QuillAudits

Audit Report
July, 2021

paybswap

Contents

Scope of Audit

The scope of this audit was to analyze and document the Paybswap Token smart contract codebase for quality, security, and correctness.

Checked Vulnerabilities

We have scanned the smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered:

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level

Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis
In this step we have analyzed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.
SmartCheck.

Static Analysis
Static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step a series of automated tools are used to test security of smart contracts.

Code Review / Manual Analysis
Manual Analysis or review of code was done to identify new vulnerability or verify the vulnerabilities found during the static analysis. Contracts were completely manually analyzed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of automated analysis were manually verified.

Gas Consumption
In this step we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit
Remix IDE, Truffle, Truffle Team, Ganache, Solhint, Mythril, Slither, SmartCheck.

Issue Categories

Every issue in this report has been assigned with a severity level. There are four levels of severity and each of them has been explained below.

High severity issues

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality and we recommend these issues to be fixed before moving to a live environment.

Medium level severity issues

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems and they should still be fixed.

Low level severity issues

Low level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are severity four issues which indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Number of issues per severity

Type	High	Medium	Low	Informational
Open	0	0	0	0
Acknowledged	0	1	2	3
Closed	0	0	2	8

Introduction

During the period of **July 1, 2021 to July 6, 2021** - QuillAudits Team performed a security audit for Paybswap smart contracts.

The code for the audit was provided by PaybSwap and their hashes are mentioned below:

File 1: **MasterVault.sol**
Branch: audit
MD5 hash: 693310C2025D1D06D44B05E16CEB95F6
Updated MD5 hash: B247E4C19AC484DAF617A6601173DFCC

File 2: **PaybMockToken.sol**
Branch: audit
MD5 hash: 51E6C4DF37AE6D39F442D362B7AB6283
Updated MD5 hash: 72BB6447619FA5A0AB80F90F9D3C0E5E

File 3: **PaybVirtualToken.sol**
Branch: audit
MD5 hash: 65FFFFECCE4EC3D6EF1C09F128EF3412
Updated MD5 hash: C73C08B040BE5EF949961B72F75F6F0F

Issues Found – Code Review / Manual Testing

A. Contract - MasterVault

High severity issues

No issues were found.

Medium severity issues

1. **Centralization Risks**

   Description
   The role owner has the authority to
   - update settings such as: master supplier, multiplier, withdrawal period and withdraw fee.
   - add new LP token to the pool, as well as update the given pool's PAYB allocation point
   - transfer ownership to any other wallet.

   Remediation
   We advise the client to handle the governance account carefully to avoid any potential hack. We also advise the client to consider the following solutions:

   - With reasonable latency for community awareness on privileged operations;
   - Multisig with community-voted 3rd-party independent co-signers;
   - DAO or Governance module increasing transparency and community involvement;

   Status: Acknowledged.

Low level severity issues

2. Infinite loop possibility

Line	Code

181	function updateStakingPool() internal {
	uint256 length = poolInfo.length;
	uint256 points = 0;
	for (uint256 pid = 1; pid < length; ++pid) {
	points = points.add(poolInfo[pid].allocPoint);
	}

214	function massUpdatePools() public {
	uint256 length = poolInfo.length;
	for (uint256 pid = 0; pid < length; ++pid) {
	updatePool(pid);
	}
	}

Description

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

Remediation

The number of pools should be limited.

Status: Acknowledged.

3. Input validation missing

Line	Code

146	// Add a new lp to the pool. Can only be called by owner.
	// DO NOT add the same LP token more than once. Rewards will be messed up if you do.
	
	function add(uint256 _allocPoint, IBEP20 _lpToken, bool _withUpdate) public onlyOwner {
	if (_withUpdate) {
	massUpdatePools();
	}

Description
As mentioned in the comment, the LP token must not be added twice by human error. It will create a discrepancy in the reward.

Remediation
It is recommended to add an input param check condition to prevent this scenario from happening.

Status: Fixed

4. Range validation missing

Line	Code

117	function updateMultiplier(uint256 multiplierNumber) public onlyOwner {
	MULTIPLIER = multiplierNumber;
	}

169	function set(uint256 _pid, uint256 _allocPoint, bool _withUpdate) public onlyOwner {
	if (_withUpdate) {
	massUpdatePools();
	}
	:
	:
	}

Description
The role can set the following state variables arbitrary large or small causing potential risks in fees and anti whale :

- updateMultiplier
- set

Remediation
We recommend setting ranges and check the following input variables:

- multiplierNumber in updateMultiplier function
- _pid and _allocPoint in set function

Status: Fixed

Informational

5. Solidity version

Line	Code
1	pragma solidity 0.6.12;

Description
Use the latest solidity version while contract deployment to prevent any compiler version level bugs.

Remediation
Use 0.8.6, which is the latest version at the time of this audit.

Status: Acknowledged.

6. Use SPDX License Identifier

Description
SPDX license identifier not provided in the source file. Before publishing, consider adding a comment containing "SPDX-License-Identifier: <SPDX-License>".

Remediation
Use "// SPDX-License-Identifier: MIT License" for non-open-source code.
Please see https://spdx.org for more information.

Status: Fixed

7. Variable naming

Line	Code

63	// Maximal withdrawal time
	uint256 public constant MAX_WITHDRAW_PERIOD = 72 hours;
	
	// Maximal withdrawal fee
	uint256 public constant MAX_WITHDRAW_FEE = 100; // 1%

Description
MAX_WITHDRAW_PERIOD and MAX_WITHDRAW_FEE were used in the constructor but not used in withdraw function. But they are used in the leavestaking function.

