

SMART CONTRACT

Security Audit Report

Customer:	BlackPig Finance
Website:	https://blackpig.finance
Platform:	Binance Smart Chain
Language:	Solidity
Date:	June 11th, 2021

Table of contents

Introduction	4
Project Background	4
Audit Scope	4
Claimed Smart Contract Features	5
Audit Summary	5
Technical Quick Stats	6
Code Quality	7
Documentation	7
Use of Dependencies	7
AS-IS overview	8
Severity Definitions	11
Audit Findings	11
Conclusion	17
Our Methodology	18
Disclaimers	20
Appendix	
• Code Flow Diagram	21
• Slither Report Log	22

THIS IS SECURITY AUDIT REPORT DOCUMENT AND WHICH MAY CONTAIN INFORMATION WHICH IS CONFIDENTIAL. WHICH INCLUDES ANY POTENTIAL VULNERABILITIES AND MALICIOUS CODES WHICH CAN BE USED TO EXPLOIT THE SOFTWARE. THIS MUST BE REFERRED INTERNALLY AND ONLY SHOULD BE MADE AVAILABLE TO PUBLIC AFTER ISSUES ARE RESOLVED.

Introduction

We were contracted by the BlackPig team to perform the Security audit of the BlackPig Token smart contract code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on June 11th, 2021.

The purpose of this audit was to address the following:

- Ensure that all claimed functions exist and function correctly.
- Identify any security vulnerabilities that may be present in the smart contract.

Project Background

BlackPig is a Decentralized Starship meme tokens that grew into a vibrant ecosystem. BlackPig (BPIG) is the central currency for the NFT, BpigSwap, Farms & Staking and other components of its ecosystem.

Audit scope

Name	Code Review and Security Analysis Report for BlackPig Token (BPIG) Smart Contract
Platform	BSC / Solidity
File	BlackPig.sol
Smart Contract Online Code	https://bscscan.com/address/0x368bd050e56ee0984ac5a22665d9f57cd76287da#readContract
File MD5 Hash	2588109EC457C2F1134B3B12D1DCA2B0
Audit Date	June 11th, 2021

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
Name: BlackPig Token	YES, This is valid.
Symbol: BPIG	YES, This is valid.
Decimals: 18	YES, This is valid.
Chain: Binance Smart Chain (BEP-20)	YES, This is valid.
Tax fee: 2% Liquidity fee: 2%	YES, This is valid. Owner can change these fees.
Max Transaction Amount: 1000 Trillions Minimum Tokens Before Swap: 3 Million	YES, This is valid. Owner can change these values.

Audit Summary

According to the standard audit assessment, Customer's solidity smart contract is **Secured**. These contracts also have owner functions (described in the centralization section below), which does not make everything 100% decentralized. Thus, the owner must execute those smart contract functions as per the business plan.

Insecure	Poor secured	Secure	Well-secured
----------	--------------	--------	--------------

You are here



We used various tools like MythX, Slither and Remix IDE. At the same time this finding is based on critical analysis of the manual audit. All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the Audit overview section. General overview is presented in AS-IS section and all identified issues can be found in the Audit overview section.

We found 0 critical, 0 high, 0 medium and 5 low and some very low level issues.

Technical Quick Stats

Main Category	Subcategory	Result
Contract Programming	Solidity version not specified	Passed
	Solidity version too old	Moderated
	Integer overflow/underflow	Passed
	Function input parameters lack of check	Passed
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Moderated
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	Passed
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Moderated
	Other programming issues	Passed
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Other code specification issues	Passed
Gas Optimization	"Out of Gas" attack	Passed
	High consumption 'for/while' loop	Moderated
	High consumption 'storage' storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Passed
	"Short Address" Attack	Passed
	"Double Spend" Attack	Passed

Overall Audit Result: PASSED

Code Quality

This audit scope has 1 smart contract. This smart contract also contains Libraries, Smart contracts inherits and Interfaces. This is a compact and well written contract.

The libraries in the BlackPig Token are part of its logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned a specific address and its properties / methods can be reused many times by other contracts in the BlackPig Token.

The BlackPig team has **not** provided scenario and unit test scripts, which would have helped to determine the integrity of the code in an automated way.

Some code parts are **not well** commented on smart contracts.

Documentation

We were given BlackPig Token smart contract code in the form of a BscScan web link. The hashes of that code are mentioned above in the table.

As mentioned above, some code parts are **not well** commented. So it is difficult to quickly understand the programming flow as well as complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

Another source of information was its official website <https://blackpig.finance/> which provided rich information about the project architecture and tokenomics.

Use of Dependencies

As per our observation, the libraries are used in this smart contract infrastructure that are based on well known industry standard open source projects. And their core code blocks are written well.

Apart from libraries, its functions are used in external smart contract calls.

AS-IS overview

(1) Interface

- (a) IERC20
- (b) IUniswapV2Factory
- (c) IUniswapV2Pair
- (d) IUniswapV2Router01
- (e) IUniswapV2Router02

(2) Inherited contracts

- (a) Context
- (b) Ownable
- (c) IERC20

(3) Usages

- (a) using SafeMath for uint256;
- (b) using Address for address;

(4) Events

- (a) event Transfer(address indexed from, address indexed to, uint256 value);
- (b) event Approval(address indexed owner, address indexed spender, uint256 value);
- (c) event Approval(address indexed owner, address indexed spender, uint value);
- (d) event Transfer(address indexed from, address indexed to, uint value);
- (e) event Burn(address indexed sender, uint amount0, uint amount1, address indexed to);
- (f) event Swap(address indexed sender, uint amount0In, uint amount1In, uint amount0Out, uint amount1Out, address indexed to);
- (g) event Sync(uint112 reserve0, uint112 reserve1);
- (h) event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);
- (i) event RewardLiquidityProviders(uint256 tokenAmount);
- (j) event SwapAndLiquifyEnabledUpdated(bool enabled);
- (k) event SwapAndLiquify(uint256 tokensSwapped, uint256 ethReceived, uint256 tokensIntoLiquidity);

