                    ↪ ReflectionTokenWithAntibot.sol#1287-1293)
                - uniswapV2Caller.
                    ↪ swapExactTokensForTokensSupportingFeeOnTransferTokens
                    ↪ (address(mainRouter),tokenAmount,0,path,block.
                    ↪ timestamp) (ReflectionTokenWithAntibot.sol
                    ↪ #1296-1302)
        - _transfer(address(this),marketingWallet,_marketingFeeTokens) (
            ↪ ReflectionTokenWithAntibot.sol#1266)
                - returndata = address(token).functionCall(data,SafeERC20:
                    ↪  low-level call failed) (node_modules/@openzeppelin
                    ↪ /contracts/token/ERC20/utils/SafeERC20.sol#110)
                - IERC20(baseTokenForPair).approve(address(mainRouter),
                    ↪ baseTokenAmount) (ReflectionTokenWithAntibot.sol
                    ↪ #1308)
                - mainRouter.addLiquidityETH{value: baseTokenAmount}(
                    ↪ address(this),tokenAmount,0,0,address(0xdead),block
                    ↪ .timestamp) (ReflectionTokenWithAntibot.sol
                    ↪ #1310-1317)
                - IGemAntiBot(gemAntiBot).onPreTransferCheck(from,to,
                    ↪ amount) (ReflectionTokenWithAntibot.sol#1172)
                - (success,returndata) = target.call{value: value}(data) (
                    ↪ node_modules/@openzeppelin/contracts/utils/Address.
                    ↪ sol#137)
                - mainRouter.addLiquidity(address(this),baseTokenForPair,
                    ↪ tokenAmount,baseTokenAmount,0,0,address(0xdead),
                    ↪ block.timestamp) (ReflectionTokenWithAntibot.sol
                    ↪ #1319-1328)
                - mainRouter.
                    ↪ swapExactTokensForETHSupportingFeeOnTransferTokens(
                    ↪ tokenAmount,0,path,address(this),block.timestamp) (
                    ↪ ReflectionTokenWithAntibot.sol#1287-1293)
                - uniswapV2Caller.
                    ↪ swapExactTokensForTokensSupportingFeeOnTransferTokens
                    ↪ (address(mainRouter),tokenAmount,0,path,block.
                    ↪ timestamp) (ReflectionTokenWithAntibot.sol
                    ↪ #1296-1302)
                - (success) = address(marketingWallet).call{value:
                    ↪ baseTokenForMarketing}() (
                    ↪ ReflectionTokenWithAntibot.sol#1249)
                - IERC20(baseTokenForPair).safeTransfer(marketingWallet,
                    ↪ baseTokenForMarketing) (ReflectionTokenWithAntibot.
                    ↪ sol#1255)
         - addLiquidity(tokensForLiquidity,baseTokenForLiquidity) (
             ↪ ReflectionTokenWithAntibot.sol#1273)
                - IERC20(baseTokenForPair).approve(address(mainRouter),
                    ↪ baseTokenAmount) (ReflectionTokenWithAntibot.sol
                    ↪ #1308)
                - mainRouter.addLiquidityETH{value: baseTokenAmount}(
                    ↪ address(this),tokenAmount,0,0,address(0xdead),block
                    ↪ .timestamp) (ReflectionTokenWithAntibot.sol
                    ↪ #1310-1317)
                - mainRouter.addLiquidity(address(this),baseTokenForPair,
                    ↪ tokenAmount,baseTokenAmount,0,0,address(0xdead),
                    ↪ block.timestamp) (ReflectionTokenWithAntibot.sol
```

```
                ↪ #1319-1328)
        External calls sending eth:
        - (success) = address(marketingWallet).call{value:
           ↪ baseTokenForMarketing}() (ReflectionTokenWithAntibot.sol
           ↪ #1249)
        - _transfer(address(this),marketingWallet,_marketingFeeTokens) (
           ↪ ReflectionTokenWithAntibot.sol#1266)
              - mainRouter.addLiquidityETH{value: baseTokenAmount}(
                 ↪ address(this),tokenAmount,0,0,address(0xdead),block
                 ↪ .timestamp) (ReflectionTokenWithAntibot.sol
                 ↪ #1310-1317)
              - (success,returndata) = target.call{value: value}(data) (
                 ↪ node_modules/@openzeppelin/contracts/utils/Address.
                 ↪ sol#137)
              - (success) = address(marketingWallet).call{value:
                 ↪ baseTokenForMarketing}() (
                 ↪ ReflectionTokenWithAntibot.sol#1249)
        - addLiquidity(tokensForLiquidity,baseTokenForLiquidity) (
           ↪ ReflectionTokenWithAntibot.sol#1273)
              - mainRouter.addLiquidityETH{value: baseTokenAmount}(
                 ↪ address(this),tokenAmount,0,0,address(0xdead),block
                 ↪ .timestamp) (ReflectionTokenWithAntibot.sol
                 ↪ #1310-1317)
        State variables written after the call(s):
        - _liquidityFeeTokens = 0 (ReflectionTokenWithAntibot.sol#1277)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #reentrancy-vulnerabilities

Reentrancy in ReflectionTokenWithAntibot.takeFee() (
    ↪ ReflectionTokenWithAntibot.sol#1227-1279):
        External calls:
        - swapTokensForBaseToken(tokensForSwap) (
           ↪ ReflectionTokenWithAntibot.sol#1242)
```

```
                - mainRouter.
                    ↪ swapExactTokensForETHSupportingFeeOnTransferTokens(
                    ↪ tokenAmount,0,path,address(this),block.timestamp) (
                    ↪ ReflectionTokenWithAntibot.sol#1287-1293)
                - uniswapV2Caller.
                    ↪ swapExactTokensForTokensSupportingFeeOnTransferTokens
                    ↪ (address(mainRouter),tokenAmount,0,path,block.
                    ↪ timestamp) (ReflectionTokenWithAntibot.sol
                    ↪ #1296-1302)
        - IERC20(baseTokenForPair).safeTransfer(marketingWallet,
            ↪ baseTokenForMarketing) (ReflectionTokenWithAntibot.sol
            ↪ #1255)
        State variables written after the call(s):
        - _marketingFeeTokens = 0 (ReflectionTokenWithAntibot.sol#1256)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #reentrancy-vulnerabilities-1

ReflectionTokenWithAntibot.addLiquidity(uint256,uint256) (
    ↪ ReflectionTokenWithAntibot.sol#1306-1329) ignores return value by
    ↪  IERC20(baseTokenForPair).approve(address(mainRouter),
    ↪ baseTokenAmount) (ReflectionTokenWithAntibot.sol#1308)
ReflectionTokenWithAntibot.addLiquidity(uint256,uint256) (
    ↪ ReflectionTokenWithAntibot.sol#1306-1329) ignores return value by
    ↪  mainRouter.addLiquidityETH{value: baseTokenAmount}(address(this)
    ↪ ,tokenAmount,0,0,address(0xdead),block.timestamp) (
    ↪ ReflectionTokenWithAntibot.sol#1310-1317)
ReflectionTokenWithAntibot.addLiquidity(uint256,uint256) (
    ↪ ReflectionTokenWithAntibot.sol#1306-1329) ignores return value by
    ↪  mainRouter.addLiquidity(address(this),baseTokenForPair,
    ↪ tokenAmount,baseTokenAmount,0,0,address(0xdead),block.timestamp)
    ↪ (ReflectionTokenWithAntibot.sol#1319-1328)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #unused-return
```

```
ReflectionTokenWithAntibot.allowance(address,address).owner (
    ↪ ReflectionTokenWithAntibot.sol#898) shadows:
        - Ownable.owner() (node_modules/@openzeppelin/contracts/access/
            ↪ Ownable.sol#43-45) (function)
ReflectionTokenWithAntibot._approve(address,address,uint256).owner (
    ↪ ReflectionTokenWithAntibot.sol#1027) shadows:
        - Ownable.owner() (node_modules/@openzeppelin/contracts/access/
            ↪ Ownable.sol#43-45) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #local-variable-shadowing

ReflectionTokenWithAntibot.updateUniswapV2Router(address)._mainPair (
    ↪ ReflectionTokenWithAntibot.sol#582-583) lacks a zero-check on :
                - mainPair = _mainPair (ReflectionTokenWithAntibot.sol
                    ↪ #584)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #missing-zero-address-validation

Reentrancy in ReflectionTokenWithAntibot._transfer(address,address,
    ↪ uint256) (ReflectionTokenWithAntibot.sol#1164-1225):
        External calls:
        - IGemAntiBot(gemAntiBot).onPreTransferCheck(from,to,amount) (
            ↪ ReflectionTokenWithAntibot.sol#1172)
        - takeFee() (ReflectionTokenWithAntibot.sol#1186)
                - returndata = address(token).functionCall(data,SafeERC20:
                    ↪  low-level call failed) (node_modules/@openzeppelin
                    ↪ /contracts/token/ERC20/utils/SafeERC20.sol#110)
                - IERC20(baseTokenForPair).approve(address(mainRouter),
                    ↪ baseTokenAmount) (ReflectionTokenWithAntibot.sol
                    ↪ #1308)
                - mainRouter.addLiquidityETH{value: baseTokenAmount}(
                    ↪ address(this),tokenAmount,0,0,address(0xdead),block
                    ↪ .timestamp) (ReflectionTokenWithAntibot.sol
                    ↪ #1310-1317)
                - IGemAntiBot(gemAntiBot).onPreTransferCheck(from,to,
                    ↪ amount) (ReflectionTokenWithAntibot.sol#1172)
                - (success,returndata) = target.call{value: value}(data) (
                    ↪ node_modules/@openzeppelin/contracts/utils/Address.
                    ↪ sol#137)
                - mainRouter.addLiquidity(address(this),baseTokenForPair,
                    ↪ tokenAmount,baseTokenAmount,0,0,address(0xdead),
                    ↪ block.timestamp) (ReflectionTokenWithAntibot.sol
                    ↪ #1319-1328)
                - mainRouter.
                    ↪ swapExactTokensForETHSupportingFeeOnTransferTokens(
                    ↪ tokenAmount,0,path,address(this),block.timestamp) (
                    ↪ ReflectionTokenWithAntibot.sol#1287-1293)
                - uniswapV2Caller.
                    ↪ swapExactTokensForTokensSupportingFeeOnTransferTokens
                    ↪ (address(mainRouter),tokenAmount,0,path,block.
                    ↪ timestamp) (ReflectionTokenWithAntibot.sol
                    ↪ #1296-1302)
                - (success) = address(marketingWallet).call{value:
                    ↪ baseTokenForMarketing}() (
                    ↪ ReflectionTokenWithAntibot.sol#1249)
                - IERC20(baseTokenForPair).safeTransfer(marketingWallet,
                    ↪ baseTokenForMarketing) (ReflectionTokenWithAntibot.
                    ↪ sol#1255)
        External calls sending eth:
        - takeFee() (ReflectionTokenWithAntibot.sol#1186)
                - mainRouter.addLiquidityETH{value: baseTokenAmount}(
                    ↪ address(this),tokenAmount,0,0,address(0xdead),block
                    ↪ .timestamp) (ReflectionTokenWithAntibot.sol
                    ↪ #1310-1317)
                - (success,returndata) = target.call{value: value}(data) (
                    ↪ node_modules/@openzeppelin/contracts/utils/Address.
                    ↪ sol#137)
```

