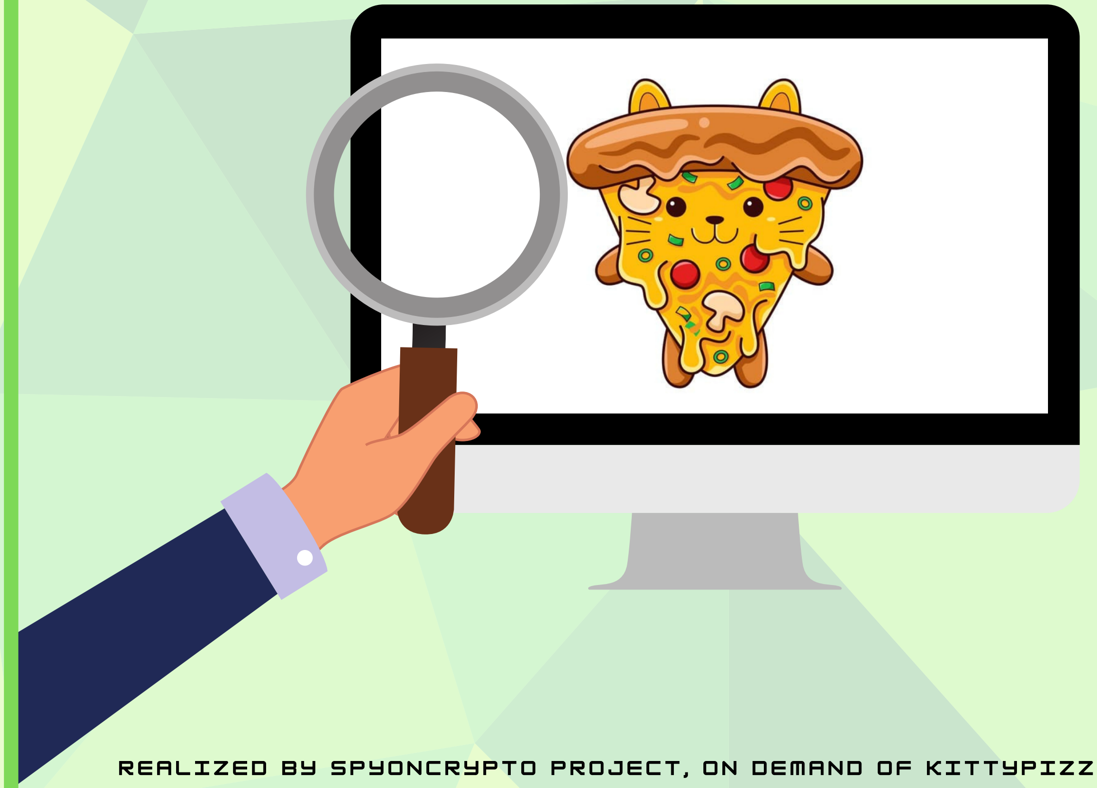
SPY ON CRYPTO

AUDIT KITTYPIZZA



REALIZED BY SPYONCRYPTO PROJECT, ON DEMAND OF KITTYPIZZA TOKEN TEAM

DISCLAIMER

This file is an audit carried out at the request of the interested party.

This report is based on a multitude of analyses and research carried out by our team according to a predefined scheme.

The various steps set out in this file will make it possible to display any vulnerabilities relating to the cybersecurity of the project studied.

These searches are based on the information available to us through the smart contract, but also through information provided by the project developers.

In order to have a better overview of the possible vulnerabilities of this project, the complete reading of this file is recommended.

However, even if this report is available to you, it is only an additional element that can help you in your investigations.

Although a great deal of background work has been done in our investigations, we may have missed some elements, so further research on your part is necessary and advisable.

The conditions mentioned above in the disclaimer are not optional, so if you are not satisfied with them, we strongly urge you to stop reading and analyzing this file and to destroy any copies you have downloaded and/or printed.

These analyses and conclusions are not intended as investment advice. SpyonCrypto is not responsible for any loss of capital, which you are the only owner of.

This report is provided to you as, and without any conditions guaranteed.

SpyonCrypto disclaims any and all liability to the law for any claim or demand by you or any other person for damages.

The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security.

No product code has been reviewed.

SUMMARY

PRESENTATION

KITTYPIZZA is the MEME pump and dump corner deployed on the BSC.

There is no presale or wallet dev, only wallet influencers. It announces a liquidity locked and the contract renounced 10 minutes after launching.

The audit was done upstream.

We based our audit on the information available at the time.

