FeedPups

Responses on Auditors Query

Issue 01

Type	Severity	Location
Owner Capabilities	Medium	`lock`, `unlock`

The contract uses a modified version of Ownable contract. These modifications have a significant flaw — a malicious owner can get his owner capabilities even after calling renounceOwnership!

Here's a list of the required steps:

1☐ The owner of the contract can call lock() to lock the contract (the lock function saves the previous owner into a variable)

2☐ After the locking period has passed the owner of the contract can call unlock() and regain the ownership.

3☐ The owner of the contract can then call the renounceOwnership function. Now, the contract allegedly has no owner (users can verify it by looking for the renounceOwnership transaction and making sure that the owner is set to the zero address).

4☐ The owner of the contract can call the unlock function again, and get the ownership back.

Recommendation

Remove `lock` and `unlock` functions.

Developer Response

We removed both the functions

Issue 02

Type	Severity	Location
Owner Capabilities	High	`setMaxTxPercent`

Description

The owner of the contract can make the tokens untradable By calling setMaxTxPercent(0)

Recommendation

Add a require statement that would limit setting `_maxTxAmount` to 0.

Developer Response

We added require statement

```
  function setMaxTxPercent(uint256 maxTxPercent) external onlyOwner() {
      require(maxTxPercent > 0, "Cannot set transaction amount less than 0 percent!");
      _maxTxAmount = _tTotal.mul(maxTxPercent).div(
          10**2
      );
  }
```

Issue 03

Type	Severity	Location
Best Practice	Informational	excludeFromFee includeInFee setCharityWallet setDevWallet setComunityWallet setMaxTxPercent

Description

Lack of events in the contract.

Recommendation

Emit events when changing the state variables of the contract.

Developer Response

Adding events on all suggested functions

Issue 04

Type	Severity	Location
Best Practice	MEDIUM	`receive`

Description

In order to prevent the contract from receiving ETH from investors, which will be resulted in a loss of funds, our recommendation is to have “whitelisted” addresses which can send ETH to the contract (for example the router address should be whitelisted).

When an investor sends ETH by mistake, the transaction will be reverted by the contract.

Developer Response

Modification done in receive function as recommended

```
  receive() external payable {

    //only allow pancakeRouter to send BNB

    require(msg.sender==address(0x10ED43C718714eb63d5aA57B78B54704E256024E));

  }
```

Issue 05

Type	Severity	Location
Volatile Code	High	`_transfer`

Description

`_transfer` may call internally to `swapExactTokensForETHSupportingFeeOnTransferTokens` and `addLiquidityETH`. Since this function can be called during `_transfer`, it may cause `_transfer` to fail unnecessarily.

Recommendation

_transfer should always work, and shouldn't fail if swapExactTokensForETHSupportingFeeOnTransferTokens or addLiquidityETH fails in order to make sure the token will always be tradable.

Developer Response

I believe _transfer will not call swapExactTokensForETHSupportingFeeOnTransferTokens and addLiquidityETH functions.
This we keep it as is without any change

Issue 06

Type	Severity	Location
Best Practice	Medium	`swapAndLiquify`

Description

The `swapAndLiquify` function converts half of the contract's tokens to BNB. The other half of the tokens are used for liquidity addition. The price of the token **drops after executing the first conversion**, having said that the other half of tokens require less than the converted BNB to be paired with it when adding liquidity.

Recommendation:

Our recommendation is to use the leftover BNBs for buyback. The team could add a simple buyback function that will use the leftover bnbs for buyback.

Developer Response

At line # 1049 added following function

```
  event SwapETHForTokens(
     uint256 amountIn,
     address[] path
  );
  address private deadAddress = 0x000000000000000000000000000000000000dEaD;
  function swapETHForTokens(uint256 amount) private {
     // generate the uniswap pair path of token -> weth
     address[] memory path = new address[](2);
     path[0] = uniswapV2Router.WETH();
     path[1] = address(this);

   // make the swap
     uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: amount}(
        0, // accept any amount of Tokens
        path,
        deadAddress, // Burn address
        block.timestamp.add(300)
```

```
    );

    emit SwapETHForTokens(amount, path);
  }

  function buyBackTokens(uint256 amount) private lockTheSwap {
        if (amount > 0) {
          swapETHForTokens(amount);
          }
  }
```

Issue 07

Type	Severity	Location
Logical Issue	Medium	`includeInReward`

Description: The code is vulnerable to the SafeMoon bug - excluding an address from the fee and then later including it back will cause the address to receive all RFI rewards for the time it was excluded (at the expense of other holders).