```
              - (success) = address(marketingWallet).call{value:
                 ↪ baseTokenForMarketing}() (
                 ↪ ReflectionTokenWithAntibot.sol#1249)
        State variables written after the call(s):
        - takeFee() (ReflectionTokenWithAntibot.sol#1186)
              - _allowances[owner][spender] = amount (
                 ↪ ReflectionTokenWithAntibot.sol#1034)
Reentrancy in ReflectionTokenWithAntibot.takeFee() (
   ↪ ReflectionTokenWithAntibot.sol#1227-1279):
        External calls:
        - swapTokensForBaseToken(tokensForLiquidity) (
           ↪ ReflectionTokenWithAntibot.sol#1262)
              - mainRouter.
                 ↪ swapExactTokensForETHSupportingFeeOnTransferTokens(
                 ↪ tokenAmount,0,path,address(this),block.timestamp) (
                 ↪ ReflectionTokenWithAntibot.sol#1287-1293)
              - uniswapV2Caller.
                 ↪ swapExactTokensForTokensSupportingFeeOnTransferTokens
                 ↪ (address(mainRouter),tokenAmount,0,path,block.
                 ↪ timestamp) (ReflectionTokenWithAntibot.sol
                 ↪ #1296-1302)
        - _transfer(address(this),marketingWallet,_marketingFeeTokens) (
           ↪ ReflectionTokenWithAntibot.sol#1266)
              - returndata = address(token).functionCall(data,SafeERC20:
                 ↪  low-level call failed) (node_modules/@openzeppelin
                 ↪ /contracts/token/ERC20/utils/SafeERC20.sol#110)
              - IERC20(baseTokenForPair).approve(address(mainRouter),
                 ↪ baseTokenAmount) (ReflectionTokenWithAntibot.sol
                 ↪ #1308)
              - mainRouter.addLiquidityETH{value: baseTokenAmount}(
                 ↪ address(this),tokenAmount,0,0,address(0xdead),block
                 ↪ .timestamp) (ReflectionTokenWithAntibot.sol
                 ↪ #1310-1317)
```

```
            - IGemAntiBot(gemAntiBot).onPreTransferCheck(from,to,
              ↪ amount) (ReflectionTokenWithAntibot.sol#1172)
            - (success,returndata) = target.call{value: value}(data) (
              ↪ node_modules/@openzeppelin/contracts/utils/Address.
              ↪ sol#137)
            - mainRouter.addLiquidity(address(this),baseTokenForPair,
              ↪ tokenAmount,baseTokenAmount,0,0,address(0xdead),
              ↪ block.timestamp) (ReflectionTokenWithAntibot.sol
              ↪ #1319-1328)
            - mainRouter.
              ↪ swapExactTokensForETHSupportingFeeOnTransferTokens(
              ↪ tokenAmount,0,path,address(this),block.timestamp) (
              ↪ ReflectionTokenWithAntibot.sol#1287-1293)
            - uniswapV2Caller.
              ↪ swapExactTokensForTokensSupportingFeeOnTransferTokens
              ↪ (address(mainRouter),tokenAmount,0,path,block.
              ↪ timestamp) (ReflectionTokenWithAntibot.sol
              ↪ #1296-1302)
            - (success) = address(marketingWallet).call{value:
              ↪ baseTokenForMarketing}() (
              ↪ ReflectionTokenWithAntibot.sol#1249)
            - IERC20(baseTokenForPair).safeTransfer(marketingWallet,
              ↪ baseTokenForMarketing) (ReflectionTokenWithAntibot.
              ↪ sol#1255)
        External calls sending eth:
        - _transfer(address(this),marketingWallet,_marketingFeeTokens) (
          ↪ ReflectionTokenWithAntibot.sol#1266)
            - mainRouter.addLiquidityETH{value: baseTokenAmount}(
              ↪ address(this),tokenAmount,0,0,address(0xdead),block
              ↪ .timestamp) (ReflectionTokenWithAntibot.sol
              ↪ #1310-1317)
            - (success,returndata) = target.call{value: value}(data) (
              ↪ node_modules/@openzeppelin/contracts/utils/Address.
              ↪ sol#137)
```

```
                - (success) = address(marketingWallet).call{value:
                   ↪ baseTokenForMarketing}() (
                   ↪ ReflectionTokenWithAntibot.sol#1249)
        State variables written after the call(s):
        - _transfer(address(this),marketingWallet,_marketingFeeTokens) (
           ↪ ReflectionTokenWithAntibot.sol#1266)
                - _allowances[owner][spender] = amount (
                   ↪ ReflectionTokenWithAntibot.sol#1034)
Reentrancy in ReflectionTokenWithAntibot.takeFee() (
   ↪ ReflectionTokenWithAntibot.sol#1227-1279):
        External calls:
        - swapTokensForBaseToken(tokensForSwap) (
           ↪ ReflectionTokenWithAntibot.sol#1242)
                - mainRouter.
                   ↪ swapExactTokensForETHSupportingFeeOnTransferTokens(
                   ↪ tokenAmount,0,path,address(this),block.timestamp) (
                   ↪ ReflectionTokenWithAntibot.sol#1287-1293)
                - uniswapV2Caller.
                   ↪ swapExactTokensForTokensSupportingFeeOnTransferTokens
                   ↪ (address(mainRouter),tokenAmount,0,path,block.
                   ↪ timestamp) (ReflectionTokenWithAntibot.sol
                   ↪ #1296-1302)
        - (success) = address(marketingWallet).call{value:
           ↪ baseTokenForMarketing}() (ReflectionTokenWithAntibot.sol
           ↪ #1249)
        - IERC20(baseTokenForPair).safeTransfer(marketingWallet,
           ↪ baseTokenForMarketing) (ReflectionTokenWithAntibot.sol
           ↪ #1255)
        - swapTokensForBaseToken(tokensForLiquidity) (
           ↪ ReflectionTokenWithAntibot.sol#1262)
                - mainRouter.
                   ↪ swapExactTokensForETHSupportingFeeOnTransferTokens(
                   ↪ tokenAmount,0,path,address(this),block.timestamp) (
                   ↪ ReflectionTokenWithAntibot.sol#1287-1293)
                - uniswapV2Caller.
                   ↪ swapExactTokensForTokensSupportingFeeOnTransferTokens
                   ↪ (address(mainRouter),tokenAmount,0,path,block.
                   ↪ timestamp) (ReflectionTokenWithAntibot.sol
                   ↪ #1296-1302)
        - _transfer(address(this),marketingWallet,_marketingFeeTokens) (
           ↪ ReflectionTokenWithAntibot.sol#1266)
                - returndata = address(token).functionCall(data,SafeERC20:
                   ↪  low-level call failed) (node_modules/@openzeppelin
                   ↪ /contracts/token/ERC20/utils/SafeERC20.sol#110)
                - IERC20(baseTokenForPair).approve(address(mainRouter),
                   ↪ baseTokenAmount) (ReflectionTokenWithAntibot.sol
                   ↪ #1308)
                - mainRouter.addLiquidityETH{value: baseTokenAmount}(
                   ↪ address(this),tokenAmount,0,0,address(0xdead),block
                   ↪ .timestamp) (ReflectionTokenWithAntibot.sol
                   ↪ #1310-1317)
                - IGemAntiBot(gemAntiBot).onPreTransferCheck(from,to,
                   ↪ amount) (ReflectionTokenWithAntibot.sol#1172)
                - (success,returndata) = target.call{value: value}(data) (
                   ↪ node_modules/@openzeppelin/contracts/utils/Address.
                   ↪ sol#137)
                - mainRouter.addLiquidity(address(this),baseTokenForPair,
                   ↪ tokenAmount,baseTokenAmount,0,0,address(0xdead),
                   ↪ block.timestamp) (ReflectionTokenWithAntibot.sol
                   ↪ #1319-1328)
                - mainRouter.
                   ↪ swapExactTokensForETHSupportingFeeOnTransferTokens(
                   ↪ tokenAmount,0,path,address(this),block.timestamp) (
                   ↪ ReflectionTokenWithAntibot.sol#1287-1293)
                - uniswapV2Caller.
                   ↪ swapExactTokensForTokensSupportingFeeOnTransferTokens
                   ↪ (address(mainRouter),tokenAmount,0,path,block.
                   ↪ timestamp) (ReflectionTokenWithAntibot.sol
```

