

# SMART CONTRACT

---

## Security Audit Report

Customer:	XPSToken
Platform:	Binance Smart Chain
Language:	Solidity
Date:	July 21st, 2021

# Table of contents

Introduction .....	4
Project Background .....	4
Audit Scope .....	4
Claimed Smart Contract Features .....	5
Audit Summary .....	6
Technical Quick Stats .....	7
Code Quality .....	8
Documentation .....	8
Use of Dependencies .....	8
AS-IS overview .....	9
Severity Definitions .....	12
Audit Findings .....	12
Conclusion .....	24
Our Methodology .....	25
Disclaimers .....	27
Appendix	
• Code Flow Diagram .....	28
• Slither Report Log .....	29

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

EtherAuthority was contracted by the XPSToken team to perform the Security audit of the XPSToken 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 July 21st, 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

XPS Token (XPOSE) BEP-20 Token in Binance Smart Chain(BSC). The token is implemented as an BEP20 smart contract. XPS Token is a cryptocurrency on the Binance Smart Chain (BSC).

## Audit scope

<b>Name</b>	<b>Code Review and Security Analysis Report for XPSToken Smart Contract</b>
<b>Platform</b>	<b>BSC / Solidity</b>
<b>File</b>	XPSToken.sol
<b>Smart Contract Online Code</b>	<a href="https://testnet.bscscan.com/address/0x3eE824E1758315Fd0B8372762af992B1d7B255a5#code">https://testnet.bscscan.com/address/0x3eE824E1758315Fd0B8372762af992B1d7B255a5#code</a>
<b>File MD5 Hash</b>	155F1335B04B9C28A7E4E9934780B750
<b>Audit Date</b>	July 21st, 2021
<b>Updated File MD5 Hash</b>	ED74C743CCC9C872030B33DDE9EC41BD
<b>Revised Audit Date</b>	July 28th, 2021

## Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
Name: XPS Token	<b>YES, This is valid.</b>
Symbol: XPOSE	<b>YES, This is valid.</b>
Decimals: 9	<b>YES, This is valid.</b>
The fee collected on all transactions need to be allocated as follows: <ul style="list-style-type: none"><li>• CommonFee: 5%</li><li>• Special Fee: 10%</li></ul>	<b>YES, This is valid. The smart contract multisig wallets can change this fee.</b>
<ul style="list-style-type: none"><li>• Locked Tokens For Team: 150000000000</li></ul>	<b>YES, This is valid.</b>
<ul style="list-style-type: none"><li>• From Supply For Team: 15%</li><li>• Release First Step: 60%</li><li>• Release Second Step: 40%</li></ul>	<b>YES, This is valid.</b>
<ul style="list-style-type: none"><li>• Community Trigger Amount: 5000 tokens</li><li>• Liquidity Trigger Amount: 5000 tokens</li><li>• Marketing Trigger Amount: 5000 tokens</li></ul>	<b>YES, This is valid.</b>
<ul style="list-style-type: none"><li>• The lockTheSwap can access the swapAndLiquify.</li><li>• The lockTheSwapCommunity can access the swapAndCommunity.</li><li>• The lockTheSwapMarketing can access the swapAndMarketing.</li></ul>	<b>YES, This is valid.</b>

## Audit Summary

According to the standard audit assessment, Customer's solidity smart contracts are **"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.



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. These issues are fixed/acknowledged in the revised smart contract code.**

**Investors Advice:** Technical audit of the smart contract does not guarantee the ethical nature of the project. Any owner controlled functions should be executed by the owner with responsibility. All investors/users are advised to do their due diligence before investing in the project.

## 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	Moderated
	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	Passed
	Features claimed	Passed
	Other programming issues	Fixed
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Moderated
	Other code specification issues	Passed
Gas Optimization	"Out of Gas" Issue	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. These are compact and well written contracts.

The libraries in XPSToken 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 XPSToken token.

The XPSToken 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 XPSToken smart contracts 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.

## 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

## XPSToken.sol

### (1) Interface

- (a) IPancakeFactory
- (b) IPancakePair
- (c) IPancakeRouter01
- (d) IPancakeRouter02
- (e) IERC20
- (f) IERC20Metadata

### (2) Inherited contracts

- (a) Context
- (b) Ownable
- (c) IERC20
- (d) ERC20

### (3) Usages

- (a) using SafeMath for uint256;

### (4) Events

- (a) event SwapAndLiquify(uint256 tokensSwapped,uint256 ethReceived, uint256 tokensIntoLiquidity);
- (b) event SwapAndCommunity(uint256 tokensSwapped,uint256 ethReceived);
- (c) event SwapAndMarketing(uint256 tokensSwapped, uint256 ethReceived);
- (d) event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);

### (5) Functions

Sl.	Functions	Type	Observation	Conclusion
1	lockTheSwap	modifier	Passed	No Issue
2	lockTheSwapCommunity	modifier	Passed	No Issue
3	lockTheSwapMarketing	modifier	Passed	No Issue

4	releaseTeamFirstStep	external	Same Name Events	Refer Audit Findings
5	releaseTeamSecondStep	external	Same Name Events	Refer Audit Findings
6	startVoteForCommonFee	external	Passed	No Issue
7	voteForCommonFee	external	Infinite loop	Refer Audit Findings
8	checkVotingCommonFee	internal	State function missing pure and view	Refer Audit Findings
9	endCommonFeeVote	internal	Infinite loop	Refer Audit Findings
10	startVoteForSpecialFee	external	Critical operation lacks event log	Refer Audit Findings
11	voteForSpecialFee	external	Infinite loop	Refer Audit Findings
12	checkVotingSpecialFee	internal	Passed	No Issue
13	endSpecialFeeVote	internal	Infinite loop	Refer Audit Findings
14	startVoteForMarketingPool Wallet	external	Critical operation lacks event log	Refer Audit Findings
15	voteForMarketingPoolWallet	external	Infinite loop	Refer Audit Findings
16	checkVotingMarketingPool Wallet	internal	State function missing pure and view	Refer Audit Findings
17	endMarketingPoolWalletVote	internal	Infinite loop	Refer Audit Findings
18	startVoteForCommunityRewardPoolWallet	external	Critical operation lacks event log	Refer Audit Findings
19	voteForCommunityRewardPoolWallet	external	Infinite loop	Refer Audit Findings
20	checkVotingCommunityRewardPoolWallet	internal	State function missing pure and view	Refer Audit Findings
21	endCommunityRewardPoolWalletVote	internal	Infinite loop	Refer Audit Findings
22	_transfer	internal	Passed	No Issue
23	receive	external	Passed	No Issue
24	isContract	read	Passed	No Issue
25	swapAndCommunity	internal	Use keywords/functions to be deprecated	Refer Audit Findings
26	swapAndMarketing	internal	Use keywords/functions to be deprecated	Refer Audit Findings
27	swapAndLiquify	write	Passed	No Issue
28	swapTokensForEth	write	Passed	No Issue

29	addLiquidity	write	Centralized risk in addLiquidity	Refer Audit Findings
30	excludeFromFee	write	access only Owner	No Issue
31	includeInFee	write	access only Owner	No Issue
32	isExcludedFromFee	read	Passed	No Issue
33	includeInSpecialFee	write	access only Owner	No Issue
34	excludeFromSpecialFee	write	access only Owner	No Issue
35	isIncludedInSpecialFee	read	Passed	No Issue
36	name	read	Function name & constructor parameter names are same	Refer Audit Findings
37	symbol	read	Function name & constructor parameter names are same	Refer Audit Findings
38	decimals	read	Function name & constructor parameter names are same	Refer Audit Findings
39	totalSupply	read	Passed	No Issue
40	balanceOf	read	Passed	No Issue
41	transfer	write	Passed	No Issue
42	allowance	read	Passed	No Issue
43	approve	write	Passed	No Issue
44	transferFrom	write	Passed	No Issue
45	increaseAllowance	write	Passed	No Issue
46	decreaseAllowance	write	Passed	No Issue
47	_mint	internal	Passed	No Issue
48	_burn	internal	Passed	No Issue
49	_approve	internal	Passed	No Issue
50	_setupDecimals	internal	Passed	No Issue
51	_beforeTokenTransfer	internal	Passed	No Issue
52	owner	read	Passed	No Issue
53	onlyOwner	modifier	Passed	No Issue
54	renounceOwnership	write	access only Owner	No Issue
55	transferOwnership	write	access only Owner	No Issue
56	_setOwner	write	Passed	No Issue
57	_msgSender	internal	Passed	No Issue
58	_msgData	internal	Passed	No Issue

## Severity Definitions

Risk Level	Description
<b>Critical</b>	Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc.
<b>High</b>	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial
<b>Medium</b>	Medium-level vulnerabilities are important to fix; however, they can't lead to tokens lose
<b>Low</b>	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution
<b>Lowest / Code Style / Best Practice</b>	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) Class 1: Fee on all transactions.[Default = 1%. Configurable from 0% to max 5%]

Class 2: Fee on transactions for specific whitelisted addresses. [Default = 5%.