(5) Functions

Sl.	Functions	Type	Observation	Conclusion
1	lockTheSwap	modifier	Passed	No Issue
2	name	read	Passed	No Issue
3	symbol	read	Passed	No Issue
4	decimals	read	Passed	No Issue
5	totalSupply	read	Passed	No Issue
6	balanceOf	read	Passed	No Issue
7	transfer	write	Passed	No Issue
8	allowance	read	Passed	No Issue
9	approve	write	Passed	No Issue
10	transferFrom	write	Passed	No Issue
11	increaseAllowance	write	Passed	No Issue
12	decreaseAllowance	write	Passed	No Issue
13	isExcludedFromReward	read	Passed	No Issue
14	totalFees	read	Passed	No Issue
15	minimumTokensBeforeSwapAmount	read	Passed	No Issue
16	deliver	write	External instead of public	Refer Audit Findings
17	reflectionFromToken	read	External instead of public	Refer Audit Findings
18	tokenFromReflection	read	Passed	No Issue
19	excludeFromReward	write	access only Owner	No Issue
20	changeMinTokenSwap	write	access only Owner	No Issue
21	includeInReward	external	Infinite loop possibility	Refer Audit Findings
22	_approve	write	Passed	No Issue
23	_transfer	write	Passed	No Issue
24	swapAndLiquify	write	Passed	No Issue
25	swapTokensForEth	write	Passed	No Issue
26	addLiquidity	write	Centralized risk in addLiquidity	Refer Audit Findings
27	_tokenTransfer	write	Passed	No Issue
28	_transferStandard	write	Passed	No Issue
29	transferToExcluded	write	Passed	No Issue
30	transferFromExcluded	write	Passed	No Issue
31	transferBothExcluded	write	Passed	No Issue
32	_reflectFee	write	Passed	No Issue
33	getValues	read	Passed	No Issue
34	_getTValues	read	Passed	No Issue
35	_getRValues	write	Passed	No Issue
36	getRate	read	Passed	No Issue
37	_getCurrentSupply	read	Passed	No Issue
38	takeLiquidity	write	Passed	No Issue

39	calculateTaxFee	read	Passed	No Issue
40	calculateLiquidityFee	read	Passed	No Issue
41	removeAllFee	write	Passed	No Issue
42	restoreAllFee	write	Passed	No Issue
43	isExcludedFromFee	read	Passed	No Issue
44	excludeFromFee	write	Missing Events	No Issue
45	includeInFee	write	Missing Events	No Issue
46	setTaxFeePercent	external	Missing Events	No Issue
47	setLiquidityFeePercent	external	Missing Events	No Issue
48	setMaxTxAmount	external	Missing Events	No Issue
49	setNumTokensSellToAddToLiquidity	external	Missing Events	No Issue
50	setSwapAndLiquifyEnabled	write	Missing Events	No Issue
51	transferForMarketingETH	write	Passed	No Issue
52	receive	external	Passed	No Issue
53	owner	read	Passed	No Issue
54	creator	read	Passed	No Issue
55	onlyOwner	modifier	Passed	No Issue
56	onlyCreator	modifier	Passed	No Issue
57	renounceOwnership	write	access only Owner	No Issue
58	transferOwnership	write	access only Owner	No Issue
59	getUnlockTime	read	Passed	No Issue
60	getTime	read	Passed	No Issue
61	lock	write	access only Owner	No Issue
62	unlock	write	Ownership can be reclaimed	Refer Audit Findings
63	msgSender	internal	Passed	No Issue
64	_msgData	internal	Passed	No Issue

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to tokens loss
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored.

Audit Findings

Critical

No critical severity vulnerabilities were found.

High

No high severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

(1) Make variables constant:

```
string private _name = "BlackPig";  
string private _symbol = "BPIG";  
uint8 private _decimals = 18;
```

Following variables will be unchanged. So, please make it constant. It will save some gas.

- name
- symbol
- decimals

Resolution: Declare those variables as constant. Just put a constant keyword.

(2) Infinite loop possibility:

```
function includeInReward(address account) external onlyOwner() {  
    require(!_isExcluded[account], "Account is already excluded");  
    for (uint256 i = 0; i < _excluded.length; i++) {  
        if (_excluded[i] == account) {  
            _excluded[i] = _excluded[_excluded.length - 1];  
            _tOwned[account] = 0;  
            _isExcluded[account] = false;  
            _excluded.pop();  
            break;  
        }  
    }  
}
```

If there are so many excluded wallets, then this logic will fail, as it might hit the block's gas limit. If there are very limited exceptions, then this will work, but will cost more gas.

Resolution: We suggest excluding limited wallets only.

(3) Centralized risk in addLiquidity:

```
function addLiquidity(uint256 tokenAmount, uint256 ethAmount) private {  
    // approve token transfer to cover all possible scenarios  
    _approve(address(this), address(uniswapV2Router), tokenAmount);  
  
    // add the liquidity  
    uniswapV2Router.addLiquidityETH{value: ethAmount}(  
        address(this),  
        tokenAmount,  
        0, // slippage is unavoidable  
        0, // slippage is unavoidable  
        owner(),  
        block.timestamp  
    );  
}
```

In addLiquidityETH function, owner() gets BPIG Tokens from the Pool. At some time, The owner will accumulate significant BPIG tokens. If the _owner is an EOA (Externally Owned Account), mishandling of its private key can have devastating consequences to the project.

Resolution: uniswapV2Router.addLiquidityETH function call to be replaced by address (this), and to restrict the management of the BPIG tokens. This will protect the BPIG tokens from being stolen if the _owner account is compromised.

(4) Possible to gain ownership after renouncing the contract ownership. Owner can renounce ownership and make contract without owner but here is a catch, owner can misuse it by performing the following operations:

- Owner calls the lock function in contract to set the current owner as _previousOwner.
- Owner calls unlock to unlock contract and set _owner = _previousOwner.
- Owner called renounceOwnership to leave the contract without the owner.
- Owner calls unlock to regain ownership.

Resolution: We suggest removing these lock/unlock functions as this seems not serving a great purpose. Otherwise, always renounce ownership first before calling the lock function.