```
               ↪ #1296-1302)
            - (success) = address(marketingWallet).call{value:
               ↪ baseTokenForMarketing}() (
               ↪ ReflectionTokenWithAntibot.sol#1249)
            - IERC20(baseTokenForPair).safeTransfer(marketingWallet,
               ↪ baseTokenForMarketing) (ReflectionTokenWithAntibot.
               ↪ sol#1255)
        - addLiquidity(tokensForLiquidity,baseTokenForLiquidity) (
            ↪ ReflectionTokenWithAntibot.sol#1273)
            - IERC20(baseTokenForPair).approve(address(mainRouter),
               ↪ baseTokenAmount) (ReflectionTokenWithAntibot.sol
               ↪ #1308)
            - mainRouter.addLiquidityETH{value: baseTokenAmount}(
               ↪ address(this),tokenAmount,0,0,address(0xdead),block
               ↪ .timestamp) (ReflectionTokenWithAntibot.sol
               ↪ #1310-1317)
            - mainRouter.addLiquidity(address(this),baseTokenForPair,
               ↪ tokenAmount,baseTokenAmount,0,0,address(0xdead),
               ↪ block.timestamp) (ReflectionTokenWithAntibot.sol
               ↪ #1319-1328)
        External calls sending eth:
        - (success) = address(marketingWallet).call{value:
            ↪ baseTokenForMarketing}() (ReflectionTokenWithAntibot.sol
            ↪ #1249)
        - _transfer(address(this),marketingWallet,_marketingFeeTokens) (
            ↪ ReflectionTokenWithAntibot.sol#1266)
            - mainRouter.addLiquidityETH{value: baseTokenAmount}(
               ↪ address(this),tokenAmount,0,0,address(0xdead),block
               ↪ .timestamp) (ReflectionTokenWithAntibot.sol
               ↪ #1310-1317)
            - (success,returndata) = target.call{value: value}(data) (
               ↪ node_modules/@openzeppelin/contracts/utils/Address.
               ↪ sol#137)
            - (success) = address(marketingWallet).call{value:
                ↪ baseTokenForMarketing}() (
                ↪ ReflectionTokenWithAntibot.sol#1249)
        - addLiquidity(tokensForLiquidity,baseTokenForLiquidity) (
            ↪ ReflectionTokenWithAntibot.sol#1273)
            - mainRouter.addLiquidityETH{value: baseTokenAmount}(
                ↪ address(this),tokenAmount,0,0,address(0xdead),block
                ↪ .timestamp) (ReflectionTokenWithAntibot.sol
                ↪ #1310-1317)
        State variables written after the call(s):
        - addLiquidity(tokensForLiquidity,baseTokenForLiquidity) (
            ↪ ReflectionTokenWithAntibot.sol#1273)
            - _allowances[owner][spender] = amount (
                ↪ ReflectionTokenWithAntibot.sol#1034)
Reentrancy in ReflectionTokenWithAntibot.transferFrom(address,address,
    ↪ uint256) (ReflectionTokenWithAntibot.sol#916-928):
        External calls:
        - _transfer(sender,recipient,amount) (ReflectionTokenWithAntibot.
            ↪ sol#921)
            - returndata = address(token).functionCall(data,SafeERC20:
                ↪  low-level call failed) (node_modules/@openzeppelin
                ↪ /contracts/token/ERC20/utils/SafeERC20.sol#110)
            - IERC20(baseTokenForPair).approve(address(mainRouter),
                ↪ baseTokenAmount) (ReflectionTokenWithAntibot.sol
                ↪ #1308)
            - mainRouter.addLiquidityETH{value: baseTokenAmount}(
                ↪ address(this),tokenAmount,0,0,address(0xdead),block
                ↪ .timestamp) (ReflectionTokenWithAntibot.sol
                ↪ #1310-1317)
            - IGemAntiBot(gemAntiBot).onPreTransferCheck(from,to,
                ↪ amount) (ReflectionTokenWithAntibot.sol#1172)
            - (success,returndata) = target.call{value: value}(data) (
                ↪ node_modules/@openzeppelin/contracts/utils/Address.
                ↪ sol#137)
```

```
          - mainRouter.addLiquidity(address(this),baseTokenForPair,
             ↪ tokenAmount,baseTokenAmount,0,0,address(0xdead),
             ↪ block.timestamp) (ReflectionTokenWithAntibot.sol
             ↪ #1319-1328)
          - mainRouter.
             ↪ swapExactTokensForETHSupportingFeeOnTransferTokens(
             ↪ tokenAmount,0,path,address(this),block.timestamp) (
             ↪ ReflectionTokenWithAntibot.sol#1287-1293)
          - uniswapV2Caller.
             ↪ swapExactTokensForTokensSupportingFeeOnTransferTokens
             ↪ (address(mainRouter),tokenAmount,0,path,block.
             ↪ timestamp) (ReflectionTokenWithAntibot.sol
             ↪ #1296-1302)
          - (success) = address(marketingWallet).call{value:
             ↪ baseTokenForMarketing}() (
             ↪ ReflectionTokenWithAntibot.sol#1249)
          - IERC20(baseTokenForPair).safeTransfer(marketingWallet,
             ↪ baseTokenForMarketing) (ReflectionTokenWithAntibot.
             ↪ sol#1255)
    External calls sending eth:
    - _transfer(sender,recipient,amount) (ReflectionTokenWithAntibot.
       ↪ sol#921)
          - mainRouter.addLiquidityETH{value: baseTokenAmount}(
             ↪ address(this),tokenAmount,0,0,address(0xdead),block
             ↪ .timestamp) (ReflectionTokenWithAntibot.sol
             ↪ #1310-1317)
          - (success,returndata) = target.call{value: value}(data) (
             ↪ node_modules/@openzeppelin/contracts/utils/Address.
             ↪ sol#137)
          - (success) = address(marketingWallet).call{value:
             ↪ baseTokenForMarketing}() (
             ↪ ReflectionTokenWithAntibot.sol#1249)
    State variables written after the call(s):
        - _approve(sender,_msgSender(),_allowances[sender][_msgSender()]
           ↪ - amount) (ReflectionTokenWithAntibot.sol#922-926)
              - _allowances[owner][spender] = amount (
                 ↪ ReflectionTokenWithAntibot.sol#1034)
Reentrancy in ReflectionTokenWithAntibot.updateUniswapV2Router(address)
   ↪ (ReflectionTokenWithAntibot.sol#575-585):
        External calls:
        - _mainPair = IUniswapV2Factory(mainRouter.factory()).createPair(
           ↪ address(this),baseTokenForPair) (
           ↪ ReflectionTokenWithAntibot.sol#582-583)
        State variables written after the call(s):
        - mainPair = _mainPair (ReflectionTokenWithAntibot.sol#584)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #reentrancy-vulnerabilities-2

Reentrancy in ReflectionTokenWithAntibot._transfer(address,address,
   ↪ uint256) (ReflectionTokenWithAntibot.sol#1164-1225):
        External calls:
        - IGemAntiBot(gemAntiBot).onPreTransferCheck(from,to,amount) (
           ↪ ReflectionTokenWithAntibot.sol#1172)
        - takeFee() (ReflectionTokenWithAntibot.sol#1186)
              - returndata = address(token).functionCall(data,SafeERC20:
                 ↪  low-level call failed) (node_modules/@openzeppelin
                 ↪ /contracts/token/ERC20/utils/SafeERC20.sol#110)
              - IERC20(baseTokenForPair).approve(address(mainRouter),
                 ↪ baseTokenAmount) (ReflectionTokenWithAntibot.sol
                 ↪ #1308)
              - mainRouter.addLiquidityETH{value: baseTokenAmount}(
                 ↪ address(this),tokenAmount,0,0,address(0xdead),block
                 ↪ .timestamp) (ReflectionTokenWithAntibot.sol
                 ↪ #1310-1317)
              - IGemAntiBot(gemAntiBot).onPreTransferCheck(from,to,
                 ↪ amount) (ReflectionTokenWithAntibot.sol#1172)
```

