ETHOS AUDIT

PVP

Crypto & Cyber Security Audit Report

October 2021

POWERED BY

DARKSCOPE

Contents

Executive Summary

Audit Details

Project Name	PVP
Codebase	https://bscscan.com/address/0x559752951e103402c8c4fe9af874f68ce0845bf0#code
Source Code	PVP.sol
Initial Audit Date	Oct 11, 2021
Methodology	Manual, Automated

Methodology

This audit's objectives are to evaluate:

- Security-related issues
- Code quality
- Relevant documentation
- Adherence to specifications
- Adherence to best practices
- Cyber-security risks

This audit examines the possibility of issues existing along the following vectors (but not limited to):

- Single & Cross-Function Reentrancy
- Front Running (Transaction Order Dependence)
- Timestamp dependence
- Integer Overflow and Underflow
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Number rounding errors
- DoS with (Unexpected) Revert
- DoS with Block Gas Limit
- Insufficient gas griefing
- Forcibly sending native currency
- Logical oversights
- Access control
- Centralization of power
- Logic-Specification Contradiction
- Functionality duplication
- Malicious token minting

The code review conducted for this audit follows the following structure:

1. Review of specifications, documentation to assess smart contract functionality
2. Manual, line-by-line review of code
3. Code's adherence to functionality as presented by documentation
4. Automated tool-driven review of smart contract functionality
5. Assess adherence to best practices
6. Provide actionable recommendations

Risk Levels

LOW

The issue is informational and does not pose an immediate risk but is relevant to security best practices.

MEDIUM

The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.

HIGH

The issue puts a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or is reasonably likely to lead to moderate financial impact.

EXTREME

The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for clients and users.

Issues Summary

Severity	Unresolved	Acknowledged	Resolved
Extreme	0	0	0
High	0	0	0
Medium	0	0	0
Low	2	0	0

Contract Details

Contract ID	0x559752951E103402C8c4Fe9Af874F68cE0845bF0
Network	BSC
Language	Solidity
Compiler	v0.8.4+commit.c7e474f2
Verification Date	Oct. 7, 2021
Contract Type	BEP-20 Token
Libraries	Custom

Token Details

Contract Name	PVP
Symbol	$PVP
Decimals	9
Total Supply	40,000,000

Functions

Function	Parameters	Visibility	Modifiers	Returns	Requires	Events	Called By
constructor						Transfer	
name		public	view	string			
symbol		public	view	string			
decimals		public	view	uint8			
totalSupply		public	view	uint256			
balanceOf	address account	public	view	uint256			
transfer	address recipient, uint256 amount	public		bool			
allowance	address owner, address spender	public	view	uint256			
approve	address spender, uint256 amount	public		bool			
transferFrom	address sender, address recipient, uint256 amount	public		bool			
increaseAllowance	address spender, uint256 addedValue	public		bool			
decreaseAllowance	address spender, uint256 subtractedValue	public		bool			
isExcludedFromReward	address account	public	view	bool			
totalFees		public	view	uint256			
minimumTokensBeforeSwapAmount		public	view	uint256			
buyBackSellLimitAmount		public	view	uint256			
deliver	uint256 tAmount	public					
reflectionFromToken	uint256 tAmount, bool deductTransferFee	public	view	uint256			
tokenFromReflection	uint256 rAmount	public	view	uint256			
excludeFromReward	address account	public	onlyOwner				
includeInReward	address account	public	onlyOwner		(_isExcluded[account], "Account is not excluded")		
_approve	address owner, address spender, uint256 amount	private			(owner != address(0), "ERC20: approve from the zero address") (spender != address(0), "ERC20: approve to the zero address")	Approval	
_transfer	address from, address to, uint256 amount	private			(from != address(0), "ERC20: transfer from the zero address") (to != address(0), "ERC20: transfer to the zero address") (amount > 0, "Transfer amount must be greater than zero") require(amount <= _maxTxAmount, "Transfer amount exceeds the maxTxAmount.")		
swapTokens	uint256 contractTokenBalance	private	lockTheSwap				
swapTokensForEth	uint256 tokenAmount	private				SwapTokensForETH	
swapETHForTokens	uint256 amoun	private				SwapETHForTokens	
addLiquidity	uint256 tokenAmount, uint256 ethAmount	private					
_tokenTransfer	address sender, address recipient,	private					

	uint256 amount,bool takeFee				
_transferStandard	address sender, address recipient, uint256 tAmount	private			Transfer
_transferToExcluded	address sender, address recipient, uint256 tAmount	private			Transfer
_transferFromExcluded	address sender, address recipient, uint256 tAmount	private			Transfer
_transferBothExcluded	address sender, address recipient, uint256 tAmount	private			Transfer
_reflectFee	uint256 rFee, uint256 tFee	private			
_getValues	uint256 tAmount	private	view	uint256, uint256, uint256, uint256, uint256, uint256	
_getTValues	uint256 tAmount	private	view	uint256, uint256, uint256	
_getRValues	uint256 tAmount, uint256 tFee, uint256 tLiquidity, uint256 currentRate	private	view pure	uint256, uint256, uint256	
_getRate		private	view	uint256	
_getCurrentSupply		private	view	uint256, uint256	
_takeLiquidity	uint256 tLiquidity	private			
calculateTaxFee	uint256 _amount	private	view	uint256	
calculateLiquidityFee	uint256 _amount	private	view	uint256	
removeAllFee		private			
restoreAllFee		private			
isExcludedFromFee	address account	public	view	bool	
excludeFromFee	address account	public	onlyOwner		
includeInFee	address account	public	onlyOwner		
_getSellBnBAmount	uint256 tokenAmount	private	view	uint256	
_removeOldSellHistories		private			
SetBuyBackMaxTimeForHistories	uint256 newMinutes	external	onlyOwner		
SetBuyBackDivisor	uint256 newDivisor	external	onlyOwner		
GetBuyBackTimeInterval		public	view	uint256	
SetBuyBackTimeInterval	uint256 newMinutes	external	onlyOwner		
SetBuyBackRangeRate	uint256 newPercent	external	onlyOwner		
GetSwapMinutes		public	view	uint256	
SetSwapMinutes	uint256 newMinutes	external	onlyOwner		
setTaxFeePercent	uint256 taxFee	external	onlyOwner		
setBuyFee	uint256 buyTaxFee, uint256 buyLiquidityFee	external	onlyOwner		
setSellFee	uint256 sellTaxFee, uint256 sellLiquidityFee	external	onlyOwner		
setLiquidityFeePercent	uint256 liquidityFee	external	onlyOwner		
setBuyBackSellLimit	uint256 buyBackSellSetLimit	external	onlyOwner		
setMaxTxAmount	uint256 maxTxAmount	external	onlyOwner		
setMarketingDivisor	uint256 divisor	external	onlyOwner		

setNumTokensSellToAddToBuyBack	uint256 _minimumTokensBeforeSwap	external	onlyOwner		
setMarketingAddress	address _marketingAddress	external	onlyOwner		
setSwapAndLiquifyEnabled	bool _enabled	public	onlyOwner		SwapAndLiquifyEnabledUpdated
setBuyBackEnabled	bool _enabled	public	onlyOwner		BuyBackEnabledUpdated
setAutoBuyBackEnabled	bool _enabled	public	onlyOwner		AutoBuyBackEnabledUpdated
prepareForPreSale		external	onlyOwner		
afterPreSale		external	onlyOwner		
transferToAddressETH	address payable recipient, uint256 amount	private			
changeRouterVersion	address _router	public	onlyOwner	address _pair	
transferForeignToken	address _token, address _to	public	onlyOwner	bool _sent	
Sweep		external	onlyOwner		
setAddressFee	address _address, bool _enable, uint256 _addressTaxFee, uint256 _addressLiquidityFee	external	onlyOwner		
setBuyAddressFee	address _address, bool _enable, uint256 _addressTaxFee, uint256 _addressLiquidityFee	external	onlyOwner		
setSellAddressFee	address _address, bool _enable, uint256 _addressTaxFee, uint256 _addressLiquidityFee	external	onlyOwner		