(5) Missing Events: Missing Events log for some functions:

- excludeFromFee
- excludeFromReward
- includeInFee
- includeInReward
- setLiquidityFeePercent
- setMaxTxPercent
- setTaxFeePercent
- setSwapAndLiquifyEnabled

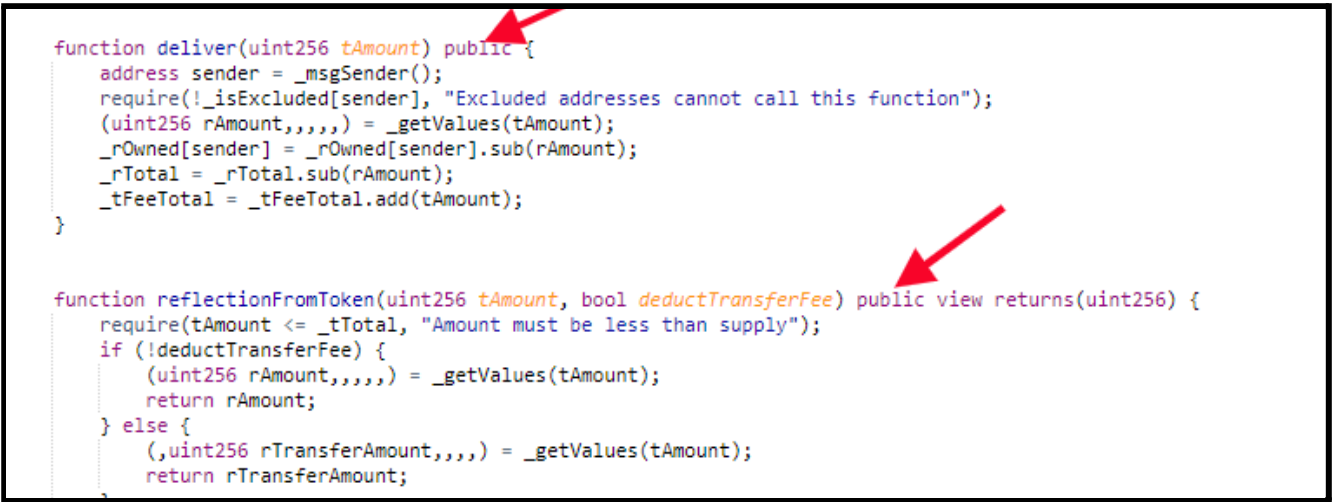
Very Low / Discussion / Best practices:

(1) Use latest solidity version:

```
pragma solidity ^0.6.12;
```

Using the latest solidity will prevent any compiler level bugs.

(2) External instead of public:



```
function deliver(uint256 tAmount) public {
    address sender = _msgSender();
    require(!_isExcluded[sender], "Excluded addresses cannot call this function");
    (uint256 rAmount,,,,) = _getValues(tAmount);
    _rOwned[sender] = _rOwned[sender].sub(rAmount);
    _rTotal = _rTotal.sub(rAmount);
    _tFeeTotal = _tFeeTotal.add(tAmount);
}

function reflectionFromToken(uint256 tAmount, bool deductTransferFee) public view returns(uint256) {
    require(tAmount <= _tTotal, "Amount must be less than supply");
    if (!deductTransferFee) {
        (uint256 rAmount,,,,) = _getValues(tAmount);
        return rAmount;
    } else {
        (,uint256 rTransferAmount,,,,) = _getValues(tAmount);
        return rTransferAmount;
    }
}
```

If any function is not called from inside the smart contract, then it is better to declare it as external instead of public. As it saves some gas as well.

<https://ethereum.stackexchange.com/questions/19380/external-vs-public-best-practices>

Centralization

This smart contract has some functions which can be executed by Admin (Owner) only. If the admin wallet private key would be compromised, then it would create trouble.

Following are Admin functions:

- `excludeFromReward`: Owner can check if the account is excluded.
- `changeMinTokenSwap`: Owner can change the minimum token swap.
- `changeSwap`: Owner can change swap value.
- `includeInReward`: Owner can include rewards.
- `excludeFromFee`: Owner can check exclude Fee.
- `includeInFee`: Owner can check include Fee.
- `setTaxFeePercent`: Owner can set tax Fee Percentage.
- `setLiquidityFeePercent`: Owner can set Liquidity Fee Percentage.
- `setMaxTxAmount`: Owner can set maximum TxAmount.
- `setNumTokensSellToAddToLiquidity`: Owner can set the number of tokens sold to add Liquidity.
- `setSwapAndLiquifyEnabled`: Owner can set swap and Liquify enable.

Conclusion

We were given a contract code. And we have used all possible tests based on given objects as files. We observed some issues in the smart contracts and those are fixed/acknowledged in the smart contracts. **So it is good to go for the production.**

Since possible test cases can be unlimited for such smart contracts protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan everything.

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. Smart Contract's high level description of functionality was presented in the As-is overview section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security state of the reviewed contract, based on standard audit procedure scope is **"Secured"**.

Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review:

In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis:

Our audit techniques included manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's web site to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

Documenting Results:

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyze the feasibility of an attack in a live system.

Suggested Solutions:

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

Disclaimers

EtherAuthority.io Disclaimer

EtherAuthority team has analyzed this smart contract in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