```
          - (success,returndata) = target.call{value: value}(data) (
             ↪ node_modules/@openzeppelin/contracts/utils/Address.
             ↪ sol#137)
          - mainRouter.addLiquidity(address(this),baseTokenForPair,
             ↪ tokenAmount,baseTokenAmount,0,0,address(0xdead),
             ↪ block.timestamp) (ReflectionTokenWithAntibot.sol
             ↪ #1319-1328)
          - mainRouter.
             ↪ swapExactTokensForETHSupportingFeeOnTransferTokens(
             ↪ tokenAmount,0,path,address(this),block.timestamp) (
             ↪ ReflectionTokenWithAntibot.sol#1287-1293)
          - uniswapV2Caller.
             ↪ swapExactTokensForTokensSupportingFeeOnTransferTokens
             ↪ (address(mainRouter),tokenAmount,0,path,block.
             ↪ timestamp) (ReflectionTokenWithAntibot.sol
             ↪ #1296-1302)
          - (success) = address(marketingWallet).call{value:
             ↪ baseTokenForMarketing}() (
             ↪ ReflectionTokenWithAntibot.sol#1249)
          - IERC20(baseTokenForPair).safeTransfer(marketingWallet,
             ↪ baseTokenForMarketing) (ReflectionTokenWithAntibot.
             ↪ sol#1255)
    External calls sending eth:
    - takeFee() (ReflectionTokenWithAntibot.sol#1186)
          - mainRouter.addLiquidityETH{value: baseTokenAmount}(
             ↪ address(this),tokenAmount,0,0,address(0xdead),block
             ↪ .timestamp) (ReflectionTokenWithAntibot.sol
             ↪ #1310-1317)
          - (success,returndata) = target.call{value: value}(data) (
             ↪ node_modules/@openzeppelin/contracts/utils/Address.
             ↪ sol#137)
          - (success) = address(marketingWallet).call{value:
             ↪ baseTokenForMarketing}() (
             ↪ ReflectionTokenWithAntibot.sol#1249)
    Event emitted after the call(s):
    - Approval(owner,spender,amount) (ReflectionTokenWithAntibot.sol
       ↪ #1035)
          - takeFee() (ReflectionTokenWithAntibot.sol#1186)
    - MarketingFeeTaken(0,baseTokenForMarketing) (
       ↪ ReflectionTokenWithAntibot.sol#1252)
          - takeFee() (ReflectionTokenWithAntibot.sol#1186)
    - MarketingFeeTaken(0,baseTokenForMarketing) (
       ↪ ReflectionTokenWithAntibot.sol#1257)
          - takeFee() (ReflectionTokenWithAntibot.sol#1186)
    - MarketingFeeTaken(_marketingFeeTokens,0) (
       ↪ ReflectionTokenWithAntibot.sol#1267)
          - takeFee() (ReflectionTokenWithAntibot.sol#1186)
    - SwapAndLiquify(tokensForLiquidity,baseTokenForLiquidity) (
       ↪ ReflectionTokenWithAntibot.sol#1274)
          - takeFee() (ReflectionTokenWithAntibot.sol#1186)
    - Transfer(sender,recipient,tTransferAmount) (
       ↪ ReflectionTokenWithAntibot.sol#634)
          - _tokenTransfer(from,to,amount) (
             ↪ ReflectionTokenWithAntibot.sol#1209)
    - Transfer(sender,recipient,tTransferAmount) (
       ↪ ReflectionTokenWithAntibot.sol#634)
          - takeFee() (ReflectionTokenWithAntibot.sol#1186)
    - Transfer(sender,recipient,tTransferAmount) (
       ↪ ReflectionTokenWithAntibot.sol#678)
          - _tokenTransfer(from,to,amount) (
             ↪ ReflectionTokenWithAntibot.sol#1209)
    - Transfer(sender,recipient,tTransferAmount) (
       ↪ ReflectionTokenWithAntibot.sol#656)
          - _tokenTransfer(from,to,amount) (
             ↪ ReflectionTokenWithAntibot.sol#1209)
    - Transfer(sender,recipient,tTransferAmount) (
       ↪ ReflectionTokenWithAntibot.sol#678)
          - takeFee() (ReflectionTokenWithAntibot.sol#1186)
```

```
        - Transfer(sender,recipient,tTransferAmount) (
            ↪ ReflectionTokenWithAntibot.sol#656)
                - takeFee() (ReflectionTokenWithAntibot.sol#1186)
        - Transfer(sender,recipient,tTransferAmount) (
            ↪ ReflectionTokenWithAntibot.sol#701)
                - takeFee() (ReflectionTokenWithAntibot.sol#1186)
        - Transfer(sender,recipient,tTransferAmount) (
            ↪ ReflectionTokenWithAntibot.sol#701)
                - _tokenTransfer(from,to,amount) (
                    ↪ ReflectionTokenWithAntibot.sol#1209)
Reentrancy in ReflectionTokenWithAntibot.takeFee() (
    ↪ ReflectionTokenWithAntibot.sol#1227-1279):
        External calls:
        - swapTokensForBaseToken(tokensForSwap) (
            ↪ ReflectionTokenWithAntibot.sol#1242)
                - mainRouter.
                    ↪ swapExactTokensForETHSupportingFeeOnTransferTokens(
                    ↪ tokenAmount,0,path,address(this),block.timestamp) (
                    ↪ ReflectionTokenWithAntibot.sol#1287-1293)
                - uniswapV2Caller.
                    ↪ swapExactTokensForTokensSupportingFeeOnTransferTokens
                    ↪ (address(mainRouter),tokenAmount,0,path,block.
                    ↪ timestamp) (ReflectionTokenWithAntibot.sol
                    ↪ #1296-1302)
        - (success) = address(marketingWallet).call{value:
            ↪ baseTokenForMarketing}() (ReflectionTokenWithAntibot.sol
            ↪ #1249)
        External calls sending eth:
        - (success) = address(marketingWallet).call{value:
            ↪ baseTokenForMarketing}() (ReflectionTokenWithAntibot.sol
            ↪ #1249)
        Event emitted after the call(s):
        - MarketingFeeTaken(0,baseTokenForMarketing) (
            ↪ ReflectionTokenWithAntibot.sol#1252)
```

```
Reentrancy in ReflectionTokenWithAntibot.takeFee() (
   ↪ ReflectionTokenWithAntibot.sol#1227-1279):
         External calls:
         - swapTokensForBaseToken(tokensForSwap) (
            ↪ ReflectionTokenWithAntibot.sol#1242)
               - mainRouter.
                  ↪ swapExactTokensForETHSupportingFeeOnTransferTokens(
                  ↪ tokenAmount,0,path,address(this),block.timestamp) (
                  ↪ ReflectionTokenWithAntibot.sol#1287-1293)
               - uniswapV2Caller.
                  ↪ swapExactTokensForTokensSupportingFeeOnTransferTokens
                  ↪ (address(mainRouter),tokenAmount,0,path,block.
                  ↪ timestamp) (ReflectionTokenWithAntibot.sol
                  ↪ #1296-1302)
         - IERC20(baseTokenForPair).safeTransfer(marketingWallet,
            ↪ baseTokenForMarketing) (ReflectionTokenWithAntibot.sol
            ↪ #1255)
         Event emitted after the call(s):
         - MarketingFeeTaken(0,baseTokenForMarketing) (
            ↪ ReflectionTokenWithAntibot.sol#1257)
Reentrancy in ReflectionTokenWithAntibot.takeFee() (
   ↪ ReflectionTokenWithAntibot.sol#1227-1279):
         External calls:
         - swapTokensForBaseToken(tokensForLiquidity) (
            ↪ ReflectionTokenWithAntibot.sol#1262)
               - mainRouter.
                  ↪ swapExactTokensForETHSupportingFeeOnTransferTokens(
                  ↪ tokenAmount,0,path,address(this),block.timestamp) (
                  ↪ ReflectionTokenWithAntibot.sol#1287-1293)
               - uniswapV2Caller.
                  ↪ swapExactTokensForTokensSupportingFeeOnTransferTokens
                  ↪ (address(mainRouter),tokenAmount,0,path,block.
                  ↪ timestamp) (ReflectionTokenWithAntibot.sol
                  ↪ #1296-1302)
```

```
        - _transfer(address(this),marketingWallet,_marketingFeeTokens) (
           ↪ ReflectionTokenWithAntibot.sol#1266)
              - returndata = address(token).functionCall(data,SafeERC20:
                 ↪  low-level call failed) (node_modules/@openzeppelin
                 ↪ /contracts/token/ERC20/utils/SafeERC20.sol#110)
              - IERC20(baseTokenForPair).approve(address(mainRouter),
                 ↪ baseTokenAmount) (ReflectionTokenWithAntibot.sol
                 ↪ #1308)
              - mainRouter.addLiquidityETH{value: baseTokenAmount}(
                 ↪ address(this),tokenAmount,0,0,address(0xdead),block
                 ↪ .timestamp) (ReflectionTokenWithAntibot.sol
                 ↪ #1310-1317)
              - IGemAntiBot(gemAntiBot).onPreTransferCheck(from,to,
                 ↪ amount) (ReflectionTokenWithAntibot.sol#1172)
              - (success,returndata) = target.call{value: value}(data) (
                 ↪ node_modules/@openzeppelin/contracts/utils/Address.
                 ↪ sol#137)
              - mainRouter.addLiquidity(address(this),baseTokenForPair,
                 ↪ tokenAmount,baseTokenAmount,0,0,address(0xdead),
                 ↪ block.timestamp) (ReflectionTokenWithAntibot.sol
                 ↪ #1319-1328)
              - mainRouter.
                 ↪ swapExactTokensForETHSupportingFeeOnTransferTokens(
                 ↪ tokenAmount,0,path,address(this),block.timestamp) (
                 ↪ ReflectionTokenWithAntibot.sol#1287-1293)
              - uniswapV2Caller.
                 ↪ swapExactTokensForTokensSupportingFeeOnTransferTokens
                 ↪ (address(mainRouter),tokenAmount,0,path,block.
                 ↪ timestamp) (ReflectionTokenWithAntibot.sol
                 ↪ #1296-1302)
              - (success) = address(marketingWallet).call{value:
                 ↪ baseTokenForMarketing}() (
                 ↪ ReflectionTokenWithAntibot.sol#1249)
```

```
            - IERC20(baseTokenForPair).safeTransfer(marketingWallet,
               ↪ baseTokenForMarketing) (ReflectionTokenWithAntibot.
               ↪ sol#1255)
    External calls sending eth:
    - _transfer(address(this),marketingWallet,_marketingFeeTokens) (
       ↪ ReflectionTokenWithAntibot.sol#1266)
            - mainRouter.addLiquidityETH{value: baseTokenAmount}(
               ↪ address(this),tokenAmount,0,0,address(0xdead),block
               ↪ .timestamp) (ReflectionTokenWithAntibot.sol
               ↪ #1310-1317)
            - (success,returndata) = target.call{value: value}(data) (
               ↪ node_modules/@openzeppelin/contracts/utils/Address.
               ↪ sol#137)
            - (success) = address(marketingWallet).call{value:
               ↪ baseTokenForMarketing}() (
               ↪ ReflectionTokenWithAntibot.sol#1249)
    Event emitted after the call(s):
    - Approval(owner,spender,amount) (ReflectionTokenWithAntibot.sol
       ↪ #1035)
            - _transfer(address(this),marketingWallet,
               ↪ _marketingFeeTokens) (ReflectionTokenWithAntibot.
               ↪ sol#1266)
    - MarketingFeeTaken(0,baseTokenForMarketing) (
       ↪ ReflectionTokenWithAntibot.sol#1252)
            - _transfer(address(this),marketingWallet,
               ↪ _marketingFeeTokens) (ReflectionTokenWithAntibot.
               ↪ sol#1266)
    - MarketingFeeTaken(0,baseTokenForMarketing) (
       ↪ ReflectionTokenWithAntibot.sol#1257)
            - _transfer(address(this),marketingWallet,
               ↪ _marketingFeeTokens) (ReflectionTokenWithAntibot.
               ↪ sol#1266)
    - MarketingFeeTaken(_marketingFeeTokens,0) (
       ↪ ReflectionTokenWithAntibot.sol#1267)
```