Global Variables

Variable	Type	Visibilty	Read by Functions	Written by Functions
marketingAddress	address payable	public	swapTokens	setMarketingAddress
deadAddress	address	public	swapETHForTokens	
_rOwned	mapping (address => uint256)	private	balanceOf, excludeFromReward, _getCurrentSupply	constructor,deliver, _transferStandard, _transferToExcluded, _transferFromExcluded, _transferBothExcluded, _takeLiquidity
_tOwned	mapping (address => uint256)	private	balanceOf, _getCurrentSupply	excludeFromReward, includeInReward, _transferToExcluded, _transferFromExcluded, _transferBothExcluded, _takeLiquidity
_allowances	mapping (address => mapping (address => uint256))	private	allowance, transferFrom, increaseAllowance, decreaseAllowance,	_approve,
_isExcludedFromFee	mapping (address => bool)	private	_transfer, isExcludedFromFee, excludeFromFee, includeInFee	constructor
_isExcluded	mapping (address => bool)	private	balanceOf, isExcludedFromReward, deliver, excludeFromReward, includeInReward, _tokenTransfer, _takeLiquidity,	excludeFromReward, includeInReward
_excluded	address[]	private	_getCurrentSupply,	excludeFromReward, includeInReward,
MAX	uint256	private		
_tTotal	uint256	private	constructor, totalSupply, reflectionFromToken, _getCurrentSupply	
_rTotal	uint256	private	constructor, tokenFromReflection, _getCurrentSupply,	deliver, _reflectFee

_tFeeTotal	_tFeeTotal	private	totalFees,	deliver, _reflectFee,
_name	string	private	name	
_symbol	string	private	symbol	
_decimals	uint8	private	decimals	
_taxFee	uint256	public	calculateTaxFee, removeAllFee, restoreAllFee	_transfer, setTaxFeePercent,prepareForPreSale, afterPreSale, setAddressFee,
_previousTaxFee	uint256	private	restoreAllFee	removeAllFee,
_liquidityFee	uint256	public	calculateLiquidityFee, removeAllFee	_transfer, restoreAllFee, setLiquidityFeePercent, prepareForPreSale, afterPreSale, setAddressFee,
_previousLiquidityFee	uint256	private	restoreAllFee	removeAllFee,
_buyTaxFee	uint256	public	_transfer	setBuyFee, setBuyAddressFee
_buyLiquidityFee	uint256	public	_transfer	setBuyFee, setBuyAddressFee
_sellTaxFee	uint256	public	_transfer	setSellFee, setSellAddressFee
_sellLiquidityFee	uint256	public	_transfer	setSellFee, setSellAddressFee
_startTimeForSwap	uint256	public	_transfer	constructor, _transfer
_intervalMinutesForSwap	uint256	public	_transfer, GetSwapMinutes	SetSwapMinutes,
_buyBackRangeRate	uint256	public	_transfer,	SetBuyBackRangeRate,
_addressFees	mapping (address => AddressFee)	public	_transfer,	setAddressFee, setBuyAddressFee, setSellAddressFee
marketingDivisor	uint256	public	swapTokens,	setMarketingDivisor,
_maxTxAmount	uint256	public	_transfer,	setMaxTxAmount, prepareForPreSale, afterPreSale,
minimumTokensBeforeSwap	uint256	private	_transfer, minimumTokensBeforeSwapAmount	
buyBackSellLimit	uint256	public	_transfer, buyBackSellLimitAmount,	
_sellHistories	SellHistories[]	public	_removeOldSellHistories,	_transfer, _removeOldSellHistories
_isAutoBuyBack	bool	public	_transfer,	setAutoBuyBackEnabled,
_buyBackDivisor	uint256	public	_transfer,	SetBuyBackDivisor
_buyBackTimeInterval	uint256	public	_transfer, GetBuyBackTimeInterval	SetBuyBackTimeInterval
_buyBackMaxTimeForHistories	uint256	public	_removeOldSellHistories,	SetBuyBackMaxTimeForHistories,
uniswapV2Router	IUniswapV2Router02	public	swapTokensForEth, swapETHForTokens, addLiquidity, _getSellBnBAmount,	constructor, changeRouterVersion
uniswapV2Pair	address	public	_transfer,	constructor, changeRouterVersion
inSwapAndLiquify	bool		_transfer	lockTheSwap
swapAndLiquifyEnabled	bool	public	_transfer	setSwapAndLiquifyEnabled
buyBackEnabled	bool	public	_transfer	setBuyBackEnabled
_isEnabledBuyBackAndBurn	bool	public		

Balance Updates

Function	Changes
constructor	_balances[msg.sender] = _totalSupply
_transfer	_balances[sender] = _balances[sender].sub(amount…) _balances[recipient] = _balances[recipient].add(amount)

Detailed Findings

Block values as a proxy for time

Severity: Low **Status**: Open

Description: Contracts often need access to time values to perform certain types of functionality. Values such as block.timestamp, and block.number can give you a sense of the current time or a time delta. However, they are not safe to use for most purposes.

Risk: In the case of block.timestamp, developers often attempt to use it to trigger time-dependent events.

Recommendation: Developers should write smart contracts with the notion that block values are not precise, and the use of them can lead to unexpected effects. Alternatively, they may make use of oracles.

Examples for this are in lines 706 / 715 / 716 / 730 and many more of the contract.

https://swcregistry.io/docs/SWC-116#time-locksol

DoS With Block Gas Limit

Severity: Low **Status**: Open

Description: Programming patterns that are harmless in centralized applications can lead to Denial of Service conditions in smart contracts when the cost of executing a function exceeds the block gas limit. Modifying an array of unknown size that increases in size over time can lead to such a Denial of Service condition.

Risk: In the case of block.timestamp, developers often attempt to use it to trigger time-dependent events.

Recommendation: Caution is advised when you expect to have large arrays that grow over time. Actions that require looping across the entire data structure should be avoided.

If you absolutely must loop over an array of unknown size, then you should plan for it to potentially take multiple blocks, and therefore require multiple transactions.

Examples for this are in lines 670 / 733 / 970 / 1042 / 1055 of the contract.

https://swcregistry.io/docs/SWC-128#dos-addresssol

Automated Analysis

The automated analysis was completed by running Slither on the codebase. A total of **198 issues** were detected. However, **none of the issues was severe** enough to be considered relevant to the security of the smart contract. This analysis continues to page 23.

Functions that send Ether to arbitrary destinations

```
PVP.swapETHForTokens(uint256) (PVP.sol#846-861) sends eth to arbitrary user
    Dangerous calls:
    - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value:
amount}(0,path,deadAddress,block.timestamp.add(300)) (PVP.sol#853-858)
PVP.addLiquidity(uint256,uint256) (PVP.sol#863-876) sends eth to arbitrary user
    Dangerous calls:
    - uniswapV2Router.addLiquidityETH{value:
ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (PVP.sol#868-875)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-
arbitrary-destinations
```

Weak PRNG

```
PVP._transfer(address,address,uint256) (PVP.sol#689-806) uses a weak PRNG: "_bBSLimit = _bBSLimitMin +
uint256(keccak256(bytes)(abi.encodePacked(block.timestamp,block.difficulty))) % (_bBSLimitMax -
_bBSLimitMin + 1) (PVP.sol#750)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG
```

Reentrancy

```
Reentrancy in PVP._transfer(address,address,uint256) (PVP.sol#689-806):
    External calls:
    - swapTokens(contractTokenBalance) (PVP.sol#718)
        -
uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),blo
ck.timestamp) (PVP.sol#835-841)
    External calls sending eth:
    - swapTokens(contractTokenBalance) (PVP.sol#718)
        - recipient.transfer(amount) (PVP.sol#1159)
    State variables written after the call(s):
    - _removeOldSellHistories() (PVP.sol#745)
        - _sellHistories[i].time = _sellHistories[j].time (PVP.sol#1046)
        - _sellHistories[i].bnbAmount = _sellHistories[j].bnbAmount (PVP.sol#1047)
        - _sellHistories.pop() (PVP.sol#1057)
Reentrancy in PVP._transfer(address,address,uint256) (PVP.sol#689-806):
    External calls:
    - swapTokens(contractTokenBalance) (PVP.sol#718)
        -
uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),blo
ck.timestamp) (PVP.sol#835-841)
    - buyBackTokens(_bBSLimit) (PVP.sol#753)
        - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value:
amount}(0,path,deadAddress,block.timestamp.add(300)) (PVP.sol#853-858)
    External calls sending eth:
    - swapTokens(contractTokenBalance) (PVP.sol#718)
        - recipient.transfer(amount) (PVP.sol#1159)
    - buyBackTokens(_bBSLimit) (PVP.sol#753)
```

```
        - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value:
amount}(0,path,deadAddress,block.timestamp.add(300)) (PVP.sol#853-858)
    State variables written after the call(s):
    - _tokenTransfer(from,to,amount,takeFee) (PVP.sol#805)
        - _rOwned[address(this)] = _rOwned[address(this)].add(rLiquidity) (PVP.sol#982)
        - _rOwned[sender] = _rOwned[sender].sub(rAmount) (PVP.sol#898)
        - _rOwned[sender] = _rOwned[sender].sub(rAmount) (PVP.sol#907)
        - _rOwned[sender] = _rOwned[sender].sub(rAmount) (PVP.sol#918)
        - _rOwned[sender] = _rOwned[sender].sub(rAmount) (PVP.sol#928)
        - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (PVP.sol#899)
        - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (PVP.sol#909)
        - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (PVP.sol#919)
        - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (PVP.sol#930)
    - _tokenTransfer(from,to,amount,takeFee) (PVP.sol#805)
        - _rTotal = _rTotal.sub(rFee) (PVP.sol#937)
    - _tokenTransfer(from,to,amount,takeFee) (PVP.sol#805)
        - _tOwned[address(this)] = _tOwned[address(this)].add(tLiquidity) (PVP.sol#984)
        - _tOwned[sender] = _tOwned[sender].sub(tAmount) (PVP.sol#917)
        - _tOwned[sender] = _tOwned[sender].sub(tAmount) (PVP.sol#927)
        - _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (PVP.sol#908)
        - _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (PVP.sol#929)
    - buyBackTokens(_bBSLimit) (PVP.sol#753)
        - inSwapAndLiquify = true (PVP.sol#535)
        - inSwapAndLiquify = false (PVP.sol#537)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
```