Configurable from 0% to max 10%]:

```
// Main fee
uint256 public _commonFee = 5;
uint256 private _maxCommonFee = 5;
uint256 public _specialFee = 10;
uint256 private _maxSpecialFee = 10;
```

Default commonFee and SpecialFee are not set as per the document provided:

```
uint256 public _commonFee = 5;
uint256 public _specialFee = 10;
```

**Resolution:** Change default fee values:

```
uint256 private _commonFee = 1;
uint256 public _specialFee = 5;
```

**Status:** Fixed.

(2) Check for the token supply limit.

```
constructor(
    string memory name,
    string memory symbol,
    uint8 decimals,
    uint256 initialSupply,
    address payable initialMarketingPoolWallet,
    address payable initialCommunityRewardPoolWallet,
    address routerAddress, // 0xD99D1c33F9fC344f81754aBC46c52416550D1 - test, 0x10ED43C718714eb63d5aA57B78854704E256024E - main
    address teamWallet
) public payable {
    _name = name;
    _symbol = symbol;
    _decimals = decimals;
    _communityRewardPoolWallet = initialCommunityRewardPoolWallet;
    _marketingPoolWallet = initialMarketingPoolWallet;

    // Initiate multisig wallets @TODO change wallets
    _multisigWallets[0xC21Ec480b55183241b012A7c39ca51D007A4acd] = true;
    _multisigWallets[0x2ec5340c9d4Cd197d87E3316A8312a401867C5E6] = true;
    _multisigWallets[0x831377AaC16848c0D54F95d6EaEF7Ac46b66F4Cc] = true;
    _multisigWallets[0x7F2f13BAe2F787D65f4dbADF34262396D1E7E3De] = true;
    _multisigWallets[0x37b72D07C8aec03E706E5B1efB47f05Ef0922A93] = true;

    IPancakeRouter02 _pancakeswapV2Router = IPancakeRouter02(routerAddress);
    WETH = _pancakeswapV2Router.WETH();

    // Create a Pancake pair for this new token
    pancakeswapV2Pair = IPancakeFactory(_pancakeswapV2Router.factory())
        .createPair(address(this), _pancakeswapV2Router.WETH());

    // set the rest of the contract variables
    pancakeswapV2Router = _pancakeswapV2Router;

    excludeFromFee(address(this));
    excludeFromFee(_msgSender());
    excludeFromFee(routerAddress);

    _teamWallet = teamWallet;
    _lockedTokensForTeam = initialSupply.mul(_percentFromSupplyForTeam).div(100);
    _timeToReleaseFirstStep = block.timestamp + 245 days;
    _timeToReleaseSecondStep = block.timestamp + 365 days;
    // set tokenOwnerAddress as owner of initial supply, more tokens can be minted later
    _mint(_msgSender(), initialSupply.sub(initialSupply.mul(_percentFromSupplyForTeam).div(100)));

    emit Transfer(address(0), _msgSender(), initialSupply);
}
```

There is no maximum or minimum limit for token supply which is set when this contract gets deployed.

**Status:** Fixed.

(3) A specific portion of the supply will be vested for pre-defined time periods.

There are 2 functions which are used to vest specific portions of the supply after the mentioned time period. But those functions can be used multiple times after it reaches the mentioned time.

```
function releaseTeamFirstStep() external {
    require(block.timestamp >= _timeToReleaseFirstStep, "It's not time yet");

    uint256 releaseAmount = _lockedTokensForTeam.mul(_percentReleaseFirstStep).div(100);
    _mint(address(0), releaseAmount);
    emit Transfer(address(0), _teamWallet, releaseAmount);
}

function releaseTeamSecondStep() external {
    require(block.timestamp >= _timeToReleaseSecondStep, "It's not time yet");

    uint256 releaseAmount = _lockedTokensForTeam.mul(_percentReleaseSecondStep).div(100);
    _mint(address(0), releaseAmount);
    emit Transfer(address(0), _teamWallet, releaseAmount);
}
```

Any user can call this function.

**Note:** If it is a part of the plan then okay.

**Status:** Acknowledged.

(4) Centralized risk in addLiquidity:

```
function addLiquidity(uint256 tokenAmount, uint256 ethAmount) private {
    // approve token transfer to cover all possible scenarios
    _approve(address(this), address(pancakeswapV2Router), tokenAmount);

    // add the liquidity
    pancakeswapV2Router.addLiquidityETH{value : ethAmount}(
        address(this),
        tokenAmount,
        0, // slippage is unavoidable
        0, // slippage is unavoidable
        owner(),
        block.timestamp
    );
}
```

In addLiquidityETH function, owner gets Tokens from the Pool. If the private key of the owner's wallet is compromised, then it will create a problem.

**Resolution:** Ideally this can be a governance smart contract. On another hand, the owner can accept this risk and handle the private key very securely.

**Status:** Acknowledged.

(5) Infinite loop:

Below functions for loop do not have \_excluded length limit , which costs more gas.

- voteForCommonFee
- checkVotingCommonFee
- endCommonFeeVote
- voteForSpecialFee
- endSpecialFeeVote
- voteForMarketingPoolWallet,
- checkVotingMarketingPoolWallet
- endMarketingPoolWalletVote
- voteForCommunityRewardPoolWallet
- checkVotingCommunityRewardPoolWallet
- endCommunityRewardPoolWalletVote

**Resolution:** upper limit should be limited in for loops.

**Status:** Acknowledged.

## **Very Low / Discussion / Best practices:**

(1) Use latest solidity version:

```
pragma solidity ^0.8.0;
```

Using the latest solidity will prevent any compiler level bugs.

**Resolution:** Please use 0.8.6 which is the latest version.

**Status:** Acknowledged.

(2) Warning: SPDX license identifier:

```
Warning: SPDX license identifier
not provided in source file.
Before publishing, consider adding
a comment containing "SPDX-
License-Identifier: <SPDX-
License>" to each source file. Use
"SPDX-License-Identifier:
UNLICENSED" for non-open-source
code. Please see https://spdx.org
for more information. -->
contracts/XPSToken.sol
```

SafeMath.sol: Warning: SPDX license identifier not provided in source file.

**Resolution:** SPDX-License-Identifier.

**Status:** Fixed.

(3) Function name & constructor parameter names are same:

```
constructor(
    string memory name,
    string memory symbol,
```

```
/**
 * @dev Returns the name of the token.
 */
function name() public view virtual returns (string memory) {
    return _name;
}

/**
 * @dev Returns the symbol of the token, usually a shorter version of the
 * name.
 */
function symbol() public view virtual returns (string memory) {
    return _symbol;
}

/**
 * @dev Returns the number of decimals used to get its user representation.
 * For example, if `decimals` equals `2`, a balance of `505` tokens should
 * be displayed to a user as `5,05` (`505 / 10 ** 2`).
 *
 * Tokens usually opt for a value of 18, imitating the relationship between
 * Ether and Wei. This is the value {BEP20} uses, unless {_setupDecimals} is
 * called.
 *
 * NOTE: This information is only used for _display_ purposes: it in
 * no way affects any of the arithmetic of the contract, including
 * {BEP20-balanceOf} and {BEP20-transfer}.
 */
function decimals() public view virtual returns (uint8) {
    return _decimals;
}
```



```
Warning: This declaration shadows
an existing declaration. -->
XPSToken.sol:165:9: | 165 | string
memory name, | ^^^^^^^^^^^^^^^^^^^^^
Note: The shadowed declaration is
here: --> XPSToken.sol:715:5: |
715 | function name() public view
virtual returns (string memory) {
| ^ (Relevant source part starts
here and spans across multiple
lines).
```

```
Warning: This declaration shadows
an existing declaration. -->
XPSToken.sol:166:9: | 166 | string
memory symbol, |
^^^^^^^^^^^^^^^^^^^^ Note: The
shadowed declaration is here: -->
XPSToken.sol:723:5: | 723 |
function symbol() public view
virtual returns (string memory) {
| ^ (Relevant source part starts
here and spans across multiple
lines).
```

```
Warning: This declaration shadows
an existing declaration. -->
XPSToken.sol:167:9: | 167 | uint8
decimals, | ^^^^^^^^^^^^^^^^^ Note:
The shadowed declaration is here:
--> XPSToken.sol:740:5: | 740 |
function decimals() public view
virtual returns (uint8) { | ^
(Relevant source part starts here
and spans across multiple lines).
```

The code has functions and constructor parameters having words :

- name
- symbol
- decimals

**Resolution:** Function name and constructor parameters name should be different.

**Status:** Fixed.

(4) Use keywords/functions to be deprecated:

```
function swapAndCommunity(uint256 amount) internal lockTheSwapCommunity {
    // capture the contract's current ETH balance.
    uint256 initialBalance = address(this).balance;

    // swap tokens for ETH
    swapTokensForEth(amount);

    // how much ETH did we just swap into?
    uint256 newBalance = address(this).balance.sub(initialBalance);

    // Send
    _communityRewardPoolWallet.send(newBalance);

    emit SwapAndCommunity(amount, newBalance);
}

function swapAndMarketing(uint256 amount) internal lockTheSwapMarketing {
    // capture the contract's current ETH balance.
    uint256 initialBalance = address(this).balance;

    // swap tokens for ETH
    swapTokensForEth(amount);

    // how much ETH did we just swap into?
    uint256 newBalance = address(this).balance.sub(initialBalance);

    // Send
    _marketingPoolWallet.send(newBalance);

    emit SwapAndMarketing(amount, newBalance);
}
```

Failure condition of 'send' ignored. Consider using 'transfer' instead.