```
                - _transfer(address(this),marketingWallet,
                   ↪ _marketingFeeTokens) (ReflectionTokenWithAntibot.
                   ↪ sol#1266)
          - MarketingFeeTaken(_marketingFeeTokens,0) (
             ↪ ReflectionTokenWithAntibot.sol#1267)
          - SwapAndLiquify(tokensForLiquidity,baseTokenForLiquidity) (
             ↪ ReflectionTokenWithAntibot.sol#1274)
                - _transfer(address(this),marketingWallet,
                   ↪ _marketingFeeTokens) (ReflectionTokenWithAntibot.
                   ↪ sol#1266)
          - Transfer(sender,recipient,tTransferAmount) (
             ↪ ReflectionTokenWithAntibot.sol#634)
                - _transfer(address(this),marketingWallet,
                   ↪ _marketingFeeTokens) (ReflectionTokenWithAntibot.
                   ↪ sol#1266)
          - Transfer(sender,recipient,tTransferAmount) (
             ↪ ReflectionTokenWithAntibot.sol#678)
                - _transfer(address(this),marketingWallet,
                   ↪ _marketingFeeTokens) (ReflectionTokenWithAntibot.
                   ↪ sol#1266)
          - Transfer(sender,recipient,tTransferAmount) (
             ↪ ReflectionTokenWithAntibot.sol#656)
                - _transfer(address(this),marketingWallet,
                   ↪ _marketingFeeTokens) (ReflectionTokenWithAntibot.
                   ↪ sol#1266)
          - Transfer(sender,recipient,tTransferAmount) (
             ↪ ReflectionTokenWithAntibot.sol#701)
                - _transfer(address(this),marketingWallet,
                   ↪ _marketingFeeTokens) (ReflectionTokenWithAntibot.
                   ↪ sol#1266)
    Reentrancy in ReflectionTokenWithAntibot.takeFee() (
       ↪ ReflectionTokenWithAntibot.sol#1227-1279):
          External calls:
```

```
        - swapTokensForBaseToken(tokensForSwap) (
           ↪ ReflectionTokenWithAntibot.sol#1242)
              - mainRouter.
                 ↪ swapExactTokensForETHSupportingFeeOnTransferTokens(
                 ↪ tokenAmount,0,path,address(this),block.timestamp) (
                 ↪ ReflectionTokenWithAntibot.sol#1287-1293)
              - uniswapV2Caller.
                 ↪ swapExactTokensForTokensSupportingFeeOnTransferTokens
                 ↪ (address(mainRouter),tokenAmount,0,path,block.
                 ↪ timestamp) (ReflectionTokenWithAntibot.sol
                 ↪ #1296-1302)
        - (success) = address(marketingWallet).call{value:
           ↪ baseTokenForMarketing}() (ReflectionTokenWithAntibot.sol
           ↪ #1249)
        - IERC20(baseTokenForPair).safeTransfer(marketingWallet,
           ↪ baseTokenForMarketing) (ReflectionTokenWithAntibot.sol
           ↪ #1255)
        - swapTokensForBaseToken(tokensForLiquidity) (
           ↪ ReflectionTokenWithAntibot.sol#1262)
              - mainRouter.
                 ↪ swapExactTokensForETHSupportingFeeOnTransferTokens(
                 ↪ tokenAmount,0,path,address(this),block.timestamp) (
                 ↪ ReflectionTokenWithAntibot.sol#1287-1293)
              - uniswapV2Caller.
                 ↪ swapExactTokensForTokensSupportingFeeOnTransferTokens
                 ↪ (address(mainRouter),tokenAmount,0,path,block.
                 ↪ timestamp) (ReflectionTokenWithAntibot.sol
                 ↪ #1296-1302)
        - _transfer(address(this),marketingWallet,_marketingFeeTokens) (
           ↪ ReflectionTokenWithAntibot.sol#1266)
              - returndata = address(token).functionCall(data,SafeERC20:
                 ↪  low-level call failed) (node_modules/@openzeppelin
                 ↪ /contracts/token/ERC20/utils/SafeERC20.sol#110)
```

```
            - IERC20(baseTokenForPair).approve(address(mainRouter),
              ↪ baseTokenAmount) (ReflectionTokenWithAntibot.sol
              ↪ #1308)
            - mainRouter.addLiquidityETH{value: baseTokenAmount}(
              ↪ address(this),tokenAmount,0,0,address(0xdead),block
              ↪ .timestamp) (ReflectionTokenWithAntibot.sol
              ↪ #1310-1317)
            - IGemAntiBot(gemAntiBot).onPreTransferCheck(from,to,
              ↪ amount) (ReflectionTokenWithAntibot.sol#1172)
            - (success,returndata) = target.call{value: value}(data) (
              ↪ node_modules/@openzeppelin/contracts/utils/Address.
              ↪ sol#137)
            - mainRouter.addLiquidity(address(this),baseTokenForPair,
              ↪ tokenAmount,baseTokenAmount,0,0,address(0xdead),
              ↪ block.timestamp) (ReflectionTokenWithAntibot.sol
              ↪ #1319-1328)
            - mainRouter.
              ↪ swapExactTokensForETHSupportingFeeOnTransferTokens(
              ↪ tokenAmount,0,path,address(this),block.timestamp) (
              ↪ ReflectionTokenWithAntibot.sol#1287-1293)
            - uniswapV2Caller.
              ↪ swapExactTokensForTokensSupportingFeeOnTransferTokens
              ↪ (address(mainRouter),tokenAmount,0,path,block.
              ↪ timestamp) (ReflectionTokenWithAntibot.sol
              ↪ #1296-1302)
            - (success) = address(marketingWallet).call{value:
              ↪ baseTokenForMarketing}() (
              ↪ ReflectionTokenWithAntibot.sol#1249)
            - IERC20(baseTokenForPair).safeTransfer(marketingWallet,
              ↪ baseTokenForMarketing) (ReflectionTokenWithAntibot.
              ↪ sol#1255)
        - addLiquidity(tokensForLiquidity,baseTokenForLiquidity) (
          ↪ ReflectionTokenWithAntibot.sol#1273)
          - IERC20(baseTokenForPair).approve(address(mainRouter),
             ↪ baseTokenAmount) (ReflectionTokenWithAntibot.sol
             ↪ #1308)
          - mainRouter.addLiquidityETH{value: baseTokenAmount}(
             ↪ address(this),tokenAmount,0,0,address(0xdead),block
             ↪ .timestamp) (ReflectionTokenWithAntibot.sol
             ↪ #1310-1317)
          - mainRouter.addLiquidity(address(this),baseTokenForPair,
             ↪ tokenAmount,baseTokenAmount,0,0,address(0xdead),
             ↪ block.timestamp) (ReflectionTokenWithAntibot.sol
             ↪ #1319-1328)
    External calls sending eth:
    - (success) = address(marketingWallet).call{value:
       ↪ baseTokenForMarketing}() (ReflectionTokenWithAntibot.sol
       ↪ #1249)
    - _transfer(address(this),marketingWallet,_marketingFeeTokens) (
       ↪ ReflectionTokenWithAntibot.sol#1266)
          - mainRouter.addLiquidityETH{value: baseTokenAmount}(
             ↪ address(this),tokenAmount,0,0,address(0xdead),block
             ↪ .timestamp) (ReflectionTokenWithAntibot.sol
             ↪ #1310-1317)
          - (success,returndata) = target.call{value: value}(data) (
             ↪ node_modules/@openzeppelin/contracts/utils/Address.
             ↪ sol#137)
          - (success) = address(marketingWallet).call{value:
             ↪ baseTokenForMarketing}() (
             ↪ ReflectionTokenWithAntibot.sol#1249)
    - addLiquidity(tokensForLiquidity,baseTokenForLiquidity) (
       ↪ ReflectionTokenWithAntibot.sol#1273)
          - mainRouter.addLiquidityETH{value: baseTokenAmount}(
             ↪ address(this),tokenAmount,0,0,address(0xdead),block
             ↪ .timestamp) (ReflectionTokenWithAntibot.sol
             ↪ #1310-1317)
    Event emitted after the call(s):
        - Approval(owner,spender,amount) (ReflectionTokenWithAntibot.sol
           ↪ #1035)
              - addLiquidity(tokensForLiquidity,baseTokenForLiquidity) (
                 ↪ ReflectionTokenWithAntibot.sol#1273)
        - SwapAndLiquify(tokensForLiquidity,baseTokenForLiquidity) (
           ↪ ReflectionTokenWithAntibot.sol#1274)
 Reentrancy in ReflectionTokenWithAntibot.transferFrom(address,address,
     ↪ uint256) (ReflectionTokenWithAntibot.sol#916-928):
        External calls:
        - _transfer(sender,recipient,amount) (ReflectionTokenWithAntibot.
           ↪ sol#921)
              - returndata = address(token).functionCall(data,SafeERC20:
                 ↪  low-level call failed) (node_modules/@openzeppelin
                 ↪ /contracts/token/ERC20/utils/SafeERC20.sol#110)
              - IERC20(baseTokenForPair).approve(address(mainRouter),
                 ↪ baseTokenAmount) (ReflectionTokenWithAntibot.sol
                 ↪ #1308)
              - mainRouter.addLiquidityETH{value: baseTokenAmount}(
                 ↪ address(this),tokenAmount,0,0,address(0xdead),block
                 ↪ .timestamp) (ReflectionTokenWithAntibot.sol
                 ↪ #1310-1317)
              - IGemAntiBot(gemAntiBot).onPreTransferCheck(from,to,
                 ↪ amount) (ReflectionTokenWithAntibot.sol#1172)
              - (success,returndata) = target.call{value: value}(data) (
                 ↪ node_modules/@openzeppelin/contracts/utils/Address.
                 ↪ sol#137)
              - mainRouter.addLiquidity(address(this),baseTokenForPair,
                 ↪ tokenAmount,baseTokenAmount,0,0,address(0xdead),
                 ↪ block.timestamp) (ReflectionTokenWithAntibot.sol
                 ↪ #1319-1328)
              - mainRouter.
                 ↪ swapExactTokensForETHSupportingFeeOnTransferTokens(
                 ↪ tokenAmount,0,path,address(this),block.timestamp) (
                 ↪ ReflectionTokenWithAntibot.sol#1287-1293)
```