Uninitialized local variables

```
PVP._transfer(address,address,uint256).sellHistory (PVP.sol#705) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
```

Unused return

```
PVP.addLiquidity(uint256,uint256) (PVP.sol#863-876) ignores return value by
uniswapV2Router.addLiquidityETH{value:
ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (PVP.sol#868-875)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

Local variable shadowing

```
PVP.allowance(address,address).owner (PVP.sol#590) shadows:
    - Ownable.owner() (PVP.sol#180-182) (function)
PVP._approve(address,address,uint256).owner (PVP.sol#681) shadows:
    - Ownable.owner() (PVP.sol#180-182) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
```

Missing events arithmetic

```
PVP.SetBuyBackMaxTimeForHistories(uint256) (PVP.sol#1062-1064) should emit an event for:
    - _buyBackMaxTimeForHistories = newMinutes * 60 (PVP.sol#1063)
PVP.SetBuyBackDivisor(uint256) (PVP.sol#1066-1068) should emit an event for:
    - _buyBackDivisor = newDivisor (PVP.sol#1067)
PVP.SetBuyBackTimeInterval(uint256) (PVP.sol#1074-1076) should emit an event for:
    - _buyBackTimeInterval = newMinutes * 60 (PVP.sol#1075)
PVP.SetBuyBackRangeRate(uint256) (PVP.sol#1078-1081) should emit an event for:
    - _buyBackRangeRate = newPercent (PVP.sol#1080)
PVP.SetSwapMinutes(uint256) (PVP.sol#1087-1089) should emit an event for:
    - _intervalMinutesForSwap = newMinutes * 60 (PVP.sol#1088)
```

```
PVP.setTaxFeePercent(uint256) (PVP.sol#1091-1093) should emit an event for:
    - _taxFee = taxFee (PVP.sol#1092)
PVP.setBuyFee(uint256,uint256) (PVP.sol#1095-1098) should emit an event for:
    - _buyTaxFee = buyTaxFee (PVP.sol#1096)
    - _buyLiquidityFee = buyLiquidityFee (PVP.sol#1097)
PVP.setSellFee(uint256,uint256) (PVP.sol#1100-1103) should emit an event for:
    - _sellTaxFee = sellTaxFee (PVP.sol#1101)
    - _sellLiquidityFee = sellLiquidityFee (PVP.sol#1102)
PVP.setLiquidityFeePercent(uint256) (PVP.sol#1105-1107) should emit an event for:
    - _liquidityFee = liquidityFee (PVP.sol#1106)
PVP.setBuyBackSellLimit(uint256) (PVP.sol#1109-1111) should emit an event for:
    - buyBackSellLimit = buyBackSellSetLimit (PVP.sol#1110)
PVP.setMaxTxAmount(uint256) (PVP.sol#1113-1115) should emit an event for:
    - _maxTxAmount = maxTxAmount (PVP.sol#1114)
PVP.setMarketingDivisor(uint256) (PVP.sol#1117-1119) should emit an event for:
    - marketingDivisor = divisor (PVP.sol#1118)
PVP.setNumTokensSellToAddToBuyBack(uint256) (PVP.sol#1121-1123) should emit an event for:
    - minimumTokensBeforeSwap = _minimumTokensBeforeSwap (PVP.sol#1122)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
```

Missing zero address validation

```
PVP.setMarketingAddress(address)._marketingAddress (PVP.sol#1125) lacks a zero-check on :
        - marketingAddress = address(_marketingAddress) (PVP.sol#1126)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-
validation
```

Reentrancy-2

```
Reentrancy in PVP._transfer(address,address,uint256) (PVP.sol#689-806):
    External calls:
    - swapTokens(contractTokenBalance) (PVP.sol#718)
        -
uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),blo
ck.timestamp) (PVP.sol#835-841)
    - buyBackTokens(_bBSLimit) (PVP.sol#753)
        - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value:
amount}(0,path,deadAddress,block.timestamp.add(300)) (PVP.sol#853-858)
    External calls sending eth:
    - swapTokens(contractTokenBalance) (PVP.sol#718)
        - recipient.transfer(amount) (PVP.sol#1159)
    - buyBackTokens(_bBSLimit) (PVP.sol#753)
        - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value:
amount}(0,path,deadAddress,block.timestamp.add(300)) (PVP.sol#853-858)
    State variables written after the call(s):
    - removeAllFee() (PVP.sol#769)
        - _liquidityFee = 0 (PVP.sol#1006)
    - _liquidityFee = _buyLiquidityFee (PVP.sol#771)
    - removeAllFee() (PVP.sol#775)
        - _liquidityFee = 0 (PVP.sol#1006)
    - _liquidityFee = _sellLiquidityFee (PVP.sol#777)
    - removeAllFee() (PVP.sol#782)
        - _liquidityFee = 0 (PVP.sol#1006)
    - _liquidityFee = _addressFees[from]._liquidityFee (PVP.sol#784)
    - _liquidityFee = _addressFees[from]._sellLiquidityFee (PVP.sol#789)
    - removeAllFee() (PVP.sol#796)
        - _liquidityFee = 0 (PVP.sol#1006)
    - _liquidityFee = _addressFees[to]._buyLiquidityFee (PVP.sol#799)
    - _tokenTransfer(from,to,amount,takeFee) (PVP.sol#805)
        - _liquidityFee = _previousLiquidityFee (PVP.sol#1011)
```

```
        - _liquidityFee = 0 (PVP.sol#1006)
    - removeAllFee() (PVP.sol#769)
        - _previousLiquidityFee = _liquidityFee (PVP.sol#1003)
    - removeAllFee() (PVP.sol#775)
        - _previousLiquidityFee = _liquidityFee (PVP.sol#1003)
    - removeAllFee() (PVP.sol#782)
        - _previousLiquidityFee = _liquidityFee (PVP.sol#1003)
    - removeAllFee() (PVP.sol#796)
        - _previousLiquidityFee = _liquidityFee (PVP.sol#1003)
    - _tokenTransfer(from,to,amount,takeFee) (PVP.sol#805)
        - _previousLiquidityFee = _liquidityFee (PVP.sol#1003)
    - removeAllFee() (PVP.sol#769)
        - _previousTaxFee = _taxFee (PVP.sol#1002)
    - removeAllFee() (PVP.sol#775)
        - _previousTaxFee = _taxFee (PVP.sol#1002)
    - removeAllFee() (PVP.sol#782)
        - _previousTaxFee = _taxFee (PVP.sol#1002)
    - removeAllFee() (PVP.sol#796)
        - _previousTaxFee = _taxFee (PVP.sol#1002)
    - _tokenTransfer(from,to,amount,takeFee) (PVP.sol#805)
        - _previousTaxFee = _taxFee (PVP.sol#1002)
    - _tokenTransfer(from,to,amount,takeFee) (PVP.sol#805)
        - _tFeeTotal = _tFeeTotal.add(tFee) (PVP.sol#938)
    - removeAllFee() (PVP.sol#769)
        - _taxFee = 0 (PVP.sol#1005)
    - _taxFee = _buyTaxFee (PVP.sol#770)
    - removeAllFee() (PVP.sol#775)
        - _taxFee = 0 (PVP.sol#1005)
    - _taxFee = _sellTaxFee (PVP.sol#776)
    - removeAllFee() (PVP.sol#782)
        - _taxFee = 0 (PVP.sol#1005)
    - _taxFee = _addressFees[from]._taxFee (PVP.sol#783)
    - _taxFee = _addressFees[from]._sellTaxFee (PVP.sol#788)
    - removeAllFee() (PVP.sol#796)
        - _taxFee = 0 (PVP.sol#1005)
    - _taxFee = _addressFees[to]._buyTaxFee (PVP.sol#798)
    - _tokenTransfer(from,to,amount,takeFee) (PVP.sol#805)
        - _taxFee = _previousTaxFee (PVP.sol#1010)
        - _taxFee = 0 (PVP.sol#1005)
Reentrancy in PVP.changeRouterVersion(address) (PVP.sol#1162-1175):
    External calls:
    - _pair =
IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.WETH())
(PVP.sol#1168-1169)
    State variables written after the call(s):
    - uniswapV2Pair = _pair (PVP.sol#1171)
    - uniswapV2Router = _uniswapV2Router (PVP.sol#1174)
Reentrancy in PVP.constructor() (PVP.sol#540-562):
    External calls:
    - uniswapV2Pair =
IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.WETH())
(PVP.sol#550-551)
    State variables written after the call(s):
    - _isExcludedFromFee[owner()] = true (PVP.sol#556)
    - _isExcludedFromFee[address(this)] = true (PVP.sol#557)
    - _startTimeForSwap = block.timestamp (PVP.sol#559)
    - uniswapV2Router = _uniswapV2Router (PVP.sol#553)
Reentrancy in PVP.transferFrom(address,address,uint256) (PVP.sol#599-603):
    External calls:
    - _transfer(sender,recipient,amount) (PVP.sol#600)
        - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value:
amount}(0,path,deadAddress,block.timestamp.add(300)) (PVP.sol#853-858)
        -
uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),blo
ck.timestamp) (PVP.sol#835-841)
```