**Resolution:** Please use Transfer instead of Send.

**Status:** Acknowledged.

(5) Same Name Events:

```
function releaseTeamFirstStep() external {
    require(block.timestamp >= _timeToReleaseFirstStep, "It's not time yet");

    uint256 releaseAmount = _lockedTokensForTeam.mul(_percentReleaseFirstStep).div(100);
    _mint(address(0), releaseAmount);
    emit Transfer(address(0), _teamWallet, releaseAmount);
}

function releaseTeamSecondStep() external {
    require(block.timestamp >= _timeToReleaseSecondStep, "It's not time yet");

    uint256 releaseAmount = _lockedTokensForTeam.mul(_percentReleaseSecondStep).div(100);
    _mint(address(0), releaseAmount);
    emit Transfer(address(0), _teamWallet, releaseAmount);
}
```

Event names are the same in releaseTeamFirstStep , releaseTeamSecondStep.

**Resolution:** In both these functions \_mint function has been used , where Transfer event got fired already. So instead of a Transfer event there should be some relevant event.

**Status:** Acknowledged.

(6) State function missing pure and view:

multiple lines).

Warning: Function state mutability  
can be restricted to view -->  
XPSToken.sol:262:5: | 262 |  
function checkVotingCommonFee()  
internal returns (uint8) { | ^  
(Relevant source part starts here  
and spans across multiple lines).

Warning: Function state mutability  
can be restricted to view -->  
XPSToken.sol:407:5: | 407 |  
function  
checkVotingMarketingPoolWallet()  
internal returns (uint8) { | ^  
(Relevant source part starts here  
and spans across multiple lines).

Warning: Function state mutability  
can be restricted to view -->  
XPSToken.sol:479:5: | 479 |  
function  
checkVotingCommunityRewardPoolWal  
let() internal returns (uint8) { |  
^ (Relevant source part starts  
here and spans across multiple  
lines).

```
function checkVotingCommunityRewardPoolWallet() internal returns (uint8) {  
    uint256 acceptedVotes = 0;  
    uint256 declinedVotes = 0;
```

```
function checkVotingCommonFee() internal returns (uint8) {  
    uint256 acceptedVotes = 0;  
    uint256 declinedVotes = 0;  
  
    for (uint i = _currentOffsetVoteCommonFee; i < _votedCommonFeeWallets.length;  
        address voteWallet = _votedCommonFeeWallets[i];  
        if (_votesCommonFee[voteWallet]) {
```

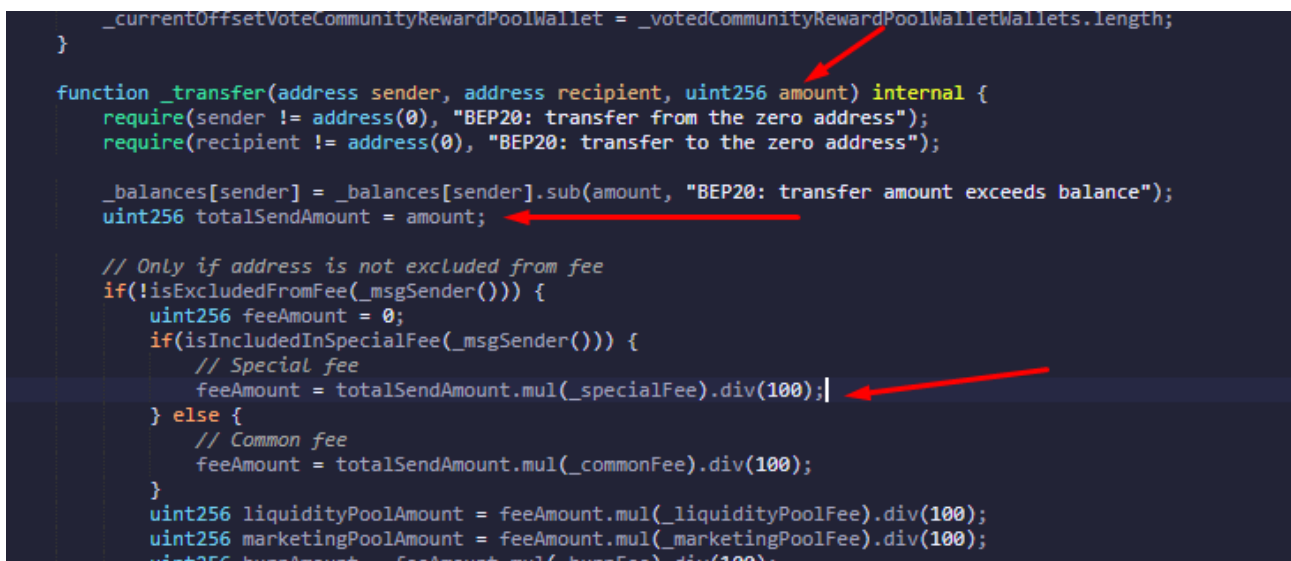
Function state mutability can be restricted to view.

**Resolution:** Please add view in below functions :

- checkVotingMarketingPoolWallet
- checkVotingCommonFee
- checkVotingCommunityRewardPoolWallet

**Status:** Acknowledged.

(7) Function input parameters lack of check:



```
    _currentOffsetVoteCommunityRewardPoolWallet = _votedCommunityRewardPoolWalletWallets.length;
}

function _transfer(address sender, address recipient, uint256 amount) internal {
    require(sender != address(0), "BEP20: transfer from the zero address");
    require(recipient != address(0), "BEP20: transfer to the zero address");

    _balances[sender] = _balances[sender].sub(amount, "BEP20: transfer amount exceeds balance");
    uint256 totalSendAmount = amount;

    // Only if address is not excluded from fee
    if(!isExcludedFromFee(_msgSender())) {
        uint256 feeAmount = 0;
        if(isIncludedInSpecialFee(_msgSender())) {
            // Special fee
            feeAmount = totalSendAmount.mul(_specialFee).div(100);
        } else {
            // Common fee
            feeAmount = totalSendAmount.mul(_commonFee).div(100);
        }
        uint256 liquidityPoolAmount = feeAmount.mul(_liquidityPoolFee).div(100);
        uint256 marketingPoolAmount = feeAmount.mul(_marketingPoolFee).div(100);
        uint256 burnAmount = feeAmount.mul(_burnFee).div(100);
    }
}
```

Variable validation is not performed in below functions: \_transfer .

**Resolution:** Put validation: variable is not empty and > 0.

**Status:** Acknowledged.

(8) Critical operation lacks event log:

The missing event log for :

- startVoteForMarketingPoolWallet
- voteForMarketingPoolWallet
- startVoteForCommunityRewardPoolWallet
- voteForCommunityRewardPoolWallet
- startVoteForSpecialFee
- voteForSpecialFee

**Resolution:** Please write an event log for listed events.

**Status:** Acknowledged.

(9) Unused variables:

```
// Main fee
uint256 public _commonFee = 5;
uint256 private _maxCommonFee = 5;
uint256 public _specialFee = 10;
uint256 private _maxSpecialFee = 10;
```

In constructor, there are 2 variables declare with values but both are not used in anywhere in contract:

1 - \_maxCommonFee

2 - \_maxSpecialFee

**Resolution:** Remove unused variables.

**Status:** Acknowledged.

(10) Make variables constant:

```
// Fees
uint256 public _liquidityPoolFee = 30;
uint256 public _marketingPoolFee = 40;
uint256 public _burnFee = 10;
uint256 public _communityRewardPoolFee = 20;

// Main fee
uint256 public _commonFee = 5;
uint256 private _maxCommonFee = 5;
uint256 public _specialFee = 10;
uint256 private _maxSpecialFee = 10;

// Vote config
uint256 private _minAcceptedVotes = 3;
uint256 private _minDeclinedVotes = 3;
```

```
// Sent to pools on transaction
bool private _swapOnTransaction = true;

// Trigger amount to auto swap
uint256 public _liquidityTriggerAmount = 5 * 10 ** 12; // = 5,000 tokens
uint256 public _marketingTriggerAmount = 5 * 10 ** 12; // = 5,000 tokens
uint256 public _communityTriggerAmount = 5 * 10 ** 12; // = 5,000 tokens
```

```
uint256 public immutable lockedTokensForTeam;  
uint256 public _percentFromSupplyForTeam = 15;  
uint256 public immutable _percentReleaseFirstStep = 60;  
uint256 public immutable _percentReleaseSecondStep = 40;
```

These variables' values will be unchanged. So, please make it constant. It will save some gas.

**Resolution:** Declare those variables as constant. Just put a constant keyword. And define constants in the constructor.

**Status:** Acknowledged.

(11) HardCoded address:

```
// Initiate multisig wallets @TODO change wallets  
_multisigWallets[0xC21Ec48Db5551B3241b012A7c39ca51D007A4acd] = true;  
_multisigWallets[0x2ec5340c9d4Cd197dB7E3316A8312a401867C5E6] = true;  
_multisigWallets[0x831377AaC16848c0D54F95d6EaEF7Ac46b66F4Cc] = true;  
_multisigWallets[0x7F2f13BAe2F787D65f4dbADF34262396D1E7E3De] = true;  
_multisigWallets[0x37b72D07C8aec03E706E5B1efB47f05Ef0922A93] = true;
```

Here some hardcoded wallet addresses have been mentioned as multisig for the contract.