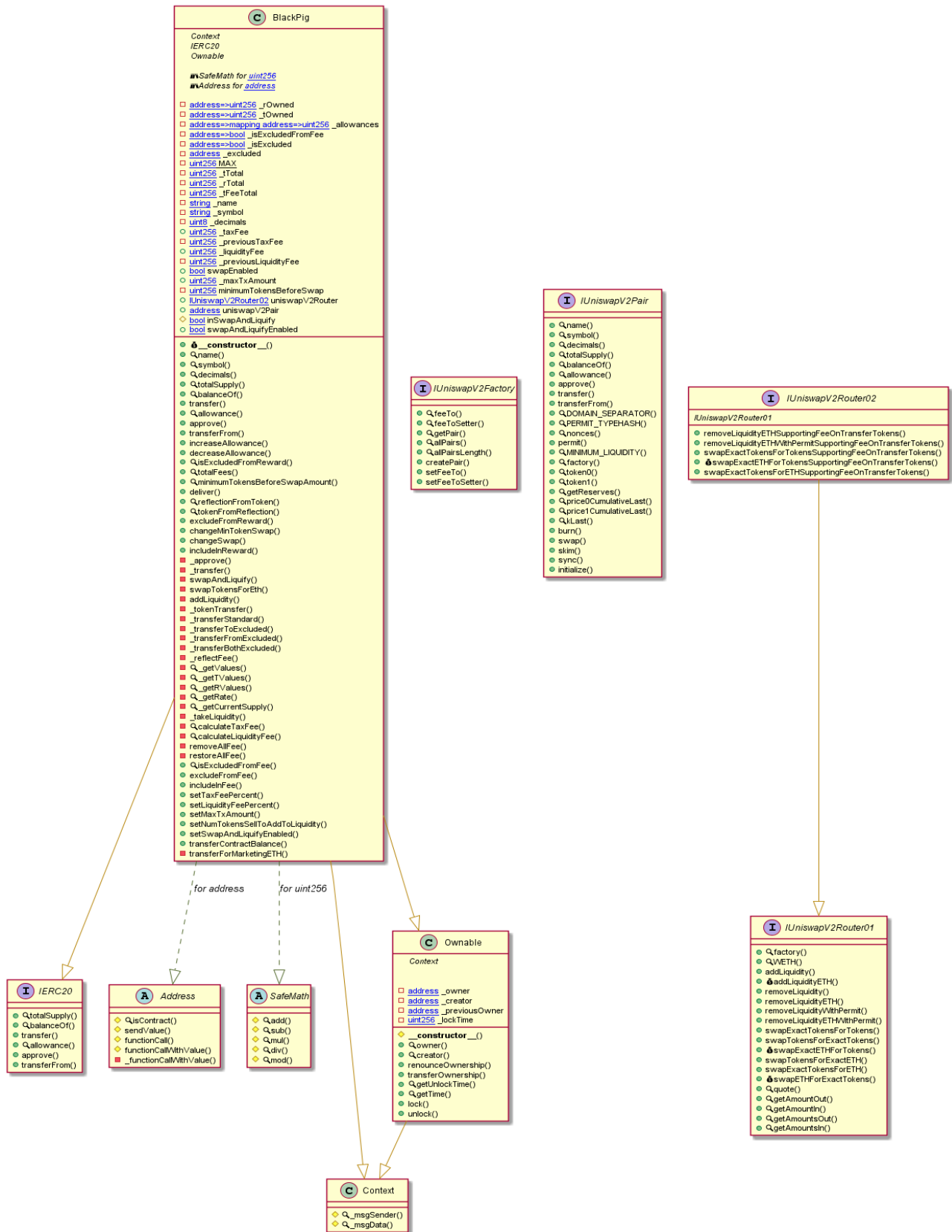
Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only. We also suggest conducting a bug bounty program to confirm the high level of security of this smart contract.

Technical Disclaimer

Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can't guarantee explicit security of the audited smart contracts.

Appendix

Code Flow Diagram - BlackPig (BPIG) Token



This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.

Email: audit@EtherAuthority.io

Slither Results Log

Slither log >> BlackPig.sol

INFO:Detectors:

BlackPig.transferContractBalance(uint256) (BlackPig.sol#878-881) sends eth to arbitrary user

Dangerous calls:

- address(creator()).transfer(amount) (BlackPig.sol#880)

Reference:

<https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations>

INFO:Detectors:

Reentrancy in BlackPig._transfer(address,address,uint256) (BlackPig.sol#593-626):

External calls:

- swapAndLiquify(contractTokenBalance) (BlackPig.sol#614)
- uniswapV2Router.addLiquidityETH{value:

ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (BlackPig.sol#676-683)

-

uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (BlackPig.sol#662-668)

External calls sending eth:

- swapAndLiquify(contractTokenBalance) (BlackPig.sol#614)
- uniswapV2Router.addLiquidityETH{value:

ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (BlackPig.sol#676-683)

State variables written after the call(s):

- _tokenTransfer(from,to,amount,takeFee) (BlackPig.sol#625)
 - _rOwned[address(this)] = _rOwned[address(this)].add(rLiquidity) (BlackPig.sol#812)
 - _rOwned[sender] = _rOwned[sender].sub(rAmount) (BlackPig.sol#737)
 - _rOwned[sender] = _rOwned[sender].sub(rAmount) (BlackPig.sol#728)
 - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (BlackPig.sol#729)
 - _rOwned[sender] = _rOwned[sender].sub(rAmount) (BlackPig.sol#758)
 - _rOwned[sender] = _rOwned[sender].sub(rAmount) (BlackPig.sol#748)
 - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (BlackPig.sol#739)
 - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (BlackPig.sol#749)
 - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (BlackPig.sol#760)
- _tokenTransfer(from,to,amount,takeFee) (BlackPig.sol#625)
 - _rTotal = _rTotal.sub(rFee) (BlackPig.sol#767)
- _tokenTransfer(from,to,amount,takeFee) (BlackPig.sol#625)
 - _tOwned[address(this)] = _tOwned[address(this)].add(tLiquidity) (BlackPig.sol#814)
 - _tOwned[sender] = _tOwned[sender].sub(tAmount) (BlackPig.sol#757)
 - _tOwned[sender] = _tOwned[sender].sub(tAmount) (BlackPig.sol#747)
 - _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (BlackPig.sol#738)
 - _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (BlackPig.sol#759)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities>

INFO:Detectors:

BlackPig.addLiquidity(uint256,uint256) (BlackPig.sol#671-684) ignores return value by

uniswapV2Router.addLiquidityETH{value:

ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (BlackPig.sol#676-683)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return>

INFO:Detectors:

BlackPig.allowance(address,address).owner (BlackPig.sol#492) shadows:

- Ownable.owner() (BlackPig.sol#348-350) (function)

BlackPig._approve(address,address,uint256).owner (BlackPig.sol#585) shadows:

- Ownable.owner() (BlackPig.sol#348-350) (function)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing>

INFO:Detectors:

Reentrancy in BlackPig._transfer(address,address,uint256) (BlackPig.sol#593-626):

External calls:

- swapAndLiquify(contractTokenBalance) (BlackPig.sol#614)

```

        - uniswapV2Router.addLiquidityETH{value:
ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (BlackPig.sol#676-683)
    -
    uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(t
his),block.timestamp) (BlackPig.sol#662-668)
        External calls sending eth:
        - swapAndLiquify(contractTokenBalance) (BlackPig.sol#614)
        - uniswapV2Router.addLiquidityETH{value:
ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (BlackPig.sol#676-683)
        State variables written after the call(s):
        - _tokenTransfer(from,to,amount,takeFee) (BlackPig.sol#625)
        -   _liquidityFee = _previousLiquidityFee (BlackPig.sol#841)
        -   _liquidityFee = 0 (BlackPig.sol#836)
        - _tokenTransfer(from,to,amount,takeFee) (BlackPig.sol#625)
        -   _previousLiquidityFee = _liquidityFee (BlackPig.sol#833)
        - _tokenTransfer(from,to,amount,takeFee) (BlackPig.sol#625)
        -   _previousTaxFee = _taxFee (BlackPig.sol#832)
        - _tokenTransfer(from,to,amount,takeFee) (BlackPig.sol#625)
        -   _tFeeTotal = _tFeeTotal.add(tFee) (BlackPig.sol#768)
        - _tokenTransfer(from,to,amount,takeFee) (BlackPig.sol#625)
        -   _taxFee = _previousTaxFee (BlackPig.sol#840)
        -   _taxFee = 0 (BlackPig.sol#835)
    Reentrancy in BlackPig.constructor() (BlackPig.sol#451-464):
        External calls:
        - uniswapV2Pair =
IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.WETH())
(BlackPig.sol#455-456)
        State variables written after the call(s):
        - _isExcludedFromFee[owner()] = true (BlackPig.sol#460)
        - _isExcludedFromFee[address(this)] = true (BlackPig.sol#461)
        - uniswapV2Router = _uniswapV2Router (BlackPig.sol#458)
    Reentrancy in BlackPig.swapAndLiquify(uint256) (BlackPig.sol#628-651):
        External calls:
        - swapTokensForEth(marketingQuota) (BlackPig.sol#637)
    -
    uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(t
his),block.timestamp) (BlackPig.sol#662-668)
        - swapTokensForEth(firstHalf) (BlackPig.sol#642)
    -
    uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(t
his),block.timestamp) (BlackPig.sol#662-668)
        State variables written after the call(s):
        - swapTokensForEth(firstHalf) (BlackPig.sol#642)
        -   _allowances[owner()][spender] = amount (BlackPig.sol#589)
    Reentrancy in BlackPig.swapAndLiquify(uint256) (BlackPig.sol#628-651):
        External calls:
        - swapTokensForEth(marketingQuota) (BlackPig.sol#637)
    -
    uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(t
his),block.timestamp) (BlackPig.sol#662-668)
        - swapTokensForEth(firstHalf) (BlackPig.sol#642)
    -
    uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(t
his),block.timestamp) (BlackPig.sol#662-668)
        - addLiquidity(remainingHalf,newBalance) (BlackPig.sol#648)
        -   uniswapV2Router.addLiquidityETH{value:
ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (BlackPig.sol#676-683)
        External calls sending eth:
        - addLiquidity(remainingHalf,newBalance) (BlackPig.sol#648)
        -   uniswapV2Router.addLiquidityETH{value:
ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (BlackPig.sol#676-683)
        State variables written after the call(s):
        - addLiquidity(remainingHalf,newBalance) (BlackPig.sol#648)
        -   _allowances[owner()][spender] = amount (BlackPig.sol#589)
    Reentrancy in BlackPig.transferFrom(address,address,uint256) (BlackPig.sol#501-505):

```


External calls:

- `_transfer(sender,recipient,amount)` (BlackPig.sol#502)
- `uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp)` (BlackPig.sol#676-683)

-

`uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp)` (BlackPig.sol#662-668)

External calls sending eth:

- `_transfer(sender,recipient,amount)` (BlackPig.sol#502)
- `uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp)` (BlackPig.sol#676-683)

State variables written after the call(s):

- `_approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount exceeds allowance))` (BlackPig.sol#503)
- `_allowances[owner][spender] = amount` (BlackPig.sol#589)

Reference: <https://github.com/cryptic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2>

INFO:Detectors:

Reentrancy in BlackPig.`_transfer(address,address,uint256)` (BlackPig.sol#593-626):

External calls:

- `swapAndLiquify(contractTokenBalance)` (BlackPig.sol#614)
- `uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp)` (BlackPig.sol#676-683)

-

`uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp)` (BlackPig.sol#662-668)

External calls sending eth:

- `swapAndLiquify(contractTokenBalance)` (BlackPig.sol#614)
- `uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp)` (BlackPig.sol#676-683)

Event emitted after the call(s):

- `Transfer(sender,recipient,tTransferAmount)` (BlackPig.sol#732)
- `_tokenTransfer(from,to,amount,takeFee)` (BlackPig.sol#625)
- `Transfer(sender,recipient,tTransferAmount)` (BlackPig.sol#752)
- `_tokenTransfer(from,to,amount,takeFee)` (BlackPig.sol#625)
- `Transfer(sender,recipient,tTransferAmount)` (BlackPig.sol#742)
- `_tokenTransfer(from,to,amount,takeFee)` (BlackPig.sol#625)
- `Transfer(sender,recipient,tTransferAmount)` (BlackPig.sol#763)
- `_tokenTransfer(from,to,amount,takeFee)` (BlackPig.sol#625)

Reentrancy in BlackPig.`constructor()` (BlackPig.sol#451-464):

External calls:

- `uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.WETH())` (BlackPig.sol#455-456)

Event emitted after the call(s):

- `Transfer(address(0),_msgSender(),_tTotal)` (BlackPig.sol#463)

Reentrancy in BlackPig.`swapAndLiquify(uint256)` (BlackPig.sol#628-651):

External calls:

- `swapTokensForEth(marketingQuota)` (BlackPig.sol#637)

-

`uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp)` (BlackPig.sol#662-668)

- `swapTokensForEth(firstHalf)` (BlackPig.sol#642)

-

`uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp)` (BlackPig.sol#662-668)

Event emitted after the call(s):

- `Approval(owner,spender,amount)` (BlackPig.sol#590)
- `swapTokensForEth(firstHalf)` (BlackPig.sol#642)

Reentrancy in BlackPig.`swapAndLiquify(uint256)` (BlackPig.sol#628-651):

External calls:

- `swapTokensForEth(marketingQuota)` (BlackPig.sol#637)

-

`uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp)` (BlackPig.sol#662-668)