```
    External calls sending eth:
    - _transfer(sender,recipient,amount) (PVP.sol#600)
        - recipient.transfer(amount) (PVP.sol#1159)
        - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value:
amount}(0,path,deadAddress,block.timestamp.add(300)) (PVP.sol#853-858)
    State variables written after the call(s):
    - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount
exceeds allowance)) (PVP.sol#601)
        - _allowances[owner][spender] = amount (PVP.sol#685)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

Reentrancy-3

```
Reentrancy in PVP._transfer(address,address,uint256) (PVP.sol#689-806):
    External calls:
    - swapTokens(contractTokenBalance) (PVP.sol#718)
        -
uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),blo
ck.timestamp) (PVP.sol#835-841)
    - buyBackTokens(_bBSLimit) (PVP.sol#753)
        - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value:
amount}(0,path,deadAddress,block.timestamp.add(300)) (PVP.sol#853-858)
    External calls sending eth:
    - swapTokens(contractTokenBalance) (PVP.sol#718)
        - recipient.transfer(amount) (PVP.sol#1159)
    - buyBackTokens(_bBSLimit) (PVP.sol#753)
        - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value:
amount}(0,path,deadAddress,block.timestamp.add(300)) (PVP.sol#853-858)
    Event emitted after the call(s):
    - SwapETHForTokens(amount,path) (PVP.sol#860)
        - buyBackTokens(_bBSLimit) (PVP.sol#753)
    - Transfer(sender,recipient,tTransferAmount) (PVP.sol#902)
        - _tokenTransfer(from,to,amount,takeFee) (PVP.sol#805)
    - Transfer(sender,recipient,tTransferAmount) (PVP.sol#922)
        - _tokenTransfer(from,to,amount,takeFee) (PVP.sol#805)
    - Transfer(sender,recipient,tTransferAmount) (PVP.sol#912)
        - _tokenTransfer(from,to,amount,takeFee) (PVP.sol#805)
    - Transfer(sender,recipient,tTransferAmount) (PVP.sol#933)
        - _tokenTransfer(from,to,amount,takeFee) (PVP.sol#805)
Reentrancy in PVP.constructor() (PVP.sol#540-562):
    External calls:
    - uniswapV2Pair =
IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),_uniswapV2Router.WETH())
(PVP.sol#550-551)
    Event emitted after the call(s):
    - Transfer(address(0),_msgSender(),_tTotal) (PVP.sol#561)
Reentrancy in PVP.swapETHForTokens(uint256) (PVP.sol#846-861):
    External calls:
    - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value:
amount}(0,path,deadAddress,block.timestamp.add(300)) (PVP.sol#853-858)
    Event emitted after the call(s):
    - SwapETHForTokens(amount,path) (PVP.sol#860)
Reentrancy in PVP.swapTokensForEth(uint256) (PVP.sol#826-844):
    External calls:
    -
uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),blo
ck.timestamp) (PVP.sol#835-841)
    Event emitted after the call(s):
    - SwapTokensForETH(tokenAmount,path) (PVP.sol#843)
Reentrancy in PVP.transferFrom(address,address,uint256) (PVP.sol#599-603):
    External calls:
    - _transfer(sender,recipient,amount) (PVP.sol#600)
```

```
        - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value:
amount}(0,path,deadAddress,block.timestamp.add(300)) (PVP.sol#853-858)
        -
uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),blo
ck.timestamp) (PVP.sol#835-841)
    External calls sending eth:
    - _transfer(sender,recipient,amount) (PVP.sol#600)
        - recipient.transfer(amount) (PVP.sol#1159)
        - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value:
amount}(0,path,deadAddress,block.timestamp.add(300)) (PVP.sol#853-858)
    Event emitted after the call(s):
    - Approval(owner,spender,amount) (PVP.sol#686)
        - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer
amount exceeds allowance)) (PVP.sol#601)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

Assembly usage

```
Ownable.unlock() (PVP.sol#215-220) uses timestamp for comparisons
    Dangerous comparisons:
    - require(bool,string)(block.timestamp > _lockTime,Contract is locked until 7 days) (PVP.sol#217)
PVP._transfer(address,address,uint256) (PVP.sol#689-806) uses timestamp for comparisons
    Dangerous comparisons:
    - overMinimumTokenBalance && _startTimeForSwap + _intervalMinutesForSwap <= block.timestamp
(PVP.sol#715)
    - _sellHistories[i].time >= startTime (PVP.sol#735)
    - balance > _bBSLimit (PVP.sol#752)
PVP.buyBackTokens(uint256) (PVP.sol#820-824) uses timestamp for comparisons
    Dangerous comparisons:
    - amount > 0 (PVP.sol#821)
PVP._removeOldSellHistories() (PVP.sol#1038-1060) uses timestamp for comparisons
    Dangerous comparisons:
    - _sellHistories[j].time >= maxStartTimeForHistories (PVP.sol#1044)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Address.isContract(address) (PVP.sol#110-119) uses assembly
    - INLINE ASM (PVP.sol#117)
Address._functionCallWithValue(address,bytes,uint256,string) (PVP.sol#147-164) uses assembly
    - INLINE ASM (PVP.sol#156-159)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

Dead Code

```
Address._functionCallWithValue(address,bytes,uint256,string) (PVP.sol#147-164) is never used and should
be removed
Address.functionCall(address,bytes) (PVP.sol#130-132) is never used and should be removed
Address.functionCall(address,bytes,string) (PVP.sol#134-136) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (PVP.sol#138-140) is never used and should be
removed
Address.functionCallWithValue(address,bytes,uint256,string) (PVP.sol#142-145) is never used and should
be removed
Address.isContract(address) (PVP.sol#110-119) is never used and should be removed
Address.sendValue(address,uint256) (PVP.sol#121-127) is never used and should be removed
Context._msgData() (PVP.sol#33-36) is never used and should be removed
PVP.addLiquidity(uint256,uint256) (PVP.sol#863-876) is never used and should be removed
SafeMath.mod(uint256,uint256) (PVP.sol#98-100) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (PVP.sol#102-105) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```

Function initializing state

```
PVP._rTotal (PVP.sol#450) is set pre-construction with a non-constant function or state variable:
    - (MAX - (MAX % _tTotal))
PVP._previousTaxFee (PVP.sol#473) is set pre-construction with a non-constant function or state
variable:
    - _taxFee
PVP._previousLiquidityFee (PVP.sol#476) 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
```

Incorrect versions of Solidity

```
Pragma version^0.8.4 (PVP.sol#26) necessitates a version too recent to be trusted. Consider deploying
with 0.6.12/0.7.6
solc-0.8.8 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

Low level calls

```
Low level call in Address.sendValue(address,uint256) (PVP.sol#121-127):
    - (success) = recipient.call{value: amount}() (PVP.sol#125)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (PVP.sol#147-164):
    - (success,returndata) = target.call{value: weiValue}(data) (PVP.sol#150)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

Naming conventions