**Resolution:** Owner has to double confirm before deploying.

**Status:** Acknowledged.

(12) Informational:

```
// Fees  
uint256 public _liquidityPoolFee = 20;  
uint256 public _marketingPoolFee = 70;  
uint256 public _burnFee = 5;  
uint256 public _communityRewardPoolFee = 5;
```

20% Liquidity Pool Fee

70% Marketing Pool Fee

5% Burn Fee

5% Community Reward Pool Fee

**Resolution:** Must verify the percentage divided.

**Status:** Fixed.

## Centralization

These smart contracts have 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:

- `excludeFromFee`: The Owner can set excluded fee status true.
- `includeInFee`: The Owner can set included fee status false.
- `includeInSpecialFee`: The Owner can set included special fee status true.
- `excludeFromSpecialFee`: The Owner can set excluded special fee status false.

## 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 issues are fixed in revised code. So, **it's good to go to 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 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).

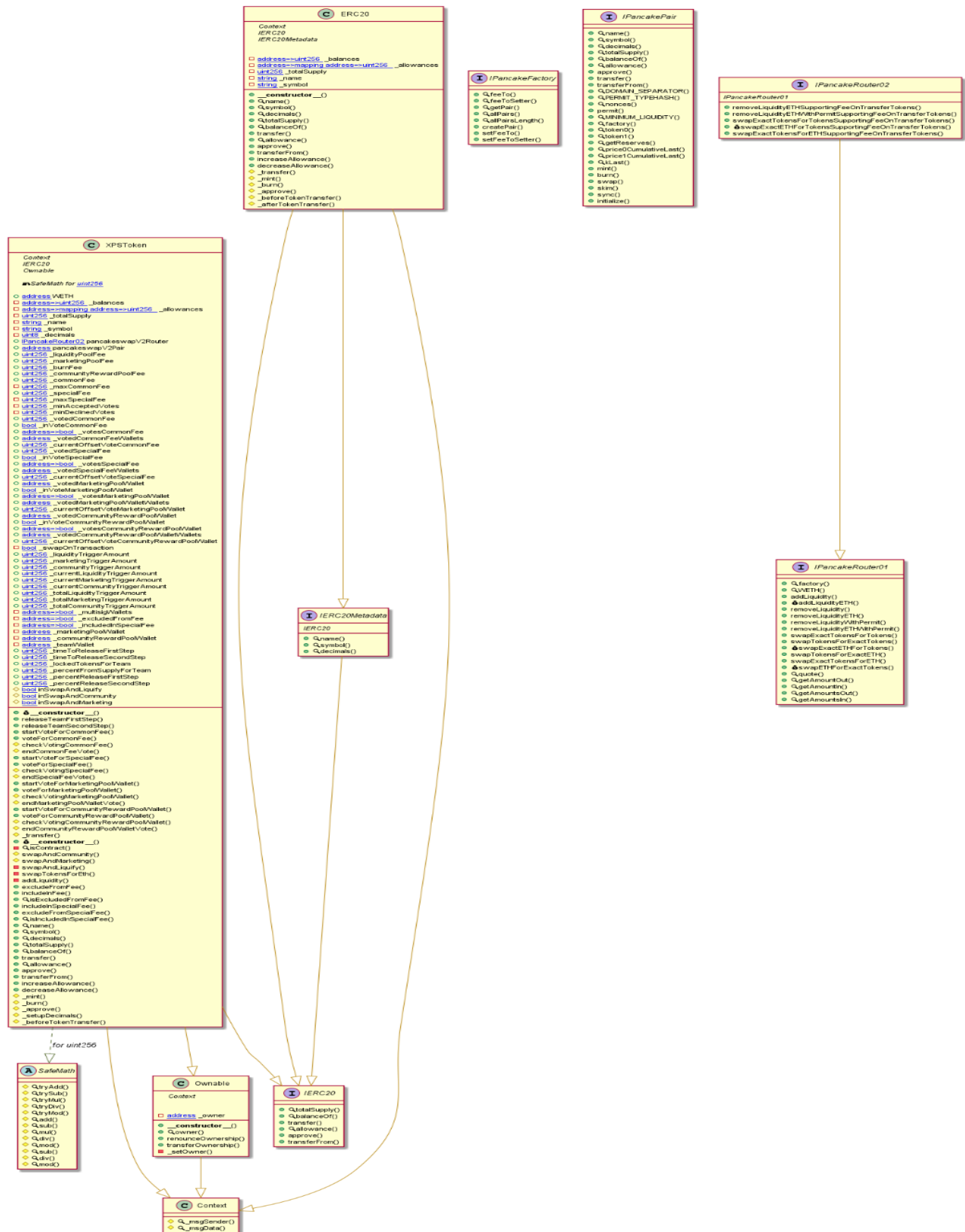
Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we 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 - XPSToken



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](mailto:audit@EtherAuthority.io)

# Slither Results Log

## Slither log >> XPSToken.sol

INFO:Detectors:

XPSToken.swapAndCommunity(uint256) (XPSToken.sol#1518-1532) sends eth to arbitrary user

Dangerous calls:

- \_communityRewardPoolWallet.send(newBalance) (XPSToken.sol#1529)

XPSToken.swapAndMarketing(uint256) (XPSToken.sol#1534-1548) sends eth to arbitrary user

Dangerous calls:

- \_marketingPoolWallet.send(newBalance) (XPSToken.sol#1545)

XPSToken.addLiquidity(uint256,uint256) (XPSToken.sol#1594-1607) sends eth to arbitrary user

Dangerous calls:

- pancakeswapV2Router.addLiquidityETH{value:

ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (XPSToken.sol#1599-1606)

Reference:

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

INFO:Detectors:

Reentrancy in XPSToken.\_transfer(address,address,uint256) (XPSToken.sol#1441-1504):

External calls:

- swapAndCommunity(\_currentCommunityTriggerAmount) (XPSToken.sol#1486)

-

pancakeswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (XPSToken.sol#1585-1591)

External calls sending eth:

- swapAndCommunity(\_currentCommunityTriggerAmount) (XPSToken.sol#1486)

- \_communityRewardPoolWallet.send(newBalance) (XPSToken.sol#1529)

State variables written after the call(s):

- \_currentCommunityTriggerAmount = 0 (XPSToken.sol#1487)

Reentrancy in XPSToken.\_transfer(address,address,uint256) (XPSToken.sol#1441-1504):

External calls:

- swapAndCommunity(\_currentCommunityTriggerAmount) (XPSToken.sol#1486)

-

pancakeswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (XPSToken.sol#1585-1591)

- swapAndMarketing(\_currentMarketingTriggerAmount) (XPSToken.sol#1491)

-

pancakeswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (XPSToken.sol#1585-1591)

External calls sending eth:

- swapAndCommunity(\_currentCommunityTriggerAmount) (XPSToken.sol#1486)

- \_communityRewardPoolWallet.send(newBalance) (XPSToken.sol#1529)

- swapAndMarketing(\_currentMarketingTriggerAmount) (XPSToken.sol#1491)

- \_marketingPoolWallet.send(newBalance) (XPSToken.sol#1545)

State variables written after the call(s):

- \_currentMarketingTriggerAmount = 0 (XPSToken.sol#1492)

Reentrancy in XPSToken.\_transfer(address,address,uint256) (XPSToken.sol#1441-1504):

External calls:

- swapAndCommunity(\_currentCommunityTriggerAmount) (XPSToken.sol#1486)

-

pancakeswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (XPSToken.sol#1585-1591)

- swapAndMarketing(\_currentMarketingTriggerAmount) (XPSToken.sol#1491)

-

pancakeswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (XPSToken.sol#1585-1591)

- swapAndLiquify(\_currentLiquidityTriggerAmount) (XPSToken.sol#1496)

- pancakeswapV2Router.addLiquidityETH{value:

ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (XPSToken.sol#1599-1606)

pancakeswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (XPSToken.sol#1585-1591)

External calls sending eth:

- swapAndCommunity(\_currentCommunityTriggerAmount) (XPSToken.sol#1486)
  - \_communityRewardPoolWallet.send(newBalance) (XPSToken.sol#1529)
- swapAndMarketing(\_currentMarketingTriggerAmount) (XPSToken.sol#1491)
  - \_marketingPoolWallet.send(newBalance) (XPSToken.sol#1545)
- swapAndLiquify(\_currentLiquidityTriggerAmount) (XPSToken.sol#1496)
  - pancakeswapV2Router.addLiquidityETH{value:

ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (XPSToken.sol#1599-1606)

State variables written after the call(s):

- \_balances[recipient] = \_balances[recipient].add(totalSendAmount) (XPSToken.sol#1502)
- \_currentLiquidityTriggerAmount = 0 (XPSToken.sol#1497)

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

INFO:Detectors:

XPSToken.\_transfer(address,address,uint256) (XPSToken.sol#1441-1504) performs a multiplication on the result of a division:

- feeAmount = totalSendAmount.mul(\_specialFee).div(100) (XPSToken.sol#1453)
- liquidityPoolAmount = feeAmount.mul(\_liquidityPoolFee).div(100) (XPSToken.sol#1458)

XPSToken.\_transfer(address,address,uint256) (XPSToken.sol#1441-1504) performs a multiplication on the result of a division:

- feeAmount = totalSendAmount.mul(\_specialFee).div(100) (XPSToken.sol#1453)
- marketingPoolAmount = feeAmount.mul(\_marketingPoolFee).div(100) (XPSToken.sol#1459)

XPSToken.\_transfer(address,address,uint256) (XPSToken.sol#1441-1504) performs a multiplication on the result of a division:

- feeAmount = totalSendAmount.mul(\_specialFee).div(100) (XPSToken.sol#1453)
- burnAmount = feeAmount.mul(\_burnFee).div(100) (XPSToken.sol#1460)

XPSToken.\_transfer(address,address,uint256) (XPSToken.sol#1441-1504) performs a multiplication on the result of a division:

- feeAmount = totalSendAmount.mul(\_specialFee).div(100) (XPSToken.sol#1453)
- communityRewardPoolAmount = feeAmount.mul(\_communityRewardPoolFee).div(100)

(XPSToken.sol#1461)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply>

INFO:Detectors:

XPSToken.swapAndCommunity(uint256) (XPSToken.sol#1518-1532) ignores return value by

\_communityRewardPoolWallet.send(newBalance) (XPSToken.sol#1529)

XPSToken.swapAndMarketing(uint256) (XPSToken.sol#1534-1548) ignores return value by

\_marketingPoolWallet.send(newBalance) (XPSToken.sol#1545)

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

INFO:Detectors:

XPSToken.addLiquidity(uint256,uint256) (XPSToken.sol#1594-1607) ignores return value by

pancakeswapV2Router.addLiquidityETH{value:

ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (XPSToken.sol#1599-1606)

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

INFO:Detectors:

XPSToken.constructor(string,string,uint8,uint256,address,address,address,address).name

(XPSToken.sol#1087) shadows:

- XPSToken.name() (XPSToken.sol#1637-1639) (function)

XPSToken.constructor(string,string,uint8,uint256,address,address,address,address).symbol

(XPSToken.sol#1088) shadows:

- XPSToken.symbol() (XPSToken.sol#1645-1647) (function)

XPSToken.constructor(string,string,uint8,uint256,address,address,address,address).decimals

(XPSToken.sol#1089) shadows:

- XPSToken.decimals() (XPSToken.sol#1662-1664) (function)

XPSToken.allowance(address,address).owner (XPSToken.sol#1696) shadows:

- Ownable.owner() (XPSToken.sol#561-563) (function)

XPSToken.\_approve(address,address,uint256).owner (XPSToken.sol#1820) shadows:

- Ownable.owner() (XPSToken.sol#561-563) (function)

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

INFO:Detectors:

XPSToken.constructor(string,string,uint8,uint256,address,address,address,address).initialCommunityRewardPoolWallet (XPSToken.sol#1092) lacks a zero-check on :

- \_communityRewardPoolWallet = initialCommunityRewardPoolWallet (XPSToken.sol#1099)



XPSToken.constructor(string,string,uint8,uint256,address,address,address,address).initialMarketingPoolWallet (XPSToken.sol#1091) lacks a zero-check on :

- \_marketingPoolWallet = initialMarketingPoolWallet (XPSToken.sol#1100)

XPSToken.constructor(string,string,uint8,uint256,address,address,address,address).teamWallet (XPSToken.sol#1094) lacks a zero-check on :

- \_teamWallet = teamWallet (XPSToken.sol#1123)

XPSToken.startVoteForMarketingPoolWallet(address).newMarketingPoolWallet (XPSToken.sol#1298) lacks a zero-check on :

- \_votedMarketingPoolWallet = newMarketingPoolWallet (XPSToken.sol#1303)

XPSToken.startVoteForCommunityRewardPoolWallet(address).newCommunityRewardPoolWallet (XPSToken.sol#1370) lacks a zero-check on :

- \_votedCommunityRewardPoolWallet = newCommunityRewardPoolWallet (XPSToken.sol#1375)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation>  
INFO:Detectors:

Reentrancy in XPSToken.\_transfer(address,address,uint256) (XPSToken.sol#1441-1504):

External calls:

- swapAndCommunity(\_currentCommunityTriggerAmount) (XPSToken.sol#1486)

-

pancakeswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (XPSToken.sol#1585-1591)

- swapAndMarketing(\_currentMarketingTriggerAmount) (XPSToken.sol#1491)

-

pancakeswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (XPSToken.sol#1585-1591)

External calls sending eth:

- swapAndCommunity(\_currentCommunityTriggerAmount) (XPSToken.sol#1486)

- \_communityRewardPoolWallet.send(newBalance) (XPSToken.sol#1529)

- swapAndMarketing(\_currentMarketingTriggerAmount) (XPSToken.sol#1491)

- \_marketingPoolWallet.send(newBalance) (XPSToken.sol#1545)

State variables written after the call(s):

- swapAndMarketing(\_currentMarketingTriggerAmount) (XPSToken.sol#1491)

- \_allowances[owner][spender] = amount (XPSToken.sol#1824)

- swapAndMarketing(\_currentMarketingTriggerAmount) (XPSToken.sol#1491)

- inSwapAndMarketing = true (XPSToken.sol#1079)

- inSwapAndMarketing = false (XPSToken.sol#1081)

Reentrancy in XPSToken.\_transfer(address,address,uint256) (XPSToken.sol#1441-1504):

External calls:

- swapAndCommunity(\_currentCommunityTriggerAmount) (XPSToken.sol#1486)

-

pancakeswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (XPSToken.sol#1585-1591)

- swapAndMarketing(\_currentMarketingTriggerAmount) (XPSToken.sol#1491)

-

pancakeswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (XPSToken.sol#1585-1591)

- swapAndLiquify(\_currentLiquidityTriggerAmount) (XPSToken.sol#1496)

- pancakeswapV2Router.addLiquidityETH{value:

ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (XPSToken.sol#1599-1606)

-

pancakeswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (XPSToken.sol#1585-1591)

External calls sending eth:

- swapAndCommunity(\_currentCommunityTriggerAmount) (XPSToken.sol#1486)

- \_communityRewardPoolWallet.send(newBalance) (XPSToken.sol#1529)

- swapAndMarketing(\_currentMarketingTriggerAmount) (XPSToken.sol#1491)

- \_marketingPoolWallet.send(newBalance) (XPSToken.sol#1545)

- swapAndLiquify(\_currentLiquidityTriggerAmount) (XPSToken.sol#1496)

- pancakeswapV2Router.addLiquidityETH{value:

ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (XPSToken.sol#1599-1606)

State variables written after the call(s):

- swapAndLiquify(\_currentLiquidityTriggerAmount) (XPSToken.sol#1496)

- \_allowances[owner][spender] = amount (XPSToken.sol#1824)

- swapAndLiquify(\_currentLiquidityTriggerAmount) (XPSToken.sol#1496)

- inSwapAndLiquify = true (XPSToken.sol#1059)

- inSwapAndLiquify = false (XPSToken.sol#1061)

Reentrancy in XPSToken.constructor(string,string,uint8,uint256,address,address,address,address)  
(XPSToken.sol#1086-1131):

External calls:

- pancakeswapV2Pair =

IPancakeFactory(\_pancakeswapV2Router.factory()).createPair(address(this),\_pancakeswapV2Router.WETH()) (XPSToken.sol#1113-1114)

State variables written after the call(s):

- \_mint(\_msgSender(),initialSupply.sub(initialSupply.mul(\_percentFromSupplyForTeam).div(100)))

(XPSToken.sol#1128)

- \_balances[account] = \_balances[account].add(amount) (XPSToken.sol#1782)

- excludeFromFee(address(this)) (XPSToken.sol#1119)

- \_excludedFromFee[account] = true (XPSToken.sol#1610)

- excludeFromFee(\_msgSender()) (XPSToken.sol#1120)

- \_excludedFromFee[account] = true (XPSToken.sol#1610)

- excludeFromFee(routerAddress) (XPSToken.sol#1121)

- \_excludedFromFee[account] = true (XPSToken.sol#1610)

- \_lockedTokensForTeam = initialSupply.mul(\_percentFromSupplyForTeam).div(100)

(XPSToken.sol#1124)

- \_teamWallet = teamWallet (XPSToken.sol#1123)

- \_timeToReleaseFirstStep = block.timestamp + 20995200 (XPSToken.sol#1125)

- \_timeToReleaseSecondStep = block.timestamp + 31536000 (XPSToken.sol#1126)

- \_mint(\_msgSender(),initialSupply.sub(initialSupply.mul(\_percentFromSupplyForTeam).div(100)))

(XPSToken.sol#1128)

- \_totalSupply = \_totalSupply.add(amount) (XPSToken.sol#1781)

- pancakeswapV2Router = \_pancakeswapV2Router (XPSToken.sol#1117)

Reentrancy in XPSToken.swapAndLiquify(uint256) (XPSToken.sol#1550-1574):