- `swapTokensForEth(firstHalf)` (BlackPig.sol#642)

uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (BlackPig.sol#662-668)

- addLiquidity(remainingHalf,newBalance) (BlackPig.sol#648)
- uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (BlackPig.sol#676-683)

External calls sending eth:

- addLiquidity(remainingHalf,newBalance) (BlackPig.sol#648)
- uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (BlackPig.sol#676-683)

Event emitted after the call(s):

- Approval(owner,spender,amount) (BlackPig.sol#590)
- addLiquidity(remainingHalf,newBalance) (BlackPig.sol#648)
- SwapAndLiquify(firstHalf,newBalance,remainingHalf) (BlackPig.sol#650)

Reentrancy in BlackPig.transferFrom(address,address,uint256) (BlackPig.sol#501-505):

External calls:

- _transfer(sender,recipient,amount) (BlackPig.sol#502)
- uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (BlackPig.sol#676-683)

uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (BlackPig.sol#662-668)

External calls sending eth:

- _transfer(sender,recipient,amount) (BlackPig.sol#502)
- uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (BlackPig.sol#676-683)

Event emitted after the call(s):

- Approval(owner,spender,amount) (BlackPig.sol#590)
- _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount exceeds allowance)) (BlackPig.sol#503)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3>

INFO:Detectors:

Ownable.unlock() (BlackPig.sol#392-397) uses timestamp for comparisons

Dangerous comparisons:

- require(bool,string)(now > _lockTime,Contract is locked until 7 days) (BlackPig.sol#394)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp>

INFO:Detectors:

Address.isContract(address) (BlackPig.sol#210-219) uses assembly

- INLINE ASM (BlackPig.sol#217)

Address._functionCallWithValue(address,bytes,uint256,string) (BlackPig.sol#247-264) uses assembly

- INLINE ASM (BlackPig.sol#256-259)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage>

INFO:Detectors:

Address._functionCallWithValue(address,bytes,uint256,string) (BlackPig.sol#247-264) is never used and should be removed

Address.functionCall(address,bytes) (BlackPig.sol#230-232) is never used and should be removed

Address.functionCall(address,bytes,string) (BlackPig.sol#234-236) is never used and should be removed

Address.functionCallWithValue(address,bytes,uint256) (BlackPig.sol#238-240) is never used and should be removed

Address.functionCallWithValue(address,bytes,uint256,string) (BlackPig.sol#242-245) is never used and should be removed

Address.isContract(address) (BlackPig.sol#210-219) is never used and should be removed

Address.sendValue(address,uint256) (BlackPig.sol#221-227) is never used and should be removed

BlackPig.transferForMarketingETH(address,uint256) (BlackPig.sol#883-885) is never used and should be removed

Context._msgData() (BlackPig.sol#326-329) is never used and should be removed

SafeMath.mod(uint256,uint256) (BlackPig.sol#311-313) is never used and should be removed

SafeMath.mod(uint256,uint256,string) (BlackPig.sol#315-318) is never used and should be removed

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code>

INFO:Detectors:

BlackPig._rTotal (BlackPig.sol#415) is set pre-construction with a non-constant function or state variable:

- (MAX - (MAX % _rTotal))

BlackPig._previousTaxFee (BlackPig.sol#425) is set pre-construction with a non-constant function or state variable:

- _taxFee

BlackPig._previousLiquidityFee (BlackPig.sol#428) is set pre-construction with a non-constant function or state variable:

- _liquidityFee

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-state-variables>

INFO:Detectors:

Low level call in Address.sendValue(address,uint256) (BlackPig.sol#221-227):

- (success) = recipient.call{value: amount}() (BlackPig.sol#225)

Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (BlackPig.sol#247-264):

- (success,returndata) = target.call{value: weiValue}(data) (BlackPig.sol#250)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls>

INFO:Detectors:

Function IUniswapV2Pair.DOMAIN_SEPARATOR() (BlackPig.sol#46) is not in mixedCase

Function IUniswapV2Pair.PERMIT_TYPEHASH() (BlackPig.sol#47) is not in mixedCase

Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (BlackPig.sol#56) is not in mixedCase

Function IUniswapV2Router01.WETH() (BlackPig.sol#75) is not in mixedCase

Parameter BlackPig.changeMinTokenSwap(uint256)._val (BlackPig.sol#566) is not in mixedCase

Parameter BlackPig.changeSwap(bool)._val (BlackPig.sol#569) is not in mixedCase

Parameter BlackPig.calculateTaxFee(uint256)._amount (BlackPig.sol#817) is not in mixedCase

Parameter BlackPig.calculateLiquidityFee(uint256)._amount (BlackPig.sol#823) is not in mixedCase

Parameter BlackPig.setNumTokensSellToAddToLiquidity(uint256)._minimumTokensBeforeSwap (BlackPig.sol#868) is not in mixedCase

Parameter BlackPig.setSwapAndLiquifyEnabled(bool)._enabled (BlackPig.sol#873) is not in mixedCase

Variable BlackPig._taxFee (BlackPig.sol#424) is not in mixedCase

Variable BlackPig._liquidityFee (BlackPig.sol#427) is not in mixedCase

Variable BlackPig._maxTxAmount (BlackPig.sol#432) is not in mixedCase

Reference:

<https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions>

INFO:Detectors:

Redundant expression "this (BlackPig.sol#327)" inContext (BlackPig.sol#321-330)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements>

INFO:Detectors:

Variable

IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amount
tADesired (BlackPig.sol#80) is too similar to

IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amount
tBDesired (BlackPig.sol#81)

Variable BlackPig._transferBothExcluded(address,address,uint256).rTransferAmount (BlackPig.sol#756) is
too similar to BlackPig._transferBothExcluded(address,address,uint256).tTransferAmount (BlackPig.sol#756)

Variable BlackPig._transferBothExcluded(address,address,uint256).rTransferAmount (BlackPig.sol#756) is
too similar to BlackPig._getTValues(uint256).tTransferAmount (BlackPig.sol#780)

Variable BlackPig._transferStandard(address,address,uint256).rTransferAmount (BlackPig.sol#727) is too
similar to BlackPig._transferStandard(address,address,uint256).tTransferAmount (BlackPig.sol#727)