```
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (PVP.sol#260) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (PVP.sol#261) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (PVP.sol#277) is not in mixedCase
Function IUniswapV2Router01.WETH() (PVP.sol#298) is not in mixedCase
Parameter PVP.calculateTaxFee(uint256)._amount (PVP.sol#987) is not in mixedCase
Parameter PVP.calculateLiquidityFee(uint256)._amount (PVP.sol#993) is not in mixedCase
Function PVP.SetBuyBackMaxTimeForHistories(uint256) (PVP.sol#1062-1064) is not in mixedCase
Function PVP.SetBuyBackDivisor(uint256) (PVP.sol#1066-1068) is not in mixedCase
Function PVP.GetBuyBackTimeInterval() (PVP.sol#1070-1072) is not in mixedCase
Function PVP.SetBuyBackTimeInterval(uint256) (PVP.sol#1074-1076) is not in mixedCase
Function PVP.SetBuyBackRangeRate(uint256) (PVP.sol#1078-1081) is not in mixedCase
Function PVP.GetSwapMinutes() (PVP.sol#1083-1085) is not in mixedCase
Function PVP.SetSwapMinutes(uint256) (PVP.sol#1087-1089) is not in mixedCase
Parameter PVP.setNumTokensSellToAddToBuyBack(uint256)._minimumTokensBeforeSwap (PVP.sol#1121) is not in
mixedCase
Parameter PVP.setMarketingAddress(address)._marketingAddress (PVP.sol#1125) is not in mixedCase
Parameter PVP.setSwapAndLiquifyEnabled(bool)._enabled (PVP.sol#1129) is not in mixedCase
Parameter PVP.setBuyBackEnabled(bool)._enabled (PVP.sol#1134) is not in mixedCase
Parameter PVP.setAutoBuyBackEnabled(bool)._enabled (PVP.sol#1139) is not in mixedCase
Parameter PVP.changeRouterVersion(address)._router (PVP.sol#1162) is not in mixedCase
Parameter PVP.transferForeignToken(address,address)._token (PVP.sol#1181) is not in mixedCase
Parameter PVP.transferForeignToken(address,address)._to (PVP.sol#1181) is not in mixedCase
Function PVP.Sweep() (PVP.sol#1187-1190) is not in mixedCase
Parameter PVP.setAddressFee(address,bool,uint256,uint256)._address (PVP.sol#1192) is not in mixedCase
Parameter PVP.setAddressFee(address,bool,uint256,uint256)._enable (PVP.sol#1192) is not in mixedCase
Parameter PVP.setAddressFee(address,bool,uint256,uint256)._addressTaxFee (PVP.sol#1192) is not in
mixedCase
Parameter PVP.setAddressFee(address,bool,uint256,uint256)._addressLiquidityFee (PVP.sol#1192) is not in
mixedCase
```

```
Parameter PVP.setBuyAddressFee(address,bool,uint256,uint256)._address (PVP.sol#1198) is not in
mixedCase
Parameter PVP.setBuyAddressFee(address,bool,uint256,uint256)._enable (PVP.sol#1198) is not in mixedCase
Parameter PVP.setBuyAddressFee(address,bool,uint256,uint256)._addressTaxFee (PVP.sol#1198) is not in
mixedCase
Parameter PVP.setBuyAddressFee(address,bool,uint256,uint256)._addressLiquidityFee (PVP.sol#1198) is not
in mixedCase
Parameter PVP.setSellAddressFee(address,bool,uint256,uint256)._address (PVP.sol#1204) is not in
mixedCase
Parameter PVP.setSellAddressFee(address,bool,uint256,uint256)._enable (PVP.sol#1204) is not in
mixedCase
Parameter PVP.setSellAddressFee(address,bool,uint256,uint256)._addressTaxFee (PVP.sol#1204) is not in
mixedCase
Parameter PVP.setSellAddressFee(address,bool,uint256,uint256)._addressLiquidityFee (PVP.sol#1204) is
not in mixedCase
Variable PVP._taxFee (PVP.sol#472) is not in mixedCase
Variable PVP._liquidityFee (PVP.sol#475) is not in mixedCase
Variable PVP._buyTaxFee (PVP.sol#478) is not in mixedCase
Variable PVP._buyLiquidityFee (PVP.sol#479) is not in mixedCase
Variable PVP._sellTaxFee (PVP.sol#481) is not in mixedCase
Variable PVP._sellLiquidityFee (PVP.sol#482) is not in mixedCase
Variable PVP._startTimeForSwap (PVP.sol#484) is not in mixedCase
Variable PVP._intervalMinutesForSwap (PVP.sol#485) is not in mixedCase
Variable PVP._buyBackRangeRate (PVP.sol#487) is not in mixedCase
Variable PVP._addressFees (PVP.sol#490) is not in mixedCase
Variable PVP._maxTxAmount (PVP.sol#494) is not in mixedCase
Variable PVP._sellHistories (PVP.sol#499) is not in mixedCase
Variable PVP._isAutoBuyBack (PVP.sol#500) is not in mixedCase
Variable PVP._buyBackDivisor (PVP.sol#501) is not in mixedCase
Variable PVP._buyBackTimeInterval (PVP.sol#502) is not in mixedCase
Variable PVP._buyBackMaxTimeForHistories (PVP.sol#503) is not in mixedCase
Variable PVP._isEnabledBuyBackAndBurn (PVP.sol#512) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-
naming-conventions
```

Redundant statement

```
Redundant expression "this (PVP.sol#34)" inContext (PVP.sol#28-37)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
```

Reentrancy-4

```
Reentrancy in PVP._transfer(address,address,uint256) (PVP.sol#689-806):
    External calls:
    - swapTokens(contractTokenBalance) (PVP.sol#718)
        - recipient.transfer(amount) (PVP.sol#1159)
    State variables written after the call(s):
    - _removeOldSellHistories() (PVP.sol#745)
        - _sellHistories[i].time = _sellHistories[j].time (PVP.sol#1046)
        - _sellHistories[i].bnbAmount = _sellHistories[j].bnbAmount (PVP.sol#1047)
        - _sellHistories.pop() (PVP.sol#1057)
Reentrancy in PVP._transfer(address,address,uint256) (PVP.sol#689-806):
    External calls:
    - swapTokens(contractTokenBalance) (PVP.sol#718)
        - recipient.transfer(amount) (PVP.sol#1159)
    External calls sending eth:
    - swapTokens(contractTokenBalance) (PVP.sol#718)
        - recipient.transfer(amount) (PVP.sol#1159)
    - buyBackTokens(_bBSLimit) (PVP.sol#753)
        - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value:
amount}(0,path,deadAddress,block.timestamp.add(300)) (PVP.sol#853-858)
```