```
                - uniswapV2Caller.
                    ↪ swapExactTokensForTokensSupportingFeeOnTransferTokens
                    ↪ (address(mainRouter),tokenAmount,0,path,block.
                    ↪ timestamp) (ReflectionTokenWithAntibot.sol
                    ↪ #1296-1302)
                - (success) = address(marketingWallet).call{value:
                    ↪ baseTokenForMarketing}() (
                    ↪ ReflectionTokenWithAntibot.sol#1249)
                - IERC20(baseTokenForPair).safeTransfer(marketingWallet,
                    ↪ baseTokenForMarketing) (ReflectionTokenWithAntibot.
                    ↪ sol#1255)
        External calls sending eth:
        - _transfer(sender,recipient,amount) (ReflectionTokenWithAntibot.
            ↪ sol#921)
                - mainRouter.addLiquidityETH{value: baseTokenAmount}(
                    ↪ address(this),tokenAmount,0,0,address(0xdead),block
                    ↪ .timestamp) (ReflectionTokenWithAntibot.sol
                    ↪ #1310-1317)
                - (success,returndata) = target.call{value: value}(data) (
                    ↪ node_modules/@openzeppelin/contracts/utils/Address.
                    ↪ sol#137)
                - (success) = address(marketingWallet).call{value:
                    ↪ baseTokenForMarketing}() (
                    ↪ ReflectionTokenWithAntibot.sol#1249)
        Event emitted after the call(s):
        - Approval(owner,spender,amount) (ReflectionTokenWithAntibot.sol
            ↪ #1035)
                - _approve(sender,_msgSender(),_allowances[sender][
                    ↪ _msgSender()] - amount) (ReflectionTokenWithAntibot
                    ↪ .sol#922-926)
    Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
        ↪ #reentrancy-vulnerabilities-3
```

```
Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/
   ↪ contracts/utils/Address.sol#201-221) uses assembly
       - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.
          ↪ sol#213-216)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #assembly-usage

Different versions of Solidity are used:
       - Version used: ['0.8.13', '^0.8.0', '^0.8.1']
       - 0.8.13 (ReflectionTokenWithAntibot.sol#2)
       - ^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol
          ↪ #4)
       - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20
          ↪ .sol#4)
       - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/
          ↪ extensions/draft-IERC20Permit.sol#4)
       - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/utils/
          ↪ SafeERC20.sol#4)
       - ^0.8.1 (node_modules/@openzeppelin/contracts/utils/Address.sol
          ↪ #4)
       - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol
          ↪ #4)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #different-pragma-directives-are-used

ReflectionTokenWithAntibot.includeInReward(address) (
   ↪ ReflectionTokenWithAntibot.sol#1011-1024) has costly operations
   ↪ inside a loop:
       - _rTotal = _rTotal + _rOwned[account] - prev_rOwned (
          ↪ ReflectionTokenWithAntibot.sol#1017)
ReflectionTokenWithAntibot.includeInReward(address) (
   ↪ ReflectionTokenWithAntibot.sol#1011-1024) has costly operations
   ↪ inside a loop:
       - _excluded.pop() (ReflectionTokenWithAntibot.sol#1020)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #costly-operations-inside-a-loop

Address.functionCall(address,bytes) (node_modules/@openzeppelin/
    ↪ contracts/utils/Address.sol#85-87) is never used and should be
    ↪ removed
Address.functionCallWithValue(address,bytes,uint256) (node_modules/
    ↪ @openzeppelin/contracts/utils/Address.sol#114-120) is never used
    ↪ and should be removed
Address.functionDelegateCall(address,bytes) (node_modules/@openzeppelin/
    ↪ contracts/utils/Address.sol#174-176) is never used and should be
    ↪ removed
Address.functionDelegateCall(address,bytes,string) (node_modules/
    ↪ @openzeppelin/contracts/utils/Address.sol#184-193) is never used
    ↪ and should be removed
Address.functionStaticCall(address,bytes) (node_modules/@openzeppelin/
    ↪ contracts/utils/Address.sol#147-149) is never used and should be
    ↪ removed
Address.functionStaticCall(address,bytes,string) (node_modules/
    ↪ @openzeppelin/contracts/utils/Address.sol#157-166) is never used
    ↪ and should be removed
Address.sendValue(address,uint256) (node_modules/@openzeppelin/contracts
    ↪ /utils/Address.sol#60-65) is never used and should be removed
Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.
    ↪ sol#21-23) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (node_modules/
    ↪ @openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#46-59) is
    ↪  never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (node_modules/
    ↪ @openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#70-81) is
    ↪  never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (node_modules/
    ↪ @openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#61-68) is
    ↪  never used and should be removed
```

```
SafeERC20.safePermit(IERC20Permit,address,address,uint256,uint256,uint8,
    ↪ bytes32,bytes32) (node_modules/@openzeppelin/contracts/token/
    ↪ ERC20/utils/SafeERC20.sol#83-97) is never used and should be
    ↪ removed
SafeERC20.safeTransferFrom(IERC20,address,address,uint256) (node_modules
    ↪ /@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#30-37)
    ↪ is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #dead-code

Pragma version0.8.13 (ReflectionTokenWithAntibot.sol#2) necessitates a
    ↪ version too recent to be trusted. Consider deploying with
    ↪ 0.6.12/0.7.6/0.8.7
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/
    ↪ Ownable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/
    ↪ IERC20.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/
    ↪ extensions/draft-IERC20Permit.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/
    ↪ utils/SafeERC20.sol#4) allows old versions
Pragma version^0.8.1 (node_modules/@openzeppelin/contracts/utils/Address
    ↪ .sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context
    ↪ .sol#4) allows old versions
solc-0.8.13 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #incorrect-versions-of-solidity

Low level call in ReflectionTokenWithAntibot.takeFee() (
    ↪ ReflectionTokenWithAntibot.sol#1227-1279):
        - (success) = address(marketingWallet).call{value:
            ↪ baseTokenForMarketing}() (ReflectionTokenWithAntibot.sol
            ↪ #1249)
```

```
Low level call in Address.sendValue(address,uint256) (node_modules/
    ↪ @openzeppelin/contracts/utils/Address.sol#60-65):
        - (success) = recipient.call{value: amount}() (node_modules/
            ↪ @openzeppelin/contracts/utils/Address.sol#63)
Low level call in Address.functionCallWithValue(address,bytes,uint256,
    ↪ string) (node_modules/@openzeppelin/contracts/utils/Address.sol
    ↪ #128-139):
        - (success,returndata) = target.call{value: value}(data) (
            ↪ node_modules/@openzeppelin/contracts/utils/Address.sol
            ↪ #137)
Low level call in Address.functionStaticCall(address,bytes,string) (
    ↪ node_modules/@openzeppelin/contracts/utils/Address.sol#157-166):
        - (success,returndata) = target.staticcall(data) (node_modules/
            ↪ @openzeppelin/contracts/utils/Address.sol#164)
Low level call in Address.functionDelegateCall(address,bytes,string) (
    ↪ node_modules/@openzeppelin/contracts/utils/Address.sol#184-193):
        - (success,returndata) = target.delegatecall(data) (node_modules/
            ↪ @openzeppelin/contracts/utils/Address.sol#191)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #low-level-calls

Function IUniswapV2Router01.WETH() (ReflectionTokenWithAntibot.sol#11)
    ↪ is not in mixedCase
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (ReflectionTokenWithAntibot.
    ↪ sol#247) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (ReflectionTokenWithAntibot.
    ↪ sol#249) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (ReflectionTokenWithAntibot.
    ↪ sol#280) is not in mixedCase
Function IUniswapV2Factory.INIT_CODE_PAIR_HASH() (
    ↪ ReflectionTokenWithAntibot.sol#352) is not in mixedCase
Parameter ReflectionTokenWithAntibot.updateUniswapV2Pair(address).
    ↪ _baseTokenForPair (ReflectionTokenWithAntibot.sol#562) is not in
    ↪ mixedCase
```