Contract is not yet deployed. On 6 July at 11pm

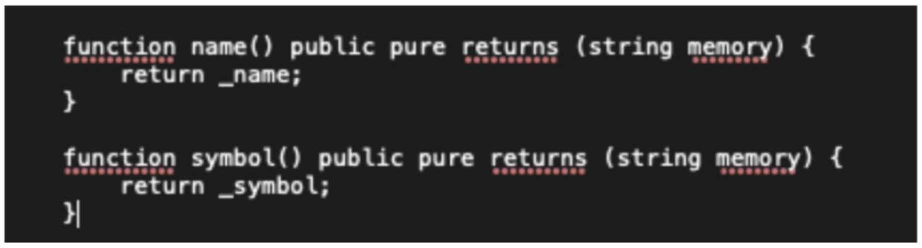
DETECTED VULNERABILITIES

0	7	3

SECURITY ISSUES

MEDIUM SWC-000

Incorrect function "name" state mutability
Function "name" state mutability is considered "pure" by compiler, but should be set to non-payable (default).



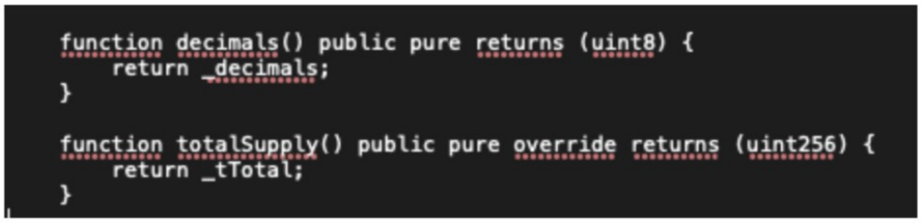
```
function name() public pure returns (string memory) {
    return _name;
}

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

MEDIUM SWC-000

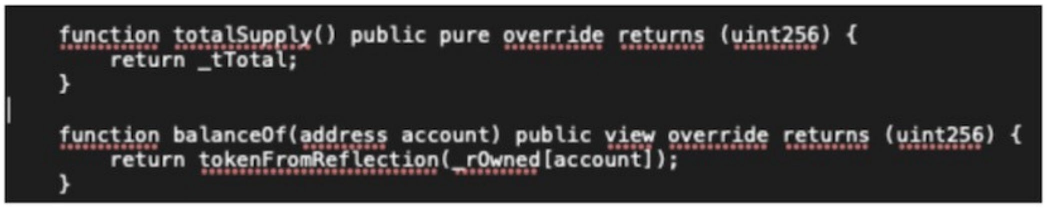
Function could be marked as external.

The function definition of "decimals" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.



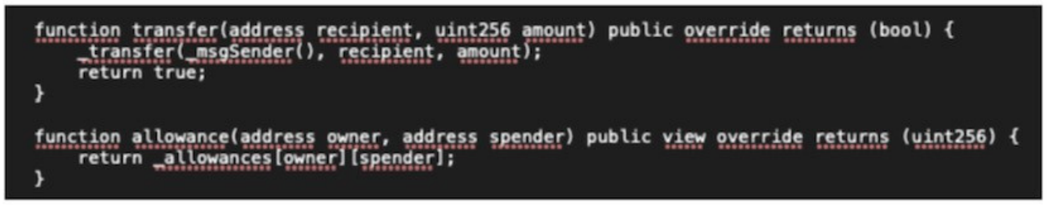
```
function decimals() public pure returns (uint8) {
    return _decimals;
}

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



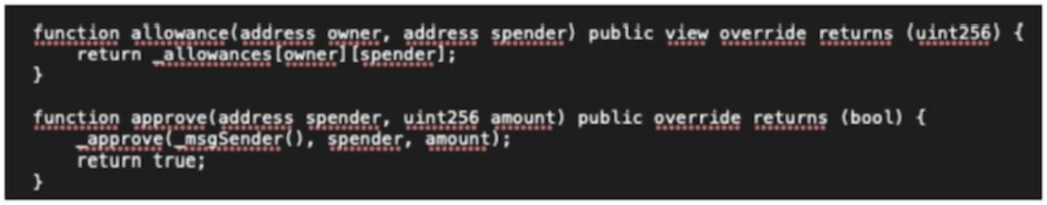
```
    function totalSupply() public pure override returns (uint256) {
        return _tTotal;
    }

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



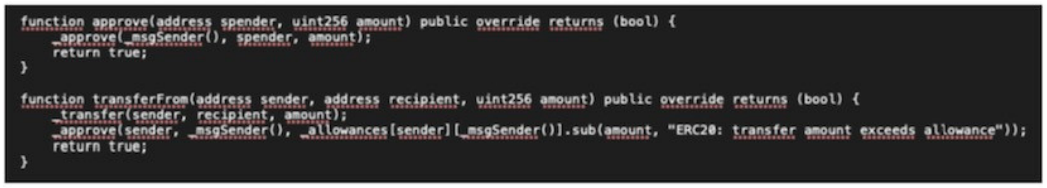
```
function transfer(address recipient, uint256 amount) public override returns (bool) {
    _transfer(_msgSender(), recipient, amount);
    return true;
}