External calls:

- swapTokensForEth(half) (XPSToken.sol#1564)

-

pancakeswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (XPSToken.sol#1585-1591)

- addLiquidity(otherHalf,newBalance) (XPSToken.sol#1571)

- pancakeswapV2Router.addLiquidityETH{value:

ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (XPSToken.sol#1599-1606)

External calls sending eth:

- addLiquidity(otherHalf,newBalance) (XPSToken.sol#1571)

- pancakeswapV2Router.addLiquidityETH{value:

ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (XPSToken.sol#1599-1606)

State variables written after the call(s):

- addLiquidity(otherHalf,newBalance) (XPSToken.sol#1571)

- \_allowances[owner][spender] = amount (XPSToken.sol#1824)

Reentrancy in XPSToken.transferFrom(address,address,uint256) (XPSToken.sol#1725-1729):

External calls:

- \_transfer(sender,recipient,amount) (XPSToken.sol#1726)

- pancakeswapV2Router.addLiquidityETH{value:

ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (XPSToken.sol#1599-1606)

-

pancakeswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (XPSToken.sol#1585-1591)

External calls sending eth:

- \_transfer(sender,recipient,amount) (XPSToken.sol#1726)

- pancakeswapV2Router.addLiquidityETH{value:

ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (XPSToken.sol#1599-1606)

- \_marketingPoolWallet.send(newBalance) (XPSToken.sol#1545)

- \_communityRewardPoolWallet.send(newBalance) (XPSToken.sol#1529)

State variables written after the call(s):

- \_approve(sender,\_msgSender(),\_allowances[sender][\_msgSender()].sub(amount,BEP20: transfer amount exceeds allowance)) (XPSToken.sol#1727)

- \_allowances[owner][spender] = amount (XPSToken.sol#1824)

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

INFO:Detectors:

Reentrancy in XPSToken.\_transfer(address,address,uint256) (XPSToken.sol#1441-1504):

External calls:

- swapAndCommunity(\_currentCommunityTriggerAmount) (XPSToken.sol#1486)



-  
pancakeswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (XPSToken.sol#1585-1591)  
- swapAndMarketing(\_currentMarketingTriggerAmount) (XPSToken.sol#1491)  
-  
pancakeswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (XPSToken.sol#1585-1591)  
External calls sending eth:  
- swapAndCommunity(\_currentCommunityTriggerAmount) (XPSToken.sol#1486)  
- \_communityRewardPoolWallet.send(newBalance) (XPSToken.sol#1529)  
- swapAndMarketing(\_currentMarketingTriggerAmount) (XPSToken.sol#1491)  
- \_marketingPoolWallet.send(newBalance) (XPSToken.sol#1545)  
Event emitted after the call(s):  
- Approval(owner,spender,amount) (XPSToken.sol#1825)  
- swapAndMarketing(\_currentMarketingTriggerAmount) (XPSToken.sol#1491)  
- SwapAndMarketing(amount,newBalance) (XPSToken.sol#1547)  
- swapAndMarketing(\_currentMarketingTriggerAmount) (XPSToken.sol#1491)  
Reentrancy in XPSToken.\_transfer(address,address,uint256) (XPSToken.sol#1441-1504):  
External calls:  
- swapAndCommunity(\_currentCommunityTriggerAmount) (XPSToken.sol#1486)  
-  
pancakeswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (XPSToken.sol#1585-1591)  
- swapAndMarketing(\_currentMarketingTriggerAmount) (XPSToken.sol#1491)  
-  
pancakeswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (XPSToken.sol#1585-1591)  
- swapAndLiquify(\_currentLiquidityTriggerAmount) (XPSToken.sol#1496)  
- pancakeswapV2Router.addLiquidityETH{value:  
ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (XPSToken.sol#1599-1606)  
-  
pancakeswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (XPSToken.sol#1585-1591)  
External calls sending eth:  
- swapAndCommunity(\_currentCommunityTriggerAmount) (XPSToken.sol#1486)  
- \_communityRewardPoolWallet.send(newBalance) (XPSToken.sol#1529)  
- swapAndMarketing(\_currentMarketingTriggerAmount) (XPSToken.sol#1491)  
- \_marketingPoolWallet.send(newBalance) (XPSToken.sol#1545)  
- swapAndLiquify(\_currentLiquidityTriggerAmount) (XPSToken.sol#1496)  
- pancakeswapV2Router.addLiquidityETH{value:  
ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (XPSToken.sol#1599-1606)  
Event emitted after the call(s):  
- Approval(owner,spender,amount) (XPSToken.sol#1825)  
- swapAndLiquify(\_currentLiquidityTriggerAmount) (XPSToken.sol#1496)  
- SwapAndLiquify(half,newBalance,otherHalf) (XPSToken.sol#1573)  
- swapAndLiquify(\_currentLiquidityTriggerAmount) (XPSToken.sol#1496)  
- Transfer(sender,recipient,amount) (XPSToken.sol#1503)  
Reentrancy in XPSToken.constructor(string,string,uint8,uint256,address,address,address,address) (XPSToken.sol#1086-1131):  
External calls:  
- pancakeswapV2Pair =  
IPancakeFactory(\_pancakeswapV2Router.factory()).createPair(address(this),\_pancakeswapV2Router.WETH()) (XPSToken.sol#1113-1114)  
Event emitted after the call(s):  
- Transfer(address(0),account,amount) (XPSToken.sol#1783)  
- \_mint(\_msgSender(),initialSupply.sub(initialSupply.mul(\_percentFromSupplyForTeam).div(100))) (XPSToken.sol#1128)  
- Transfer(address(0),\_msgSender(),initialSupply) (XPSToken.sol#1130)  
Reentrancy in XPSToken.swapAndCommunity(uint256) (XPSToken.sol#1518-1532):  
External calls:  
- swapTokensForEth(amount) (XPSToken.sol#1523)  
-  
pancakeswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (XPSToken.sol#1585-1591)  
External calls sending eth:

- `_communityRewardPoolWallet.send(newBalance)` (XPSToken.sol#1529)

Event emitted after the call(s):

- `SwapAndCommunity(amount,newBalance)` (XPSToken.sol#1531)

Reentrancy in `XPSToken.swapAndLiquify(uint256)` (XPSToken.sol#1550-1574):

External calls:

- `swapTokensForEth(half)` (XPSToken.sol#1564)

-

`pancakeswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp)` (XPSToken.sol#1585-1591)

- `addLiquidity(otherHalf,newBalance)` (XPSToken.sol#1571)
- `pancakeswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp)` (XPSToken.sol#1599-1606)

External calls sending eth:

- `addLiquidity(otherHalf,newBalance)` (XPSToken.sol#1571)
- `pancakeswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp)` (XPSToken.sol#1599-1606)

Event emitted after the call(s):

- `Approval(owner,spender,amount)` (XPSToken.sol#1825)
- `addLiquidity(otherHalf,newBalance)` (XPSToken.sol#1571)
- `SwapAndLiquify(half,newBalance,otherHalf)` (XPSToken.sol#1573)

Reentrancy in `XPSToken.swapAndMarketing(uint256)` (XPSToken.sol#1534-1548):

External calls:

- `swapTokensForEth(amount)` (XPSToken.sol#1539)

-

`pancakeswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp)` (XPSToken.sol#1585-1591)

External calls sending eth:

- `_marketingPoolWallet.send(newBalance)` (XPSToken.sol#1545)

Event emitted after the call(s):

- `SwapAndMarketing(amount,newBalance)` (XPSToken.sol#1547)

Reentrancy in `XPSToken.transferFrom(address,address,uint256)` (XPSToken.sol#1725-1729):

External calls:

- `_transfer(sender,recipient,amount)` (XPSToken.sol#1726)
- `pancakeswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp)` (XPSToken.sol#1599-1606)

-

`pancakeswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp)` (XPSToken.sol#1585-1591)

External calls sending eth:

- `_transfer(sender,recipient,amount)` (XPSToken.sol#1726)
- `pancakeswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp)` (XPSToken.sol#1599-1606)
- `_marketingPoolWallet.send(newBalance)` (XPSToken.sol#1545)
- `_communityRewardPoolWallet.send(newBalance)` (XPSToken.sol#1529)

Event emitted after the call(s):

- `Approval(owner,spender,amount)` (XPSToken.sol#1825)
- `_approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,BEP20: transfer amount exceeds allowance))` (XPSToken.sol#1727)

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

INFO:Detectors:

`XPSToken.releaseTeamFirstStep()` (XPSToken.sol#1133-1139) uses timestamp for comparisons

Dangerous comparisons:

- `require(bool,string)(block.timestamp >= _timeToReleaseFirstStep,It's not time yet)` (XPSToken.sol#1134)

`XPSToken.releaseTeamSecondStep()` (XPSToken.sol#1141-1147) uses timestamp for comparisons

Dangerous comparisons:

- `require(bool,string)(block.timestamp >= _timeToReleaseSecondStep,It's not time yet)` (XPSToken.sol#1142)

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

INFO:Detectors:

`XPSToken.isContract(address)` (XPSToken.sol#1512-1516) uses assembly

- `INLINE ASM` (XPSToken.sol#1514)

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

INFO:Detectors:

`Context._msgData()` (XPSToken.sol#530-532) is never used and should be removed

ERC20.\_burn(address,uint256) (XPSToken.sol#864-879) is never used and should be removed  
ERC20.\_mint(address,uint256) (XPSToken.sol#841-851) is never used and should be removed  
SafeMath.div(uint256,uint256,string) (XPSToken.sol#179-188) is never used and should be removed  
SafeMath.mod(uint256,uint256) (XPSToken.sol#139-141) is never used and should be removed  
SafeMath.mod(uint256,uint256,string) (XPSToken.sol#205-214) is never used and should be removed  
SafeMath.tryAdd(uint256,uint256) (XPSToken.sol#10-16) is never used and should be removed  
SafeMath.tryDiv(uint256,uint256) (XPSToken.sol#52-57) is never used and should be removed  
SafeMath.tryMod(uint256,uint256) (XPSToken.sol#64-69) is never used and should be removed  
SafeMath.tryMul(uint256,uint256) (XPSToken.sol#35-45) is never used and should be removed  
SafeMath.trySub(uint256,uint256) (XPSToken.sol#23-28) is never used and should be removed  
XPSToken.\_setupDecimals(uint8) (XPSToken.sol#1835-1837) is never used and should be removed  
XPSToken.isContract(address) (XPSToken.sol#1512-1516) is never used and should be removed  
Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code>

INFO:Detectors:

Pragma version^0.8.0 (XPSToken.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6

solc-0.8.0 is not recommended for deployment

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity>

INFO:Detectors:

XPSToken (XPSToken.sol#956-1855) should inherit from IERC20Metadata (XPSToken.sol#497-512)

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

INFO:Detectors:

Function IPancakePair.DOMAIN\_SEPARATOR() (XPSToken.sol#248) is not in mixedCase

Function IPancakePair.PERMIT\_TYPEHASH() (XPSToken.sol#249) is not in mixedCase

Function IPancakePair.MINIMUM\_LIQUIDITY() (XPSToken.sol#266) is not in mixedCase

Function IPancakeRouter01.WETH() (XPSToken.sol#286) is not in mixedCase

Variable XPSToken.WETH (XPSToken.sol#959) is not in mixedCase

Variable XPSToken.\_liquidityPoolFee (XPSToken.sol#975) is not in mixedCase

Variable XPSToken.\_marketingPoolFee (XPSToken.sol#976) is not in mixedCase

Variable XPSToken.\_burnFee (XPSToken.sol#977) is not in mixedCase

Variable XPSToken.\_communityRewardPoolFee (XPSToken.sol#978) is not in mixedCase

Variable XPSToken.\_commonFee (XPSToken.sol#981) is not in mixedCase

Variable XPSToken.\_specialFee (XPSToken.sol#983) is not in mixedCase

Variable XPSToken.\_votedCommonFee (XPSToken.sol#991) is not in mixedCase

Variable XPSToken.\_inVoteCommonFee (XPSToken.sol#992) is not in mixedCase

Variable XPSToken.\_votesCommonFee (XPSToken.sol#993) is not in mixedCase

Variable XPSToken.\_votedCommonFeeWallets (XPSToken.sol#994) is not in mixedCase

Variable XPSToken.\_currentOffsetVoteCommonFee (XPSToken.sol#995) is not in mixedCase

Variable XPSToken.\_votedSpecialFee (XPSToken.sol#997) is not in mixedCase

Variable XPSToken.\_inVoteSpecialFee (XPSToken.sol#998) is not in mixedCase

Variable XPSToken.\_votesSpecialFee (XPSToken.sol#999) is not in mixedCase

Variable XPSToken.\_votedSpecialFeeWallets (XPSToken.sol#1000) is not in mixedCase

Variable XPSToken.\_currentOffsetVoteSpecialFee (XPSToken.sol#1001) is not in mixedCase

Variable XPSToken.\_votedMarketingPoolWallet (XPSToken.sol#1004) is not in mixedCase

Variable XPSToken.\_inVoteMarketingPoolWallet (XPSToken.sol#1005) is not in mixedCase

Variable XPSToken.\_votesMarketingPoolWallet (XPSToken.sol#1006) is not in mixedCase

Variable XPSToken.\_votedMarketingPoolWalletWallets (XPSToken.sol#1007) is not in mixedCase

Variable XPSToken.\_currentOffsetVoteMarketingPoolWallet (XPSToken.sol#1008) is not in mixedCase

Variable XPSToken.\_votedCommunityRewardPoolWallet (XPSToken.sol#1010) is not in mixedCase

Variable XPSToken.\_inVoteCommunityRewardPoolWallet (XPSToken.sol#1011) is not in mixedCase

Variable XPSToken.\_votesCommunityRewardPoolWallet (XPSToken.sol#1012) is not in mixedCase

Variable XPSToken.\_votedCommunityRewardPoolWalletWallets (XPSToken.sol#1013) is not in mixedCase

Variable XPSToken.\_currentOffsetVoteCommunityRewardPoolWallet (XPSToken.sol#1014) is not in mixedCase

Variable XPSToken.\_liquidityTriggerAmount (XPSToken.sol#1020) is not in mixedCase

Variable XPSToken.\_marketingTriggerAmount (XPSToken.sol#1021) is not in mixedCase

Variable XPSToken.\_communityTriggerAmount (XPSToken.sol#1022) is not in mixedCase

Variable XPSToken.\_currentLiquidityTriggerAmount (XPSToken.sol#1025) is not in mixedCase

Variable XPSToken.\_currentMarketingTriggerAmount (XPSToken.sol#1026) is not in mixedCase

Variable XPSToken.\_currentCommunityTriggerAmount (XPSToken.sol#1027) is not in mixedCase

Variable XPSToken.\_totalLiquidityTriggerAmount (XPSToken.sol#1030) is not in mixedCase

Variable XPSToken.\_totalMarketingTriggerAmount (XPSToken.sol#1031) is not in mixedCase

Variable XPSToken.\_totalCommunityTriggerAmount (XPSToken.sol#1032) is not in mixedCase

Variable XPSToken.\_timeToReleaseFirstStep (XPSToken.sol#1049) is not in mixedCase

Variable XPSToken.\_timeToReleaseSecondStep (XPSToken.sol#1050) is not in mixedCase



Variable XPSToken.\_lockedTokensForTeam (XPSToken.sol#1051) is not in mixedCase  
Variable XPSToken.\_percentFromSupplyForTeam (XPSToken.sol#1052) is not in mixedCase  
Variable XPSToken.\_percentReleaseFirstStep (XPSToken.sol#1053) is not in mixedCase  
Variable XPSToken.\_percentReleaseSecondStep (XPSToken.sol#1054) is not in mixedCase  
Reference:  
<https://github.com/cryptic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions>  
INFO:Detectors:

Reentrancy in XPSToken.\_transfer(address,address,uint256) (XPSToken.sol#1441-1504):  
External calls:  
- swapAndCommunity(\_currentCommunityTriggerAmount) (XPSToken.sol#1486)  
- \_communityRewardPoolWallet.send(newBalance) (XPSToken.sol#1529)  
State variables written after the call(s):  
- \_currentCommunityTriggerAmount = 0 (XPSToken.sol#1487)

Reentrancy in XPSToken.\_transfer(address,address,uint256) (XPSToken.sol#1441-1504):  
External calls:  
- swapAndCommunity(\_currentCommunityTriggerAmount) (XPSToken.sol#1486)  
- \_communityRewardPoolWallet.send(newBalance) (XPSToken.sol#1529)  
- swapAndMarketing(\_currentMarketingTriggerAmount) (XPSToken.sol#1491)  
- \_marketingPoolWallet.send(newBalance) (XPSToken.sol#1545)  
State variables written after the call(s):  
- swapAndMarketing(\_currentMarketingTriggerAmount) (XPSToken.sol#1491)  
- \_allowances[owner][spender] = amount (XPSToken.sol#1824)  
- \_currentMarketingTriggerAmount = 0 (XPSToken.sol#1492)  
- swapAndMarketing(\_currentMarketingTriggerAmount) (XPSToken.sol#1491)  
- inSwapAndMarketing = true (XPSToken.sol#1079)  
- inSwapAndMarketing = false (XPSToken.sol#1081)  
Event emitted after the call(s):  
- Approval(owner,spender,amount) (XPSToken.sol#1825)  
- swapAndMarketing(\_currentMarketingTriggerAmount) (XPSToken.sol#1491)  
- SwapAndMarketing(amount,newBalance) (XPSToken.sol#1547)  
- swapAndMarketing(\_currentMarketingTriggerAmount) (XPSToken.sol#1491)