```
Parameter ReflectionTokenWithAntibot.updateMaxWallet(uint256)._maxWallet
    ↪  (ReflectionTokenWithAntibot.sol#586) is not in mixedCase
Parameter ReflectionTokenWithAntibot.updateMaxTransactionAmount(uint256)
    ↪ ._maxTransactionAmount (ReflectionTokenWithAntibot.sol#592) is
    ↪ not in mixedCase
Parameter ReflectionTokenWithAntibot.calculateRewardFee(uint256)._amount
    ↪  (ReflectionTokenWithAntibot.sol#828) is not in mixedCase
Parameter ReflectionTokenWithAntibot.calculateLiquidityFee(uint256).
    ↪ _amount (ReflectionTokenWithAntibot.sol#836) is not in mixedCase
Parameter ReflectionTokenWithAntibot.calculateMarketingFee(uint256).
    ↪ _amount (ReflectionTokenWithAntibot.sol#844) is not in mixedCase
Parameter ReflectionTokenWithAntibot.updateLiquidityFee(uint16,uint16).
    ↪ _sellLiquidityFee (ReflectionTokenWithAntibot.sol#1045) is not in
    ↪  mixedCase
Parameter ReflectionTokenWithAntibot.updateLiquidityFee(uint16,uint16).
    ↪ _buyLiquidityFee (ReflectionTokenWithAntibot.sol#1046) is not in
    ↪ mixedCase
Parameter ReflectionTokenWithAntibot.updateMarketingFee(uint16,uint16).
    ↪ _sellMarketingFee (ReflectionTokenWithAntibot.sol#1067) is not in
    ↪  mixedCase
Parameter ReflectionTokenWithAntibot.updateMarketingFee(uint16,uint16).
    ↪ _buyMarketingFee (ReflectionTokenWithAntibot.sol#1068) is not in
    ↪ mixedCase
Parameter ReflectionTokenWithAntibot.updateRewardFee(uint16,uint16).
    ↪ _sellRewardFee (ReflectionTokenWithAntibot.sol#1089) is not in
    ↪ mixedCase
Parameter ReflectionTokenWithAntibot.updateRewardFee(uint16,uint16).
    ↪ _buyRewardFee (ReflectionTokenWithAntibot.sol#1090) is not in
    ↪ mixedCase
Parameter ReflectionTokenWithAntibot.updateMarketingWallet(address,bool)
    ↪ ._marketingWallet (ReflectionTokenWithAntibot.sol#1111) is not in
    ↪  mixedCase
Parameter ReflectionTokenWithAntibot.updateMarketingWallet(address,bool)
    ↪ ._isMarketingFeeBaseToken (ReflectionTokenWithAntibot.sol#1112)
    ↪ is not in mixedCase
Parameter ReflectionTokenWithAntibot.updateMinAmountToTakeFee(uint256).
    ↪ _minAmountToTakeFee (ReflectionTokenWithAntibot.sol#1122) is not
    ↪ in mixedCase
Constant ReflectionTokenWithAntibot.uniswapV2Caller (
    ↪ ReflectionTokenWithAntibot.sol#380-381) is not in
    ↪ UPPER_CASE_WITH_UNDERSCORES
Constant ReflectionTokenWithAntibot.feeContract (
    ↪ ReflectionTokenWithAntibot.sol#382) is not in
    ↪ UPPER_CASE_WITH_UNDERSCORES
Function IERC20Permit.DOMAIN_SEPARATOR() (node_modules/@openzeppelin/
    ↪ contracts/token/ERC20/extensions/draft-IERC20Permit.sol#59) is
    ↪ not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #conformance-to-solidity-naming-conventions

Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256
    ↪ ,uint256,uint256,address,uint256).amountADesired (
    ↪ ReflectionTokenWithAntibot.sol#16) is too similar to
    ↪ IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,
    ↪ uint256,uint256,address,uint256).amountBDesired (
    ↪ ReflectionTokenWithAntibot.sol#17)
Variable ReflectionTokenWithAntibot._transferToExcluded(address,address,
    ↪ uint256).rTransferAmount (ReflectionTokenWithAntibot.sol#644) is
    ↪ too similar to ReflectionTokenWithAntibot._getValues(uint256).
    ↪ tTransferAmount (ReflectionTokenWithAntibot.sol#723)
Variable ReflectionTokenWithAntibot._transferBothExcluded(address,
    ↪ address,uint256).rTransferAmount (ReflectionTokenWithAntibot.sol
    ↪ #688) is too similar to ReflectionTokenWithAntibot.
    ↪ _transferStandard(address,address,uint256).tTransferAmount (
    ↪ ReflectionTokenWithAntibot.sol#625)
Variable ReflectionTokenWithAntibot._getRValues(uint256,uint256,uint256,
    ↪ uint256,uint256).rTransferAmount (ReflectionTokenWithAntibot.sol
    ↪ #784-786) is too similar to ReflectionTokenWithAntibot.
    ↪ _transferBothExcluded(address,address,uint256).tTransferAmount (
    ↪ ReflectionTokenWithAntibot.sol#690)
Variable ReflectionTokenWithAntibot._transferToExcluded(address,address,
    ↪ uint256).rTransferAmount (ReflectionTokenWithAntibot.sol#644) is
    ↪ too similar to ReflectionTokenWithAntibot._transferFromExcluded(
    ↪ address,address,uint256).tTransferAmount (
    ↪ ReflectionTokenWithAntibot.sol#668)
Variable ReflectionTokenWithAntibot._transferToExcluded(address,address,
    ↪ uint256).rTransferAmount (ReflectionTokenWithAntibot.sol#644) is
    ↪ too similar to ReflectionTokenWithAntibot._transferToExcluded(
    ↪ address,address,uint256).tTransferAmount (
    ↪ ReflectionTokenWithAntibot.sol#646)
Variable ReflectionTokenWithAntibot._transferStandard(address,address,
    ↪ uint256).rTransferAmount (ReflectionTokenWithAntibot.sol#623) is
    ↪ too similar to ReflectionTokenWithAntibot._transferFromExcluded(
    ↪ address,address,uint256).tTransferAmount (
    ↪ ReflectionTokenWithAntibot.sol#668)
Variable ReflectionTokenWithAntibot._transferStandard(address,address,
    ↪ uint256).rTransferAmount (ReflectionTokenWithAntibot.sol#623) is
    ↪ too similar to ReflectionTokenWithAntibot._getValues(uint256).
    ↪ tTransferAmount (ReflectionTokenWithAntibot.sol#723)
Variable ReflectionTokenWithAntibot._getValues(uint256).rTransferAmount
    ↪ (ReflectionTokenWithAntibot.sol#728) is too similar to
    ↪ ReflectionTokenWithAntibot._getValues(uint256).tTransferAmount (
    ↪ ReflectionTokenWithAntibot.sol#723)
Variable ReflectionTokenWithAntibot._transferBothExcluded(address,
    ↪ address,uint256).rTransferAmount (ReflectionTokenWithAntibot.sol
    ↪ #688) is too similar to ReflectionTokenWithAntibot._getTValues(
    ↪ uint256).tTransferAmount (ReflectionTokenWithAntibot.sol#759-761)
Variable ReflectionTokenWithAntibot._transferFromExcluded(address,
    ↪ address,uint256).rTransferAmount (ReflectionTokenWithAntibot.sol
    ↪ #666) is too similar to ReflectionTokenWithAntibot.
    ↪ _transferBothExcluded(address,address,uint256).tTransferAmount (
    ↪ ReflectionTokenWithAntibot.sol#690)
```

```
Variable ReflectionTokenWithAntibot._getRValues(uint256,uint256,uint256,
    ↪ uint256,uint256).rTransferAmount (ReflectionTokenWithAntibot.sol
    ↪ #784-786) is too similar to ReflectionTokenWithAntibot.
    ↪ _transferStandard(address,address,uint256).tTransferAmount (
    ↪ ReflectionTokenWithAntibot.sol#625)
Variable ReflectionTokenWithAntibot._getValues(uint256).rTransferAmount
    ↪ (ReflectionTokenWithAntibot.sol#728) is too similar to
    ↪ ReflectionTokenWithAntibot._transferFromExcluded(address,address,
    ↪ uint256).tTransferAmount (ReflectionTokenWithAntibot.sol#668)
Variable ReflectionTokenWithAntibot.reflectionFromToken(uint256,bool).
    ↪ rTransferAmount (ReflectionTokenWithAntibot.sol#980) is too
    ↪ similar to ReflectionTokenWithAntibot._transferBothExcluded(
    ↪ address,address,uint256).tTransferAmount (
    ↪ ReflectionTokenWithAntibot.sol#690)
Variable ReflectionTokenWithAntibot._transferFromExcluded(address,
    ↪ address,uint256).rTransferAmount (ReflectionTokenWithAntibot.sol
    ↪ #666) is too similar to ReflectionTokenWithAntibot.
    ↪ _transferStandard(address,address,uint256).tTransferAmount (
    ↪ ReflectionTokenWithAntibot.sol#625)
Variable ReflectionTokenWithAntibot._getValues(uint256).rTransferAmount
    ↪ (ReflectionTokenWithAntibot.sol#728) is too similar to
    ↪ ReflectionTokenWithAntibot._transferToExcluded(address,address,
    ↪ uint256).tTransferAmount (ReflectionTokenWithAntibot.sol#646)
Variable ReflectionTokenWithAntibot._getRValues(uint256,uint256,uint256,
    ↪ uint256,uint256).rTransferAmount (ReflectionTokenWithAntibot.sol
    ↪ #784-786) is too similar to ReflectionTokenWithAntibot.
    ↪ _getTValues(uint256).tTransferAmount (ReflectionTokenWithAntibot.
    ↪ sol#759-761)
Variable ReflectionTokenWithAntibot.reflectionFromToken(uint256,bool).
    ↪ rTransferAmount (ReflectionTokenWithAntibot.sol#980) is too
    ↪ similar to ReflectionTokenWithAntibot._transferStandard(address,
    ↪ address,uint256).tTransferAmount (ReflectionTokenWithAntibot.sol
    ↪ #625)
```