```
State variables written after the call(s):
- removeAllFee() (PVP.sol#769)
    - _liquidityFee = 0 (PVP.sol#1006)
- _liquidityFee = _buyLiquidityFee (PVP.sol#771)
- removeAllFee() (PVP.sol#775)
    - _liquidityFee = 0 (PVP.sol#1006)
- _liquidityFee = _sellLiquidityFee (PVP.sol#777)
- removeAllFee() (PVP.sol#782)
    - _liquidityFee = 0 (PVP.sol#1006)
- _liquidityFee = _addressFees[from]._liquidityFee (PVP.sol#784)
- _liquidityFee = _addressFees[from]._sellLiquidityFee (PVP.sol#789)
- removeAllFee() (PVP.sol#796)
    - _liquidityFee = 0 (PVP.sol#1006)
- _liquidityFee = _addressFees[to]._buyLiquidityFee (PVP.sol#799)
- _tokenTransfer(from,to,amount,takeFee) (PVP.sol#805)
    - _liquidityFee = _previousLiquidityFee (PVP.sol#1011)
    - _liquidityFee = 0 (PVP.sol#1006)
- removeAllFee() (PVP.sol#769)
    - _previousLiquidityFee = _liquidityFee (PVP.sol#1003)
- removeAllFee() (PVP.sol#775)
    - _previousLiquidityFee = _liquidityFee (PVP.sol#1003)
- removeAllFee() (PVP.sol#782)
    - _previousLiquidityFee = _liquidityFee (PVP.sol#1003)
- removeAllFee() (PVP.sol#796)
    - _previousLiquidityFee = _liquidityFee (PVP.sol#1003)
- _tokenTransfer(from,to,amount,takeFee) (PVP.sol#805)
    - _previousLiquidityFee = _liquidityFee (PVP.sol#1003)
- removeAllFee() (PVP.sol#769)
    - _previousTaxFee = _taxFee (PVP.sol#1002)
- removeAllFee() (PVP.sol#775)
    - _previousTaxFee = _taxFee (PVP.sol#1002)
- removeAllFee() (PVP.sol#782)
    - _previousTaxFee = _taxFee (PVP.sol#1002)
- removeAllFee() (PVP.sol#796)
    - _previousTaxFee = _taxFee (PVP.sol#1002)
- _tokenTransfer(from,to,amount,takeFee) (PVP.sol#805)
    - _previousTaxFee = _taxFee (PVP.sol#1002)
- _tokenTransfer(from,to,amount,takeFee) (PVP.sol#805)
    - _rOwned[address(this)] = _rOwned[address(this)].add(rLiquidity) (PVP.sol#982)
    - _rOwned[sender] = _rOwned[sender].sub(rAmount) (PVP.sol#898)
    - _rOwned[sender] = _rOwned[sender].sub(rAmount) (PVP.sol#907)
    - _rOwned[sender] = _rOwned[sender].sub(rAmount) (PVP.sol#918)
    - _rOwned[sender] = _rOwned[sender].sub(rAmount) (PVP.sol#928)
    - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (PVP.sol#899)
    - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (PVP.sol#909)
    - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (PVP.sol#919)
    - _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) (PVP.sol#930)
- _tokenTransfer(from,to,amount,takeFee) (PVP.sol#805)
    - _rTotal = _rTotal.sub(rFee) (PVP.sol#937)
- _tokenTransfer(from,to,amount,takeFee) (PVP.sol#805)
    - _tFeeTotal = _tFeeTotal.add(tFee) (PVP.sol#938)
- _tokenTransfer(from,to,amount,takeFee) (PVP.sol#805)
    - _tOwned[address(this)] = _tOwned[address(this)].add(tLiquidity) (PVP.sol#984)
    - _tOwned[sender] = _tOwned[sender].sub(tAmount) (PVP.sol#917)
    - _tOwned[sender] = _tOwned[sender].sub(tAmount) (PVP.sol#927)
    - _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (PVP.sol#908)
    - _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount) (PVP.sol#929)
- removeAllFee() (PVP.sol#769)
    - _taxFee = 0 (PVP.sol#1005)
- _taxFee = _buyTaxFee (PVP.sol#770)
- removeAllFee() (PVP.sol#775)
    - _taxFee = 0 (PVP.sol#1005)
- _taxFee = _sellTaxFee (PVP.sol#776)
- removeAllFee() (PVP.sol#782)
    - _taxFee = 0 (PVP.sol#1005)
```

```
    - _taxFee = _addressFees[from]._taxFee (PVP.sol#783)
    - _taxFee = _addressFees[from]._sellTaxFee (PVP.sol#788)
    - removeAllFee() (PVP.sol#796)
        - _taxFee = 0 (PVP.sol#1005)
    - _taxFee = _addressFees[to]._buyTaxFee (PVP.sol#798)
    - _tokenTransfer(from,to,amount,takeFee) (PVP.sol#805)
        - _taxFee = _previousTaxFee (PVP.sol#1010)
        - _taxFee = 0 (PVP.sol#1005)
    - buyBackTokens(_bBSLimit) (PVP.sol#753)
        - inSwapAndLiquify = true (PVP.sol#535)
        - inSwapAndLiquify = false (PVP.sol#537)
    Event emitted after the call(s):
    - SwapETHForTokens(amount,path) (PVP.sol#860)
        - buyBackTokens(_bBSLimit) (PVP.sol#753)
    - Transfer(sender,recipient,tTransferAmount) (PVP.sol#902)
        - _tokenTransfer(from,to,amount,takeFee) (PVP.sol#805)
    - Transfer(sender,recipient,tTransferAmount) (PVP.sol#912)
        - _tokenTransfer(from,to,amount,takeFee) (PVP.sol#805)
    - Transfer(sender,recipient,tTransferAmount) (PVP.sol#922)
        - _tokenTransfer(from,to,amount,takeFee) (PVP.sol#805)
    - Transfer(sender,recipient,tTransferAmount) (PVP.sol#933)
        - _tokenTransfer(from,to,amount,takeFee) (PVP.sol#805)
Reentrancy in PVP.transferFrom(address,address,uint256) (PVP.sol#599-603):
    External calls:
    - _transfer(sender,recipient,amount) (PVP.sol#600)
        - recipient.transfer(amount) (PVP.sol#1159)
    External calls sending eth:
    - _transfer(sender,recipient,amount) (PVP.sol#600)
        - recipient.transfer(amount) (PVP.sol#1159)
        - uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value:
amount}(0,path,deadAddress,block.timestamp.add(300)) (PVP.sol#853-858)
    State variables written after the call(s):
    - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount
exceeds allowance)) (PVP.sol#601)
        - _allowances[owner][spender] = amount (PVP.sol#685)
    Event emitted after the call(s):
    - Approval(owner,spender,amount) (PVP.sol#686)
        - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer
amount exceeds allowance)) (PVP.sol#601)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4
```

Variable names are too similar

```
Variable
IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amount
ADesired (PVP.sol#303) is too similar to
IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amount
BDesired (PVP.sol#304)
Variable PVP._transferToExcluded(address,address,uint256).rTransferAmount (PVP.sol#906) is too similar
to PVP._transferBothExcluded(address,address,uint256).tTransferAmount (PVP.sol#926)
Variable PVP._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (PVP.sol#958) is too similar
to PVP._transferBothExcluded(address,address,uint256).tTransferAmount (PVP.sol#926)
Variable PVP.reflectionFromToken(uint256,bool).rTransferAmount (PVP.sol#647) is too similar to
PVP._transferBothExcluded(address,address,uint256).tTransferAmount (PVP.sol#926)
Variable PVP._transferToExcluded(address,address,uint256).rTransferAmount (PVP.sol#906) is too similar
to PVP._transferFromExcluded(address,address,uint256).tTransferAmount (PVP.sol#916)
Variable PVP._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (PVP.sol#958) is too similar
to PVP._transferFromExcluded(address,address,uint256).tTransferAmount (PVP.sol#916)
Variable PVP.reflectionFromToken(uint256,bool).rTransferAmount (PVP.sol#647) is too similar to
PVP._transferFromExcluded(address,address,uint256).tTransferAmount (PVP.sol#916)
Variable PVP._transferFromExcluded(address,address,uint256).rTransferAmount (PVP.sol#916) is too
similar to PVP._transferStandard(address,address,uint256).tTransferAmount (PVP.sol#897)
```