Reentrancy in XPSToken.\_transfer(address,address,uint256) (XPSToken.sol#1441-1504):  
External calls:  
- swapAndCommunity(\_currentCommunityTriggerAmount) (XPSToken.sol#1486)  
- \_communityRewardPoolWallet.send(newBalance) (XPSToken.sol#1529)  
- swapAndMarketing(\_currentMarketingTriggerAmount) (XPSToken.sol#1491)  
- \_marketingPoolWallet.send(newBalance) (XPSToken.sol#1545)  
External calls sending eth:  
- swapAndCommunity(\_currentCommunityTriggerAmount) (XPSToken.sol#1486)  
- \_communityRewardPoolWallet.send(newBalance) (XPSToken.sol#1529)  
- swapAndMarketing(\_currentMarketingTriggerAmount) (XPSToken.sol#1491)  
- \_marketingPoolWallet.send(newBalance) (XPSToken.sol#1545)  
- swapAndLiquify(\_currentLiquidityTriggerAmount) (XPSToken.sol#1496)  
- pancakeswapV2Router.addLiquidityETH{value:  
ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (XPSToken.sol#1599-1606)  
State variables written after the call(s):  
- swapAndLiquify(\_currentLiquidityTriggerAmount) (XPSToken.sol#1496)  
- \_allowances[owner][spender] = amount (XPSToken.sol#1824)  
- \_balances[recipient] = \_balances[recipient].add(totalSendAmount) (XPSToken.sol#1502)  
- \_currentLiquidityTriggerAmount = 0 (XPSToken.sol#1497)  
- swapAndLiquify(\_currentLiquidityTriggerAmount) (XPSToken.sol#1496)  
- inSwapAndLiquify = true (XPSToken.sol#1059)  
- inSwapAndLiquify = false (XPSToken.sol#1061)  
Event emitted after the call(s):  
- Approval(owner,spender,amount) (XPSToken.sol#1825)  
- swapAndLiquify(\_currentLiquidityTriggerAmount) (XPSToken.sol#1496)  
- SwapAndLiquify(half,newBalance,otherHalf) (XPSToken.sol#1573)  
- swapAndLiquify(\_currentLiquidityTriggerAmount) (XPSToken.sol#1496)  
- Transfer(sender,recipient,amount) (XPSToken.sol#1503)

Reentrancy in XPSToken.swapAndCommunity(uint256) (XPSToken.sol#1518-1532):  
External calls:  
- \_communityRewardPoolWallet.send(newBalance) (XPSToken.sol#1529)  
Event emitted after the call(s):  
- SwapAndCommunity(amount,newBalance) (XPSToken.sol#1531)

Reentrancy in XPSToken.swapAndMarketing(uint256) (XPSToken.sol#1534-1548):

External calls:

- `_marketingPoolWallet.send(newBalance)` (XPSToken.sol#1545)

Event emitted after the call(s):

- `SwapAndMarketing(amount,newBalance)` (XPSToken.sol#1547)

Reentrancy in `XPSToken.transferFrom(address,address,uint256)` (XPSToken.sol#1725-1729):

External calls:

- `_transfer(sender,recipient,amount)` (XPSToken.sol#1726)
  - `_marketingPoolWallet.send(newBalance)` (XPSToken.sol#1545)
  - `_communityRewardPoolWallet.send(newBalance)` (XPSToken.sol#1529)

External calls sending eth:

- `_transfer(sender,recipient,amount)` (XPSToken.sol#1726)
  - `pancakeswapV2Router.addLiquidityETH{value: ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp)` (XPSToken.sol#1599-1606)
    - `_marketingPoolWallet.send(newBalance)` (XPSToken.sol#1545)
    - `_communityRewardPoolWallet.send(newBalance)` (XPSToken.sol#1529)

State variables written after the call(s):

- `_approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,BEP20: transfer amount exceeds allowance))` (XPSToken.sol#1727)
  - `_allowances[owner][spender] = amount` (XPSToken.sol#1824)

Event emitted after the call(s):

- `Approval(owner,spender,amount)` (XPSToken.sol#1825)
  - `_approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,BEP20: transfer amount exceeds allowance))` (XPSToken.sol#1727)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4>

INFO:Detectors:

Variable

`IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountA` Desired (XPSToken.sol#291) is too similar to

`IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountB` Desired (XPSToken.sol#292)

Variable `XPSToken._votedCommonFee` (XPSToken.sol#991) is too similar to `XPSToken._votesCommonFee` (XPSToken.sol#993)

Variable `XPSToken._votedCommunityRewardPoolWallet` (XPSToken.sol#1010) is too similar to `XPSToken._votesCommunityRewardPoolWallet` (XPSToken.sol#1012)

Variable `XPSToken._votedMarketingPoolWallet` (XPSToken.sol#1004) is too similar to `XPSToken._votesMarketingPoolWallet` (XPSToken.sol#1006)

Variable `XPSToken._votedSpecialFee` (XPSToken.sol#997) is too similar to `XPSToken._votesSpecialFee` (XPSToken.sol#999)

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

INFO:Detectors:

`XPSToken._maxCommonFee` (XPSToken.sol#982) is never used in `XPSToken` (XPSToken.sol#956-1855)

`XPSToken._maxSpecialFee` (XPSToken.sol#984) is never used in `XPSToken` (XPSToken.sol#956-1855)

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

INFO:Detectors:

`XPSToken._burnFee` (XPSToken.sol#977) should be constant

`XPSToken._communityRewardPoolFee` (XPSToken.sol#978) should be constant

`XPSToken._communityTriggerAmount` (XPSToken.sol#1022) should be constant

`XPSToken._liquidityPoolFee` (XPSToken.sol#975) should be constant

`XPSToken._liquidityTriggerAmount` (XPSToken.sol#1020) should be constant

`XPSToken._marketingPoolFee` (XPSToken.sol#976) should be constant

`XPSToken._marketingTriggerAmount` (XPSToken.sol#1021) should be constant

`XPSToken._maxCommonFee` (XPSToken.sol#982) should be constant

`XPSToken._maxSpecialFee` (XPSToken.sol#984) should be constant

`XPSToken._minAcceptedVotes` (XPSToken.sol#987) should be constant

`XPSToken._minDeclinedVotes` (XPSToken.sol#988) should be constant

`XPSToken._percentFromSupplyForTeam` (XPSToken.sol#1052) should be constant

`XPSToken._percentReleaseFirstStep` (XPSToken.sol#1053) should be constant

`XPSToken._percentReleaseSecondStep` (XPSToken.sol#1054) should be constant

`XPSToken._swapOnTransaction` (XPSToken.sol#1017) should be constant

Reference:

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

INFO:Detectors:

`renounceOwnership()` should be declared external:

- `Ownable.renounceOwnership()` (XPSToken.sol#580-582)

`transferOwnership(address)` should be declared external:

- Ownable.transferOwnership(address) (XPSToken.sol#588-591)

name() should be declared external:

- ERC20.name() (XPSToken.sol#651-653)

symbol() should be declared external:

- ERC20.symbol() (XPSToken.sol#659-661)

decimals() should be declared external:

- ERC20.decimals() (XPSToken.sol#676-678)

totalSupply() should be declared external:

- ERC20.totalSupply() (XPSToken.sol#683-685)
- XPSToken.totalSupply() (XPSToken.sol#1669-1671)

balanceOf(address) should be declared external:

- ERC20.balanceOf(address) (XPSToken.sol#690-692)
- XPSToken.balanceOf(address) (XPSToken.sol#1676-1678)

transfer(address,uint256) should be declared external:

- ERC20.transfer(address,uint256) (XPSToken.sol#702-705)
- XPSToken.transfer(address,uint256) (XPSToken.sol#1688-1691)

allowance(address,address) should be declared external:

- ERC20.allowance(address,address) (XPSToken.sol#710-712)
- XPSToken.allowance(address,address) (XPSToken.sol#1696-1698)

approve(address,uint256) should be declared external:

- ERC20.approve(address,uint256) (XPSToken.sol#721-724)
- XPSToken.approve(address,uint256) (XPSToken.sol#1707-1710)

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

- ERC20.transferFrom(address,address,uint256) (XPSToken.sol#739-753)
- XPSToken.transferFrom(address,address,uint256) (XPSToken.sol#1725-1729)

increaseAllowance(address,uint256) should be declared external:

- ERC20.increaseAllowance(address,uint256) (XPSToken.sol#767-770)

decreaseAllowance(address,uint256) should be declared external:

- ERC20.decreaseAllowance(address,uint256) (XPSToken.sol#786-794)

includeInFee(address) should be declared external:

- XPSToken.includeInFee(address) (XPSToken.sol#1613-1615)

includeInSpecialFee(address) should be declared external:

- XPSToken.includeInSpecialFee(address) (XPSToken.sol#1622-1624)

excludeFromSpecialFee(address) should be declared external:

- XPSToken.excludeFromSpecialFee(address) (XPSToken.sol#1626-1628)

name() should be declared external:

- XPSToken.name() (XPSToken.sol#1637-1639)

symbol() should be declared external:

- XPSToken.symbol() (XPSToken.sol#1645-1647)

decimals() should be declared external:

- XPSToken.decimals() (XPSToken.sol#1662-1664)

increaseAllowance(address,uint256) should be declared external:

- XPSToken.increaseAllowance(address,uint256) (XPSToken.sol#1743-1746)

decreaseAllowance(address,uint256) should be declared external:

- XPSToken.decreaseAllowance(address,uint256) (XPSToken.sol#1762-1765)

Reference:

<https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external>

INFO:Slither:XPSToken.sol analyzed (11 contracts with 75 detectors), 149 result(s) found

INFO:Slither:Use <https://crytic.io/> to get access to additional detectors and Github integration



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](mailto:audit@EtherAuthority.io)**