```
Variable ReflectionTokenWithAntibot._transferFromExcluded(address,
    ↪ address,uint256).rTransferAmount (ReflectionTokenWithAntibot.sol
    ↪ #666) is too similar to ReflectionTokenWithAntibot._getTValues(
    ↪ uint256).tTransferAmount (ReflectionTokenWithAntibot.sol#759-761)
Variable ReflectionTokenWithAntibot._transferToExcluded(address,address,
    ↪ uint256).rTransferAmount (ReflectionTokenWithAntibot.sol#644) is
    ↪ too similar to ReflectionTokenWithAntibot._transferBothExcluded(
    ↪ address,address,uint256).tTransferAmount (
    ↪ ReflectionTokenWithAntibot.sol#690)
Variable ReflectionTokenWithAntibot.reflectionFromToken(uint256,bool).
    ↪ rTransferAmount (ReflectionTokenWithAntibot.sol#980) is too
    ↪ similar to ReflectionTokenWithAntibot._getTValues(uint256).
    ↪ tTransferAmount (ReflectionTokenWithAntibot.sol#759-761)
Variable ReflectionTokenWithAntibot._transferStandard(address,address,
    ↪ uint256).rTransferAmount (ReflectionTokenWithAntibot.sol#623) is
    ↪ too similar to ReflectionTokenWithAntibot._transferBothExcluded(
    ↪ address,address,uint256).tTransferAmount (
    ↪ ReflectionTokenWithAntibot.sol#690)
Variable ReflectionTokenWithAntibot._transferToExcluded(address,address,
    ↪ uint256).rTransferAmount (ReflectionTokenWithAntibot.sol#644) is
    ↪ too similar to ReflectionTokenWithAntibot._transferStandard(
    ↪ address,address,uint256).tTransferAmount (
    ↪ ReflectionTokenWithAntibot.sol#625)
Variable ReflectionTokenWithAntibot._getValues(uint256).rTransferAmount
    ↪ (ReflectionTokenWithAntibot.sol#728) is too similar to
    ↪ ReflectionTokenWithAntibot._transferBothExcluded(address,address,
    ↪ uint256).tTransferAmount (ReflectionTokenWithAntibot.sol#690)
Variable ReflectionTokenWithAntibot._transferBothExcluded(address,
    ↪ address,uint256).rTransferAmount (ReflectionTokenWithAntibot.sol
    ↪ #688) is too similar to ReflectionTokenWithAntibot._getValues(
    ↪ uint256).tTransferAmount (ReflectionTokenWithAntibot.sol#723)
Variable ReflectionTokenWithAntibot._transferStandard(address,address,
    ↪ uint256).rTransferAmount (ReflectionTokenWithAntibot.sol#623) is
    ↪ too similar to ReflectionTokenWithAntibot._transferStandard(
    ↪ address,address,uint256).tTransferAmount (
    ↪ ReflectionTokenWithAntibot.sol#625)
Variable ReflectionTokenWithAntibot._getValues(uint256).rTransferAmount
    ↪ (ReflectionTokenWithAntibot.sol#728) is too similar to
    ↪ ReflectionTokenWithAntibot._transferStandard(address,address,
    ↪ uint256).tTransferAmount (ReflectionTokenWithAntibot.sol#625)
Variable ReflectionTokenWithAntibot._transferBothExcluded(address,
    ↪ address,uint256).rTransferAmount (ReflectionTokenWithAntibot.sol
    ↪ #688) is too similar to ReflectionTokenWithAntibot.
    ↪ _transferFromExcluded(address,address,uint256).tTransferAmount (
    ↪ ReflectionTokenWithAntibot.sol#668)
Variable ReflectionTokenWithAntibot._transferToExcluded(address,address,
    ↪ uint256).rTransferAmount (ReflectionTokenWithAntibot.sol#644) is
    ↪ too similar to ReflectionTokenWithAntibot._getTValues(uint256).
    ↪ tTransferAmount (ReflectionTokenWithAntibot.sol#759-761)
Variable ReflectionTokenWithAntibot._getRValues(uint256,uint256,uint256,
    ↪ uint256,uint256).rTransferAmount (ReflectionTokenWithAntibot.sol
    ↪ #784-786) is too similar to ReflectionTokenWithAntibot._getValues
    ↪ (uint256).tTransferAmount (ReflectionTokenWithAntibot.sol#723)
Variable ReflectionTokenWithAntibot._getRValues(uint256,uint256,uint256,
    ↪ uint256,uint256).rTransferAmount (ReflectionTokenWithAntibot.sol
    ↪ #784-786) is too similar to ReflectionTokenWithAntibot.
    ↪ _transferFromExcluded(address,address,uint256).tTransferAmount (
    ↪ ReflectionTokenWithAntibot.sol#668)
Variable ReflectionTokenWithAntibot._transferStandard(address,address,
    ↪ uint256).rTransferAmount (ReflectionTokenWithAntibot.sol#623) is
    ↪ too similar to ReflectionTokenWithAntibot._getTValues(uint256).
    ↪ tTransferAmount (ReflectionTokenWithAntibot.sol#759-761)
Variable ReflectionTokenWithAntibot._getValues(uint256).rTransferAmount
    ↪ (ReflectionTokenWithAntibot.sol#728) is too similar to
    ↪ ReflectionTokenWithAntibot._getTValues(uint256).tTransferAmount (
    ↪ ReflectionTokenWithAntibot.sol#759-761)
Variable ReflectionTokenWithAntibot._getRValues(uint256,uint256,uint256,
    ↪ uint256,uint256).rTransferAmount (ReflectionTokenWithAntibot.sol
    ↪ #784-786) is too similar to ReflectionTokenWithAntibot.
    ↪ _transferToExcluded(address,address,uint256).tTransferAmount (
    ↪ ReflectionTokenWithAntibot.sol#646)
Variable ReflectionTokenWithAntibot._transferFromExcluded(address,
    ↪ address,uint256).rTransferAmount (ReflectionTokenWithAntibot.sol
    ↪ #666) is too similar to ReflectionTokenWithAntibot.
    ↪ _transferFromExcluded(address,address,uint256).tTransferAmount (
    ↪ ReflectionTokenWithAntibot.sol#668)
Variable ReflectionTokenWithAntibot._transferFromExcluded(address,
    ↪ address,uint256).rTransferAmount (ReflectionTokenWithAntibot.sol
    ↪ #666) is too similar to ReflectionTokenWithAntibot._getValues(
    ↪ uint256).tTransferAmount (ReflectionTokenWithAntibot.sol#723)
Variable ReflectionTokenWithAntibot._transferFromExcluded(address,
    ↪ address,uint256).rTransferAmount (ReflectionTokenWithAntibot.sol
    ↪ #666) is too similar to ReflectionTokenWithAntibot.
    ↪ _transferToExcluded(address,address,uint256).tTransferAmount (
    ↪ ReflectionTokenWithAntibot.sol#646)
Variable ReflectionTokenWithAntibot.reflectionFromToken(uint256,bool).
    ↪ rTransferAmount (ReflectionTokenWithAntibot.sol#980) is too
    ↪ similar to ReflectionTokenWithAntibot._getValues(uint256).
    ↪ tTransferAmount (ReflectionTokenWithAntibot.sol#723)
Variable ReflectionTokenWithAntibot.reflectionFromToken(uint256,bool).
    ↪ rTransferAmount (ReflectionTokenWithAntibot.sol#980) is too
    ↪ similar to ReflectionTokenWithAntibot._transferFromExcluded(
    ↪ address,address,uint256).tTransferAmount (
    ↪ ReflectionTokenWithAntibot.sol#668)
Variable ReflectionTokenWithAntibot._transferBothExcluded(address,
    ↪ address,uint256).rTransferAmount (ReflectionTokenWithAntibot.sol
    ↪ #688) is too similar to ReflectionTokenWithAntibot.
    ↪ _transferBothExcluded(address,address,uint256).tTransferAmount (
    ↪ ReflectionTokenWithAntibot.sol#690)
Variable ReflectionTokenWithAntibot._transferBothExcluded(address,
    ↪ address,uint256).rTransferAmount (ReflectionTokenWithAntibot.sol
    ↪ #688) is too similar to ReflectionTokenWithAntibot.
    ↪ _transferToExcluded(address,address,uint256).tTransferAmount (
    ↪ ReflectionTokenWithAntibot.sol#646)
Variable ReflectionTokenWithAntibot._transferStandard(address,address,
    ↪ uint256).rTransferAmount (ReflectionTokenWithAntibot.sol#623) is
    ↪ too similar to ReflectionTokenWithAntibot._transferToExcluded(
    ↪ address,address,uint256).tTransferAmount (
    ↪ ReflectionTokenWithAntibot.sol#646)
Variable ReflectionTokenWithAntibot.reflectionFromToken(uint256,bool).
    ↪ rTransferAmount (ReflectionTokenWithAntibot.sol#980) is too
    ↪ similar to ReflectionTokenWithAntibot._transferToExcluded(address
    ↪ ,address,uint256).tTransferAmount (ReflectionTokenWithAntibot.sol
    ↪ #646)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #variable-names-are-too-similar

updateUniswapV2Router(address) should be declared external:
        - ReflectionTokenWithAntibot.updateUniswapV2Router(address) (
            ↪ ReflectionTokenWithAntibot.sol#575-585)
approve(address,uint256) should be declared external:
        - ReflectionTokenWithAntibot.approve(address,uint256) (
            ↪ ReflectionTokenWithAntibot.sol#907-914)
setAutomatedMarketMakerPair(address,bool) should be declared external:
        - ReflectionTokenWithAntibot.setAutomatedMarketMakerPair(address,
            ↪ bool) (ReflectionTokenWithAntibot.sol#1131-1136)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (node_modules/@openzeppelin/
            ↪ contracts/access/Ownable.sol#61-63)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (node_modules/@openzeppelin/
            ↪ contracts/access/Ownable.sol#69-72)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #public-function-that-could-be-declared-external
ReflectionTokenWithAntibot.sol analyzed (14 contracts with 78 detectors)
    ↪ , 125 result(s) found
```

Conclusion:

Most of the vulnerabilities found by the analysis have already been addressed by the smart contract code review.

5 Conclusion

We examined the design and implementation of Anu Initiative in this audit and found several issues of various severities. We advise Anu Initiative CLG team to implement the recommendations contained in all 7 of our findings to further enhance the code's security. It is of utmost priority to start by addressing the most severe exploit discovered by the auditors then followed by the remaining exploits, and finally we will be conducting a re-audit following the implementation of the remediation plan contained in this report.

We would much appreciate any constructive feedback or suggestions regarding our methodology, audit findings, or potential scope gaps in this report.

BLOCKHAT
SECURITY