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



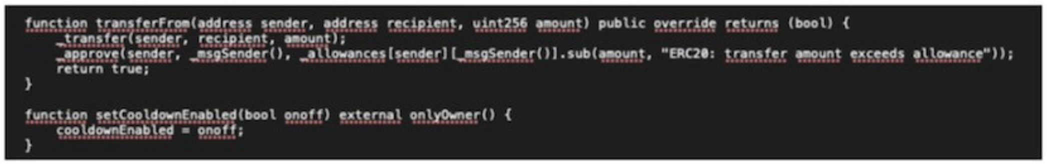
```
function allowance(address owner, address spender) public view override returns (uint256) {
    return _allowances[owner][spender];
}

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



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

function transferFrom(address sender, address recipient, uint256 amount) public override returns (bool) {
    _transfer(sender, recipient, amount);
    _approve(sender, _msgSender(), _allowances[sender][_msgSender()].sub(amount, "ERC20: transfer amount exceeds allowance"));
    return true;
}
```



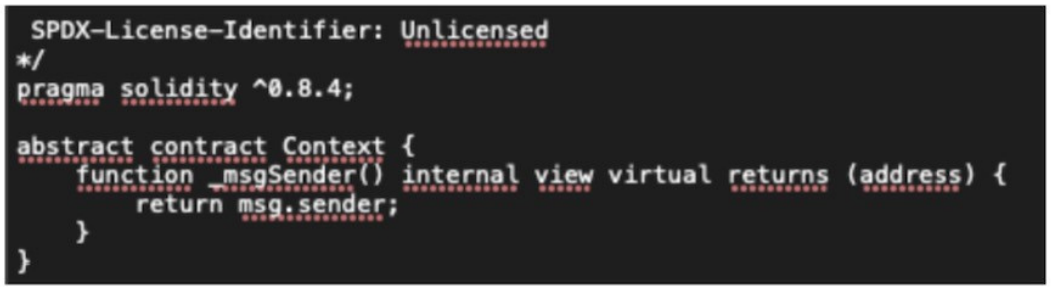
```
function transferFrom(address sender, address recipient, uint256 amount) public override returns (bool) {
    _transfer(sender, recipient, amount);
    _approve(sender, _msgSender(), _allowances[sender][_msgSender()].sub(amount, "ERC20: transfer amount exceeds allowance"));
    return true;
}

function setCooldownEnabled(bool onoff) external onlyOwner() {
    cooldownEnabled = onoff;
}
```

LOW SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.4"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.