```
Variable PVP._transferToExcluded(address,address,uint256).rTransferAmount (PVP.sol#906) is too similar
to PVP._getValues(uint256).tTransferAmount (PVP.sol#942)
Variable PVP._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (PVP.sol#958) is too similar
to PVP._getValues(uint256).tTransferAmount (PVP.sol#942)
Variable PVP.reflectionFromToken(uint256,bool).rTransferAmount (PVP.sol#647) is too similar to
PVP._getValues(uint256).tTransferAmount (PVP.sol#942)
Variable PVP._transferToExcluded(address,address,uint256).rTransferAmount (PVP.sol#906) is too similar
to PVP._transferToExcluded(address,address,uint256).tTransferAmount (PVP.sol#906)
Variable PVP._transferBothExcluded(address,address,uint256).rTransferAmount (PVP.sol#926) is too
similar to PVP._transferStandard(address,address,uint256).tTransferAmount (PVP.sol#897)
Variable PVP.reflectionFromToken(uint256,bool).rTransferAmount (PVP.sol#647) is too similar to
PVP._transferToExcluded(address,address,uint256).tTransferAmount (PVP.sol#906)
Variable PVP._transferFromExcluded(address,address,uint256).rTransferAmount (PVP.sol#916) is too
similar to PVP._transferBothExcluded(address,address,uint256).tTransferAmount (PVP.sol#926)
Variable PVP._transferStandard(address,address,uint256).rTransferAmount (PVP.sol#897) is too similar to
PVP._transferStandard(address,address,uint256).tTransferAmount (PVP.sol#897)
Variable PVP._transferFromExcluded(address,address,uint256).rTransferAmount (PVP.sol#916) is too
similar to PVP._transferFromExcluded(address,address,uint256).tTransferAmount (PVP.sol#916)
Variable PVP._transferBothExcluded(address,address,uint256).rTransferAmount (PVP.sol#926) is too
similar to PVP._transferBothExcluded(address,address,uint256).tTransferAmount (PVP.sol#926)
Variable PVP._transferToExcluded(address,address,uint256).rTransferAmount (PVP.sol#906) is too similar
to PVP._transferStandard(address,address,uint256).tTransferAmount (PVP.sol#897)
Variable PVP._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (PVP.sol#958) is too similar
to PVP._transferStandard(address,address,uint256).tTransferAmount (PVP.sol#897)
Variable PVP._transferFromExcluded(address,address,uint256).rTransferAmount (PVP.sol#916) is too
similar to PVP._getValues(uint256).tTransferAmount (PVP.sol#942)
Variable PVP.reflectionFromToken(uint256,bool).rTransferAmount (PVP.sol#647) is too similar to
PVP._transferStandard(address,address,uint256).tTransferAmount (PVP.sol#897)
Variable PVP._transferBothExcluded(address,address,uint256).rTransferAmount (PVP.sol#926) is too
similar to PVP._transferFromExcluded(address,address,uint256).tTransferAmount (PVP.sol#916)
Variable PVP._transferToExcluded(address,address,uint256).rTransferAmount (PVP.sol#906) is too similar
to PVP._getTValues(uint256).tTransferAmount (PVP.sol#950)
Variable PVP._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (PVP.sol#958) is too similar
to PVP._getTValues(uint256).tTransferAmount (PVP.sol#950)
Variable PVP.reflectionFromToken(uint256,bool).rTransferAmount (PVP.sol#647) is too similar to
PVP._getTValues(uint256).tTransferAmount (PVP.sol#950)
Variable PVP._transferBothExcluded(address,address,uint256).rTransferAmount (PVP.sol#926) is too
similar to PVP._getValues(uint256).tTransferAmount (PVP.sol#942)
Variable PVP._transferFromExcluded(address,address,uint256).rTransferAmount (PVP.sol#916) is too
similar to PVP._getTValues(uint256).tTransferAmount (PVP.sol#950)
Variable PVP._getRValues(uint256,uint256,uint256,uint256).rTransferAmount (PVP.sol#958) is too similar
to PVP._transferToExcluded(address,address,uint256).tTransferAmount (PVP.sol#906)
Variable PVP._getValues(uint256).rTransferAmount (PVP.sol#943) is too similar to
PVP._transferToExcluded(address,address,uint256).tTransferAmount (PVP.sol#906)
Variable PVP._transferBothExcluded(address,address,uint256).rTransferAmount (PVP.sol#926) is too
similar to PVP._transferToExcluded(address,address,uint256).tTransferAmount (PVP.sol#906)
Variable PVP._transferFromExcluded(address,address,uint256).rTransferAmount (PVP.sol#916) is too
similar to PVP._transferToExcluded(address,address,uint256).tTransferAmount (PVP.sol#906)
Variable PVP._transferStandard(address,address,uint256).rTransferAmount (PVP.sol#897) is too similar to
PVP._transferFromExcluded(address,address,uint256).tTransferAmount (PVP.sol#916)
Variable PVP._getValues(uint256).rTransferAmount (PVP.sol#943) is too similar to
PVP._transferFromExcluded(address,address,uint256).tTransferAmount (PVP.sol#916)
Variable PVP._transferStandard(address,address,uint256).rTransferAmount (PVP.sol#897) is too similar to
PVP._transferBothExcluded(address,address,uint256).tTransferAmount (PVP.sol#926)
Variable PVP._transferStandard(address,address,uint256).rTransferAmount (PVP.sol#897) is too similar to
PVP._getTValues(uint256).tTransferAmount (PVP.sol#950)
Variable PVP._getValues(uint256).rTransferAmount (PVP.sol#943) is too similar to
PVP._getTValues(uint256).tTransferAmount (PVP.sol#950)
Variable PVP._getValues(uint256).rTransferAmount (PVP.sol#943) is too similar to
PVP._transferBothExcluded(address,address,uint256).tTransferAmount (PVP.sol#926)
Variable PVP._getValues(uint256).rTransferAmount (PVP.sol#943) is too similar to
PVP._transferStandard(address,address,uint256).tTransferAmount (PVP.sol#897)
Variable PVP._transferStandard(address,address,uint256).rTransferAmount (PVP.sol#897) is too similar to
PVP._getValues(uint256).tTransferAmount (PVP.sol#942)
```

```
Variable PVP._transferBothExcluded(address,address,uint256).rTransferAmount (PVP.sol#926) is too
similar to PVP._getTValues(uint256).tTransferAmount (PVP.sol#950)
Variable PVP._getValues(uint256).rTransferAmount (PVP.sol#943) is too similar to
PVP._getValues(uint256).tTransferAmount (PVP.sol#942)
Variable PVP._transferStandard(address,address,uint256).rTransferAmount (PVP.sol#897) is too similar to
PVP._transferToExcluded(address,address,uint256).tTransferAmount (PVP.sol#906)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-
similarDocumentation#local-variable-shadowing
```

Too many digits

```
PVP.prepareForPreSale() (PVP.sol#1144-1149) uses literals with too many digits:
    - _maxTxAmount = 1000000000 * 10 ** 6 * 10 ** 9 (PVP.sol#1148)
PVP.afterPreSale() (PVP.sol#1151-1156) uses literals with too many digits:
    - _maxTxAmount = 3000000 * 10 ** 6 * 10 ** 9 (PVP.sol#1155)
PVP.slitherConstructorVariables() (PVP.sol#433-1211) uses literals with too many digits:
    - deadAddress = 0x000000000000000000000000000000000000dEaD (PVP.sol#438)
PVP.slitherConstructorVariables() (PVP.sol#433-1211) uses literals with too many digits:
    - _tTotal = 40000000 * 10 ** 9 (PVP.sol#449)
PVP.slitherConstructorVariables() (PVP.sol#433-1211) uses literals with too many digits:
    - _maxTxAmount = 3000000 * 10 ** 9 (PVP.sol#494)
PVP.slitherConstructorVariables() (PVP.sol#433-1211) uses literals with too many digits:
    - minimumTokensBeforeSwap = 200000 * 10 ** 9 (PVP.sol#495)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
```

State variables that could be declared constant

```
PVP._decimals (PVP.sol#455) should be constant
PVP._isEnabledBuyBackAndBurn (PVP.sol#512) should be constant
PVP._name (PVP.sol#453) should be constant
PVP._symbol (PVP.sol#454) should be constant
PVP._tTotal (PVP.sol#449) should be constant
PVP.deadAddress (PVP.sol#438) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-
declared-constantktop/Code/PVP.sol#681) shadows:
    - Ownable.owner() (PVP.sol#180-182) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
```

Public function that could be declared external

```
renounceOwnership() should be declared external:
    - Ownable.renounceOwnership() (PVP.sol#189-192)
transferOwnership(address) should be declared external:
    - Ownable.transferOwnership(address) (PVP.sol#194-198)
getUnlockTime() should be declared external:
    - Ownable.getUnlockTime() (PVP.sol#200-202)
getTime() should be declared external:
    - Ownable.getTime() (PVP.sol#204-206)
lock(uint256) should be declared external:
    - Ownable.lock(uint256) (PVP.sol#208-213)
unlock() should be declared external:
    - Ownable.unlock() (PVP.sol#215-220)
name() should be declared external:
    - PVP.name() (PVP.sol#564-566)
symbol() should be declared external:
    - PVP.symbol() (PVP.sol#568-570)
decimals() should be declared external:
    - PVP.decimals() (PVP.sol#572-574)
```