Variable BlackPig._transferBothExcluded(address,address,uint256).rTransferAmount (BlackPig.sol#756) is
too similar to BlackPig._transferStandard(address,address,uint256).tTransferAmount (BlackPig.sol#727)

Variable BlackPig._transferFromExcluded(address,address,uint256).rTransferAmount (BlackPig.sol#746) is
too similar to BlackPig._getTValues(uint256).tTransferAmount (BlackPig.sol#780)

Variable BlackPig._transferToExcluded(address,address,uint256).rTransferAmount (BlackPig.sol#736) is too
similar to BlackPig._getTValues(uint256).tTransferAmount (BlackPig.sol#780)

Variable BlackPig._getValues(uint256).rTransferAmount (BlackPig.sol#773) is too similar to
BlackPig._getTValues(uint256).tTransferAmount (BlackPig.sol#780)

Variable BlackPig._transferToExcluded(address,address,uint256).rTransferAmount (BlackPig.sol#736) is too
similar to BlackPig._transferBothExcluded(address,address,uint256).tTransferAmount (BlackPig.sol#756)

Variable BlackPig._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (BlackPig.sol#788) is too
similar to BlackPig._getTValues(uint256).tTransferAmount (BlackPig.sol#780)

Variable BlackPig._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (BlackPig.sol#788) is too
similar to BlackPig._transferBothExcluded(address,address,uint256).tTransferAmount (BlackPig.sol#756)

Variable BlackPig.reflectionFromToken(uint256,bool).rTransferAmount (BlackPig.sol#545) is too similar to
BlackPig._getTValues(uint256).tTransferAmount (BlackPig.sol#780)

Variable BlackPig._transferStandard(address,address,uint256).rTransferAmount (BlackPig.sol#727) is too
similar to BlackPig._getTValues(uint256).tTransferAmount (BlackPig.sol#780)

Variable BlackPig._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (BlackPig.sol#788) is too
similar to BlackPig._getValues(uint256).tTransferAmount (BlackPig.sol#772)

Variable BlackPig._transferToExcluded(address,address,uint256).rTransferAmount (BlackPig.sol#736) is too
similar to BlackPig._transferToExcluded(address,address,uint256).tTransferAmount (BlackPig.sol#736)

Variable BlackPig._transferBothExcluded(address,address,uint256).rTransferAmount (BlackPig.sol#756) is too similar to BlackPig._getValues(uint256).tTransferAmount (BlackPig.sol#772)

Variable BlackPig._transferToExcluded(address,address,uint256).rTransferAmount (BlackPig.sol#736) is too similar to BlackPig._getValues(uint256).tTransferAmount (BlackPig.sol#772)

Variable BlackPig._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (BlackPig.sol#788) is too similar to BlackPig._transferStandard(address,address,uint256).tTransferAmount (BlackPig.sol#727)

Variable BlackPig._transferToExcluded(address,address,uint256).rTransferAmount (BlackPig.sol#736) is too similar to BlackPig._transferStandard(address,address,uint256).tTransferAmount (BlackPig.sol#727)

Variable BlackPig._transferToExcluded(address,address,uint256).rTransferAmount (BlackPig.sol#736) is too similar to BlackPig._transferFromExcluded(address,address,uint256).tTransferAmount (BlackPig.sol#746)

Variable BlackPig.reflectionFromToken(uint256,bool).rTransferAmount (BlackPig.sol#545) is too similar to BlackPig._transferToExcluded(address,address,uint256).tTransferAmount (BlackPig.sol#736)

Variable BlackPig.reflectionFromToken(uint256,bool).rTransferAmount (BlackPig.sol#545) is too similar to BlackPig._transferBothExcluded(address,address,uint256).tTransferAmount (BlackPig.sol#756)

Variable BlackPig._getValues(uint256).rTransferAmount (BlackPig.sol#773) is too similar to BlackPig._transferBothExcluded(address,address,uint256).tTransferAmount (BlackPig.sol#756)

Variable BlackPig._transferFromExcluded(address,address,uint256).rTransferAmount (BlackPig.sol#746) is too similar to BlackPig._transferStandard(address,address,uint256).tTransferAmount (BlackPig.sol#727)

Variable BlackPig._transferFromExcluded(address,address,uint256).rTransferAmount (BlackPig.sol#746) is too similar to BlackPig._transferBothExcluded(address,address,uint256).tTransferAmount (BlackPig.sol#756)

Variable BlackPig._transferStandard(address,address,uint256).rTransferAmount (BlackPig.sol#727) is too similar to BlackPig._transferFromExcluded(address,address,uint256).tTransferAmount (BlackPig.sol#746)

Variable BlackPig.reflectionFromToken(uint256,bool).rTransferAmount (BlackPig.sol#545) is too similar to BlackPig._getValues(uint256).tTransferAmount (BlackPig.sol#772)

Variable BlackPig.reflectionFromToken(uint256,bool).rTransferAmount (BlackPig.sol#545) is too similar to BlackPig._transferFromExcluded(address,address,uint256).tTransferAmount (BlackPig.sol#746)

Variable BlackPig._transferStandard(address,address,uint256).rTransferAmount (BlackPig.sol#727) is too similar to BlackPig._getValues(uint256).tTransferAmount (BlackPig.sol#772)

Variable BlackPig._transferStandard(address,address,uint256).rTransferAmount (BlackPig.sol#727) is too similar to BlackPig._transferBothExcluded(address,address,uint256).tTransferAmount (BlackPig.sol#756)

Variable BlackPig._transferStandard(address,address,uint256).rTransferAmount (BlackPig.sol#727) is too similar to BlackPig._transferToExcluded(address,address,uint256).tTransferAmount (BlackPig.sol#736)

Variable BlackPig._transferFromExcluded(address,address,uint256).rTransferAmount (BlackPig.sol#746) is too similar to BlackPig._getValues(uint256).tTransferAmount (BlackPig.sol#772)

Variable BlackPig._getValues(uint256).rTransferAmount (BlackPig.sol#773) is too similar to BlackPig._transferFromExcluded(address,address,uint256).tTransferAmount (BlackPig.sol#746)

Variable BlackPig._transferBothExcluded(address,address,uint256).rTransferAmount (BlackPig.sol#756) is too similar to BlackPig._transferToExcluded(address,address,uint256).tTransferAmount (BlackPig.sol#736)