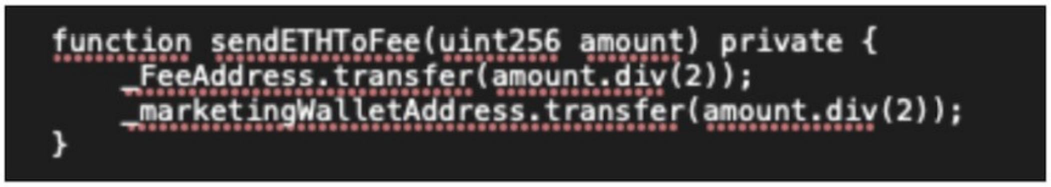
```
 SPDX-License-Identifier: Unlicensed
*/
pragma solidity ^0.8.4;

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

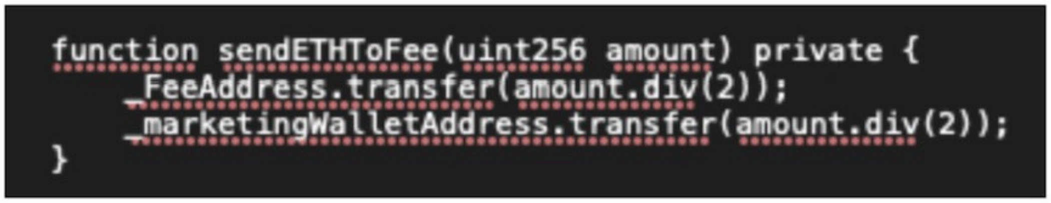
LOW SWC-103

Call with hardcoded gas amount.

The highlighted function call forwards a fixed amount of gas. This is discouraged as the gas cost of EVM instructions may change in the future, which could break this contract's assumptions. If this was done to prevent reentrancy attacks, consider alternative methods such as the checks-effects-interactions pattern or reentrancy locks instead.



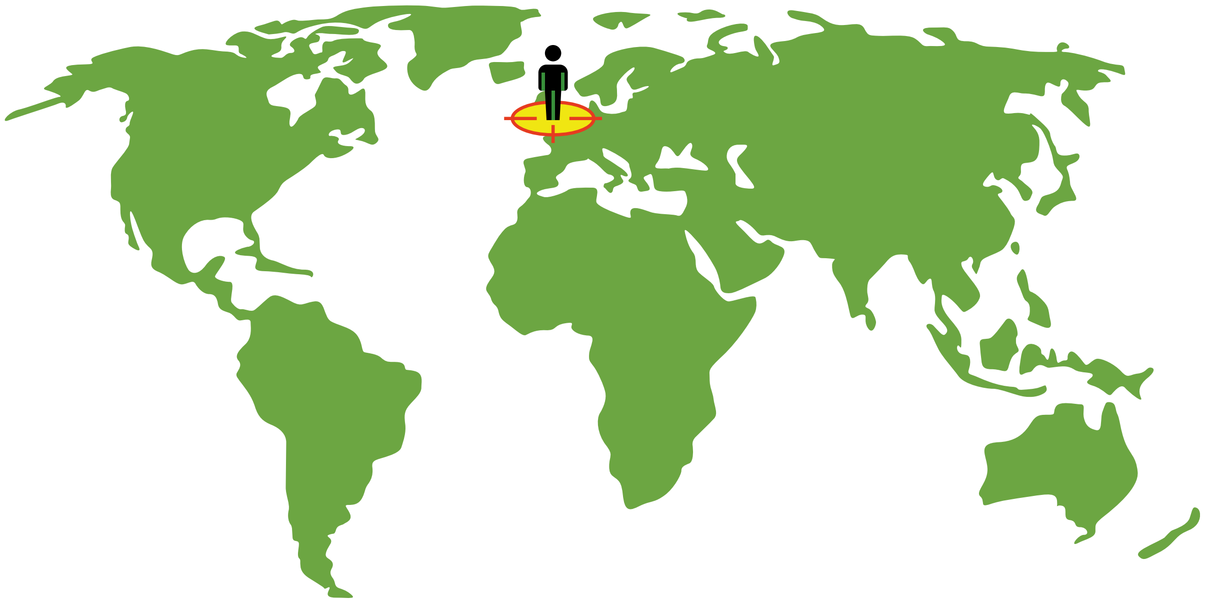
```
function sendETHToFee(uint256 amount) private {
    _FeeAddress.transfer(amount.div(2));
    _marketingWalletAddress.transfer(amount.div(2));
}
```



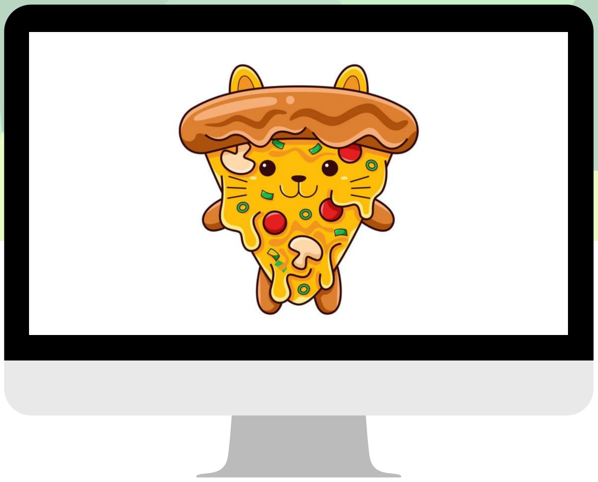
```
function sendETHToFee(uint256 amount) private {
    _FeeAddress.transfer(amount.div(2));
    _marketingWalletAddress.transfer(amount.div(2));
}
```

LOCATION TEAM

TEAM UNITED KINGDOM



SOCIAL MEDIA



https://t.me/kittypizzatoken

https://twitter.com/kittypizza

NOTE AND CONCLUSION

The contract due to KITTYPIZZA is safe and presents only minor risks.

The promises of locked-in liquidity and the renounced of the contract 10 minutes after launching contribute to the security of the investment.

This MEME token is based on the principle of a pump and dump which should be shared by different influencers.

The decision of the devs not to grant themselves a team allocation, in order to focus on marketing.

APPROVED

AUDIT PERFORMED ON 6 JULY 2021 AT 23:00 UTC

SPYON

TG :	SPYONCRYPTO
TG CHAT:	SPYONCRYPTO CHAT
TWITTER:	@spyoncrypto
MAIL:	SOCIAL@SPYONCRYPTO.COM
SITE WEB:	WWW.SPYONCRYPTO.COM