```
totalSupply() should be declared external:
    - PVP.totalSupply() (PVP.sol#576-578)
transfer(address,uint256) should be declared external:
    - PVP.transfer(address,uint256) (PVP.sol#585-588)
allowance(address,address) should be declared external:
    - PVP.allowance(address,address) (PVP.sol#590-592)
approve(address,uint256) should be declared external:
    - PVP.approve(address,uint256) (PVP.sol#594-597)
transferFrom(address,address,uint256) should be declared external:
    - PVP.transferFrom(address,address,uint256) (PVP.sol#599-603)
increaseAllowance(address,uint256) should be declared external:
    - PVP.increaseAllowance(address,uint256) (PVP.sol#605-608)
decreaseAllowance(address,uint256) should be declared external:
    - PVP.decreaseAllowance(address,uint256) (PVP.sol#610-613)
isExcludedFromReward(address) should be declared external:
    - PVP.isExcludedFromReward(address) (PVP.sol#615-617)
totalFees() should be declared external:
    - PVP.totalFees() (PVP.sol#619-621)
minimumTokensBeforeSwapAmount() should be declared external:
    - PVP.minimumTokensBeforeSwapAmount() (PVP.sol#623-625)
buyBackSellLimitAmount() should be declared external:
    - PVP.buyBackSellLimitAmount() (PVP.sol#627-629)
deliver(uint256) should be declared external:
    - PVP.deliver(uint256) (PVP.sol#631-638)
reflectionFromToken(uint256,bool) should be declared external:
    - PVP.reflectionFromToken(uint256,bool) (PVP.sol#641-650)
excludeFromReward(address) should be declared external:
    - PVP.excludeFromReward(address) (PVP.sol#658-666)
isExcludedFromFee(address) should be declared external:
    - PVP.isExcludedFromFee(address) (PVP.sol#1014-1016)
excludeFromFee(address) should be declared external:
    - PVP.excludeFromFee(address) (PVP.sol#1018-1020)
includeInFee(address) should be declared external:
    - PVP.includeInFee(address) (PVP.sol#1022-1024)
GetBuyBackTimeInterval() should be declared external:
    - PVP.GetBuyBackTimeInterval() (PVP.sol#1070-1072)
GetSwapMinutes() should be declared external:
    - PVP.GetSwapMinutes() (PVP.sol#1083-1085)
setBuyBackEnabled(bool) should be declared external:
    - PVP.setBuyBackEnabled(bool) (PVP.sol#1134-1137)
setAutoBuyBackEnabled(bool) should be declared external:
    - PVP.setAutoBuyBackEnabled(bool) (PVP.sol#1139-1142)
changeRouterVersion(address) should be declared external:
    - PVP.changeRouterVersion(address) (PVP.sol#1162-1175)
transferForeignToken(address,address) should be declared external:
    - PVP.transferForeignToken(address,address) (PVP.sol#1181-1185)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-
declared-external
```

Local variable shadowing

```
PVP.allowance(address,address).owner (PVP.sol#590) shadows:
    - Ownable.owner() (PVP.sol#180-182) (function)
PVP._approve(address,address,uint256).owner (PVP.sol#681) shadows:
    - Ownable.owner() (PVP.sol#180-182) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
```

Code Documentation

The code has only minimal comments. This could be improved to help others understand the contract.

Adherence to Specifications

The smart contract adheres to the standard BEP-20 token conformity.

Adherence to Best Practices

The smart contract adheres to the majority of best practices associated with a standard BEP-20 token. The few things that don't follow the best practices are noted in the automated review.

Darkscope Cyber Security Analysis

Cyber Threat Scan

This Cyber Scan report is an examination of the risk profile of cyberspace on behalf of the organization. This scan presents a snapshot profile of the external cyber risk, as it was conducted over a short period of time, usually days. The information gathered from the internet, social media and the darkweb about the organization is not exhaustive or complete due to the continuous growth of cyberspace, its size and the constantly changing nature of focus of the darkweb, particularly. A more comprehensive understanding requires longer monitoring with a broader scope, such as that provided by Darkscope's Cyber Threat Sentinel or Cyber Watchtower services.

Darkscope delivers this report with all due diligence and best efforts but cannot guarantee its accuracy.

This section is in four parts.

1. The Cyber Interference Risk Score provides an overall rating of the cyber risk for the organization.
2. The Cyber Threat Sentinel results rate the risk to the organization from key areas of cyber-attack – phishing, DDoS and Ransom DDoS, website hijacking and ransomware. These are prevalent forms of attack against an organization, its partners, and customers. Understanding these risks can help an organization prepare itself against these forms of attack.
3. Warnings and alerts. Using a traffic light system: Green – information, Orange – warnings, Red – alerts; these items require action to mitigate weaknesses or risks to the organization.
4. Available Information. This information is in cyberspace. It may include emails of former employees or contracts that are no longer current or show infrastructure links that can be exploited (DDoS) that may have weak security or is redundant.

Darkscope Cyber Interference™ Risk Score

This score is a summary of your overall cyber risk. It is compiled from all the risk data Darkscope collects across the internet, social media and the darkweb about you and profiles your organization within your industry sector and geographic region. Using baseline data collected across millions of data points daily and algorithms that compare your risk factors, your Cyber Interference Risk Score is the most reliable overall assessment of the specific cyber risk for your organization.

Darkscope provides CIRS with more detail as part of its other enterprise cyber intelligence services. When included in Cyber Threat Sentinel and Cyber Watchtower services, it contains more detail such as Partner Risk Score, Darkweb Risk Level, Impersonate and Social Media Sentiment Rating.

PVP has a low CIRS score and is within it is expected industry and location range. This means PVP has a smaller footprint then most other business in cyberspace and a reduced risk of being attacked, compared with other businesses in its region and industry.

A Low Cyber Interference Risk Score indicates less external interest in the organization, region, or industry and that the organization is not actively being examined. However, threat actors always have an increased interest in entities like PVP. It is recommended to adjust the Cyber Security Program to mitigate the findings from this report.

Cyber Threat Sentinel Results

The Cryptocurrency Audit results identify your risk across four key cyber risk areas: Phishing, DDoS/RDDoS, Website Hijacking, and Ransomware. The rating scale is Low – Medium – High – Extreme. Each threat type explains how it determines your result and how you should interpret or react when the risk is high.

PHISHING ATTACK

To calculate the risk of a phishing attack, we use the information an attacker has or could find in cyberspace about PVP's people, roles, and internal processes. We incorporate past breaches, current cyber-attacks, and campaigns to determine how likely is it that an attacker would choose PVP as a target.
We have identified a Low risk for PVP based on the analysis we did

DDoS & RDDoS ATTACK

Our system analyses the customer external-facing infrastructure using a black-box approach. This means we simulate what an attacker would be able to find in cyberspace about PVP. This includes domains, sub-domains, applications, and existing protections such as Web application firewalls or load balancers. We also include the location of services and determine the local readiness for DDoS attacks. Smaller countries like New Zealand, for example, have often limited preventive measures available due to its location and internet capabilities when compared with Germany or the US.
We have identified a Low risk for PVP based on the analysis we did.

WEBSITE HIJACK

Our system analysis the customer external-facing infrastructure from a black-box approach. This means we simulate what an attacker would be able to find in cyberspace about Demo. This includes domains, sub-domains, applications, and existing protections such as Web application firewalls or load balancers. Out of this information, we determine how vulnerable a customer might be.
We have identified that PVP has a Low risk of being attacked.

RAMSOMWARE

To calculate the risk of Ransomware attacks, we correlate all available information and create a risk profile containing staff, product/service, and business information. Ransomware is most likely to be successful if the attacker knows about the internal processes and communications of the target. We compare this profile with thousands of other businesses in the same industry and region to create a risk value.
PVP has a LOW risk of being targeted with ransomware.

Cyber Risks

These Cyber Risks are rated using a Traffic Light system.

Red is an alert. This indicates an imminent risk to the organization that requires action to fix, prevent or mitigate.

Orange is a warning. These identified risks should be included in a risk register and work program to update, change, or replace.

Green is relevant information. These items show out-of-date practices, expired or end-of-life tools or software. Items identified should be updated or replaced, as they can become vulnerabilities if not fixed.

RISK LEVEL	IDENTIFIED RISK	NOTES AND RECOMMENDATIONS
	INFO: All of the tested domains support TLS 1.3	All of the tested domains support TLS version 1.3
	Info: There was no WAF detected in front of the website tokenpvp.com	Our test could not identify any WAF in front of the main webpage. This itself is not a security finding. *We recommend that you review the current configuration and determine the need for a WAF.*
	Warning: The domain tokenpvp.com has no DMARC record	Our tests show that the domain tokenpvp.com has no DMARC record enabled. *We recommend reviewing this finding and adding a DMARC record to the DNS configuration.*

Available Information - Email Addresses

Publicly available email addresses are a normal part of every organization's external-facing operation. They are used in marketing, publicity and engaging customers and the public. The cyber risk they represent is that they provide a list of targets for phishing or scam emails and can also be spoofed to scam or phish your customers, partners, or staff. Knowing which emails are public lets you make these people aware of their heightened risk of becoming a target and to be more diligent and capable of identifying unusual or suspicious behaviour and activity.

Email	Name	Position	Department

Available Information – Infrastructure

Information associated with the hostname, such as IP addresses, DNS and Netblock owner, can provide an attacker with a point of entry (brute force or weak login/password) or a less protected point of attack (DDoS). Ensuring all your IP addresses are well protected will reduce the effect of any attack or breach attempt.

Hosts

Hostname	IP Address	Type	Reverse DNS	Netblock Owner
Tokenpvp.com	54.205.240.192	A	ec2-54-205-240-192.compute-1.amazonaws.com	AMAZON-AES United States