Variable BlackPig._getValues(uint256).rTransferAmount (BlackPig.sol#773) is too similar to BlackPig._transferStandard(address,address,uint256).tTransferAmount (BlackPig.sol#727)

Variable BlackPig.reflectionFromToken(uint256,bool).rTransferAmount (BlackPig.sol#545) is too similar to BlackPig._transferStandard(address,address,uint256).tTransferAmount (BlackPig.sol#727)

Variable BlackPig._transferFromExcluded(address,address,uint256).rTransferAmount (BlackPig.sol#746) is too similar to BlackPig._transferToExcluded(address,address,uint256).tTransferAmount (BlackPig.sol#736)

Variable BlackPig._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (BlackPig.sol#788) is too similar to BlackPig._transferToExcluded(address,address,uint256).tTransferAmount (BlackPig.sol#736)

Variable BlackPig._getValues(uint256).rTransferAmount (BlackPig.sol#773) is too similar to BlackPig._transferToExcluded(address,address,uint256).tTransferAmount (BlackPig.sol#736)

Variable BlackPig._transferBothExcluded(address,address,uint256).rTransferAmount (BlackPig.sol#756) is too similar to BlackPig._transferFromExcluded(address,address,uint256).tTransferAmount (BlackPig.sol#746)

Variable BlackPig._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (BlackPig.sol#788) is too similar to BlackPig._transferFromExcluded(address,address,uint256).tTransferAmount (BlackPig.sol#746)

Variable BlackPig._getValues(uint256).rTransferAmount (BlackPig.sol#773) is too similar to BlackPig._getValues(uint256).tTransferAmount (BlackPig.sol#772)

Variable BlackPig._transferFromExcluded(address,address,uint256).rTransferAmount (BlackPig.sol#746) is too similar to BlackPig._transferFromExcluded(address,address,uint256).tTransferAmount (BlackPig.sol#746)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar>

INFO:Detectors:

BlackPig.slitherConstructorVariables() (BlackPig.sol#400-890) uses literals with too many digits:

- _tTotal = 10000000000000000 * 10 ** 18 (BlackPig.sol#414)

BlackPig.slitherConstructorVariables() (BlackPig.sol#400-890) uses literals with too many digits:

- _maxTxAmount = 10000000000000000 * 10 ** 18 (BlackPig.sol#432)

BlackPig.slitherConstructorVariables() (BlackPig.sol#400-890) uses literals with too many digits:

- minimumTokensBeforeSwap = 3000000 * 10 ** 18 (BlackPig.sol#433)

Reference: <https://github.com/cryptic/slither/wiki/Detector-Documentation#too-many-digits>

INFO:Detectors:

BlackPig._decimals (BlackPig.sol#421) should be constant

BlackPig._name (BlackPig.sol#419) should be constant

BlackPig._symbol (BlackPig.sol#420) should be constant

BlackPig._tTotal (BlackPig.sol#414) should be constant

Reference:

<https://github.com/cryptic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant>

INFO:Detectors:

renounceOwnership() should be declared external:

- Ownable.renounceOwnership() (BlackPig.sol#366-369)

transferOwnership(address) should be declared external:

- Ownable.transferOwnership(address) (BlackPig.sol#371-375)

getUnlockTime() should be declared external:

- Ownable.getUnlockTime() (BlackPig.sol#377-379)

getTime() should be declared external:

- Ownable.getTime() (BlackPig.sol#381-383)

lock(uint256) should be declared external:

- Ownable.lock(uint256) (BlackPig.sol#385-390)

unlock() should be declared external:

- Ownable.unlock() (BlackPig.sol#392-397)

name() should be declared external:

- BlackPig.name() (BlackPig.sol#466-468)

symbol() should be declared external:

- BlackPig.symbol() (BlackPig.sol#470-472)

decimals() should be declared external:

- BlackPig.decimals() (BlackPig.sol#474-476)

totalSupply() should be declared external:

- BlackPig.totalSupply() (BlackPig.sol#478-480)

transfer(address,uint256) should be declared external:

- BlackPig.transfer(address,uint256) (BlackPig.sol#487-490)

allowance(address,address) should be declared external:

- BlackPig.allowance(address,address) (BlackPig.sol#492-494)

approve(address,uint256) should be declared external:

- BlackPig.approve(address,uint256) (BlackPig.sol#496-499)

transferFrom(address,address,uint256) should be declared external:

- BlackPig.transferFrom(address,address,uint256) (BlackPig.sol#501-505)

increaseAllowance(address,uint256) should be declared external:

- BlackPig.increaseAllowance(address,uint256) (BlackPig.sol#507-510)

decreaseAllowance(address,uint256) should be declared external:

- BlackPig.decreaseAllowance(address,uint256) (BlackPig.sol#512-515)

isExcludedFromReward(address) should be declared external:

- BlackPig.isExcludedFromReward(address) (BlackPig.sol#517-519)

totalFees() should be declared external:

- BlackPig.totalFees() (BlackPig.sol#521-523)

minimumTokensBeforeSwapAmount() should be declared external:

- BlackPig.minimumTokensBeforeSwapAmount() (BlackPig.sol#525-527)

deliver(uint256) should be declared external:

- BlackPig.deliver(uint256) (BlackPig.sol#529-536)

reflectionFromToken(uint256,bool) should be declared external:

- BlackPig.reflectionFromToken(uint256,bool) (BlackPig.sol#539-548)

excludeFromReward(address) should be declared external:

- BlackPig.excludeFromReward(address) (BlackPig.sol#556-564)

changeMinTokenSwap(uint256) should be declared external:

- BlackPig.changeMinTokenSwap(uint256) (BlackPig.sol#566-568)

changeSwap(bool) should be declared external:

- BlackPig.changeSwap(bool) (BlackPig.sol#569-571)

isExcludedFromFee(address) should be declared external:

- BlackPig.isExcludedFromFee(address) (BlackPig.sol#844-846)

excludeFromFee(address) should be declared external:

- BlackPig.excludeFromFee(address) (BlackPig.sol#848-850)

includeInFee(address) should be declared external:

- BlackPig.includeInFee(address) (BlackPig.sol#852-854)

setSwapAndLiquifyEnabled(bool) should be declared external:



This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.

Email: audit@EtherAuthority.io