In `includeInReward` _rOwned is not updated. _rOwned should be updated and be calculated according to the current rate.

Recommendation: Properly calculate _rOwned of the included address in includeInReward based on its _tOwned amount

Developer Response

As recommended by auditors we have remove this function from the contract

Issue 08

Type	Severity	Location
Owner Capabilities	Medium	`addLiquidity`

Description

The recipient of the newly created LP tokens is the owner of the contract. The newly created LP tokens are unlocked.

```
    // add the liquidity
        uniswapV2Router.addLiquidityETH{value: ethAmount}(
```

```
            address(this),
            tokenAmount,
            0, // slippage is unavoidable
            0, // slippage is unavoidable
            owner(),
            block.timestamp
        );
```

Recommendation

Our recommendation is to change the recipient of the newly created LP tokens to the contract in order to ensure that the LP tokens are locked or to simply locked the tokens in the contract for a certain period.

Developer Response

With this we are not able to remove liquidity if anything goes wrong so I suggest to keep as is

Issue 09

Type	Severity	Location
Gas Optimization	Informational	`_approve`

approve is being called every transaction on the same tokens and for the same spender (the router).

Recommendation

In order to reduce gas costs, `approve` could be called once (with max int), and then check if it is needed again using `allowance`.

Developer Response

I believe it should be ok to use approve

Issue 10

Type	Severity	Location
Gas Optimization	Informational	`_approve`

Description

burnFee, and DevWallet seems to be not used in the contract.

Recommendation

In order to reduce gas costs, Remove unused code.

Developer Response

burnFee function we might use in future currently it is set as 0
DevWallet we removed

Issue 11

Type	Severity	Location
Volatile Code	Medium	`_getCurrentSupply`

Description

The `_excluded` array can grow indefinitely. `_transfer` calls `getCurrentSupply` function which iterates over `_excluded` array. Iterating over an unbounded array can exceed the block gas limit which will make the token untradable.

```
        address(this),
        tokenAmount,
        0, // slippage is unavoidable
        0, // slippage is unavoidable
        owner(),
        block.timestamp
    );
```

Recommendation

One possible solution is to limit the number of excluded addresses. We encourage the owner of the project and the community to frequently monitor the number of excluded addresses. The issue described could be recovered by calling `includeInReward` which removes an element from the _excluded array.

Developer Response

We add limit in excluding rewards to 20

```
  function excludeFromReward(address account) external onlyOwner() {
    require(account != 0x05fF2B0DB69458A0750badebc4f9e13aDd608C7F, 'We can not exclude Pancake router.');
require(!_isExcluded[account], "Account is already excluded");
        require(_excluded.length < 21, "Exclude reward is full");
    if(_rOwned[account] > 0) {
      _tOwned[account] = tokenFromReflection(_rOwned[account]);
    }
    _isExcluded[account] = true;
    _excluded.push(account);
  }
```

Issue 12

Type	Severity	Location
Volatile Code	Low	`swapAndLiquify`

Description

`numTokensSellToAddToLiquidity` is the number of tokens that will be sold by the contract as part of the automatic liquidity addition mechanism. At the moment it is set to 0.1% of the total supply. The team needs to acknowledge that the price impact every time swap and liquify will take place is determines by the amount of tokens in the pool.

Developer Response

We have set as 750,000,000,000,000.00

Issue 13

Type	Severity	Location
Volatile Code	Low	`swapAndLiquify`

Description

The public functions deliver, reflectionFromToken, includeInFee, excludeFromFee, isExcludedFromReward, totalFees , excludeFromReward, setSwapAndLiquifyEnabled, isExcludedFromFee should be declared as external to save on gas fees.

Developer Response

Set all recommended function as external

Issue 14

Type	Severity	Location
Gas Optimization	Informational	

Description

`_name`, `_symbol`, `_decimals` could have been declared immutable to save on gas.

Developer Response

We set it as private.

Updated File on GitHub:

https://github.com/FeedPups/FeedPupsContract/blob/main/contract-27082021.sol

Test Results with different versions

https://testnet.bscscan.com/token/0x9123d9f3686b3a20980ba982e48065ed41afba6a (Final Version)

https://testnet.bscscan.com/token/0xa01943b2faa147663cfa370b1273564465a6fcf4

https://testnet.bscscan.com/token/0x71e687158409677227b0cbcfe6aa3d67522b6d93