Remediation
So if this variable is for unstaking, then change its name so it can MAX_UNSTAKE_PERIOD and MAX_UNSTAKE_FEE.

Status: Fixed

8. Same name for events

Line	Code

280	function enterStaking(uint256 _amount) public supplierExcluded{
	PoolInfo storage pool = poolInfo[0];
	:
	:
	emit Deposit(msg.sender, 0, _amount);
	}

302	function leaveStaking(uint256 _amount) public {
	PoolInfo storage pool = poolInfo[0];
	UserInfo storage user = userInfo[0][msg.sender];
	:
	:
	emit Withdraw(msg.sender, 0, _amount);
	}

Description
function enterStaking emitting event Deposit This same event is used in deposit function, And function LeaveStaking emitting event name Withdraw, This same event name is used in withdraw function.

Remediation
So to avoid deposit and withdraw confessions make different event names like stake For enterStaking and unstake for leaveStaking.

Status: Fixed

9. Missing Events for Significant Transactions

Line	Code

113	function updateMasterSupplier(address newSupplier) public onlyOwner {
	masterSupplier = newSupplier;
	}

117	function updateMultiplier(uint256 multiplierNumber) public onlyOwner {
	MULTIPLIER = multiplierNumber;
	}

126	function setWithdrawPeriod(uint256 _withdrawPeriod) external onlyOwner
	:
	withdrawalCooldown = _withdrawPeriod;
	}

136	function setWithdrawFee(uint256 _withdrawFee) external onlyOwner {
	:
	withdrawalFee = _withdrawFee;
	}

146	function add(uint256 _allocPoint, IBEP20 _lpToken, bool _withUpdate)
	public onlyOwner {
	:
	:
	}

169	function set(uint256 _pid, uint256 _allocPoint, bool _withUpdate) public
	onlyOwner {
	:
	:
	}

Description

The missing event makes it difficult to track off-chain changes. An event should be emitted for significant transactions calling the following functions:

- updateMasterSupplier
- updateMultiplier
- setWithdrawPeriod
- setWithdrawFee
- add
- set

Remediation
We recommend emitting an event to log in following functions:

- updateMasterSupplier
- updateMultiplier
- setWithdrawPeriod
- setWithdrawFee
- add
- set

Status: Fixed

10. Order of Functions

Description
The following functions and function types should be re-ordered:
- The view functions
- The internal functions

Ordering helps readers identify which functions they can call and to find the constructor and fallback definitions easier.

Remediation
Functions should be grouped according to their visibility and ordered:

- constructor
- fallback function (if exists)
- external
- public
- internal
- private

Within a grouping, place the view and pure functions last.

Status: Fixed

11. Public function that could be declared external

Description
The following public functions that are never called by the contract should be declared external to save gas:

- updateMasterSupplier()
- updateMultiplier()
- add()
- set()
- pendingPayb()
- deposit()
- withdraw()
- enterStaking()
- leaveStaking()
- timeToLeaveStaking()
- currentSupplierAllowance()
- getPoolInvervalReturn()
- emergencyWithdraw()

Remediation
Use the external attribute for functions never called from the contract.

Status: Fixed

B. Contract - PaybMockToken

High severity issues

No issues were found.

Medium severity issues

No issues were found.

Low level severity issues

No issues were found.

Informational

1. Use SPDX License Identifier

Description
SPDX license identifier not provided in the source file. Before publishing, consider adding a comment containing "SPDX-License-Identifier: <SPDX-License>".

Remediation
Use "// SPDX-License-Identifier: MIT License" for non-open-source code.
Please see https://spdx.org for more information.

Status: Fixed

2. Solidity version

Line	Code
1	pragma solidity 0.6.12;

Description
Please use the latest solidity version while contract deployment to prevent any compiler version level bugs.

Remediation
Please consider using 0.8.6, which is the latest version at the time of this audit.

Status: Acknowledged.

C. Contract - PaybVirtualToken

High severity issues

No issues were found.

Medium severity issues

No issues were found.

Low level severity issues

1. Mint function declared onlyOwner

Line	Code

7	function mint(address _to, uint256 _amount) public onlyOwner {
	_mint(_to, _amount);
	}

297 in MasterVault	paybVirtual.mint(msg.sender, _amount);

Description

MasterVault contract has enterStaking function, which is calling the mint function of this contract, but Mint function is onlyOwner so please make sure MasterVault is the owner of this contract. This same thing also applies to the burn function at line #11

Remediation

Please transfer the ownership of PaybVirtualToken contract to MasterVault.

Status: Acknowledged.

Informational

2. Solidity version

Line	Code
1	pragma solidity 0.6.12;

Description
Use the latest solidity version while contract deployment to prevent any compiler version level bugs.

Remediation
Please consider using 0.8.6, which is the latest version at the time of this audit

Status: Acknowledged.

3. Use SPDX License Identifier

Description
SPDX license identifier not provided in the source file. Before publishing, consider adding a comment containing "SPDX-License-Identifier: <SPDX-License>".

Remediation
Use "// SPDX-License-Identifier: MIT License" for non-open-source code.
Please see https://spdx.org for more information.

Status: Fixed

Functional test

File 1: MasterVault.sol

Function Names	Testing results
supplierExcluded	Passed
updateMasterSupplier	Passed
updateMultiplier	Passed
poolLength	Passed
setWithdrawPeriod	Passed
setWithdrawFee	Passed
add	Passed
set	Passed
updateStakingPool	Infinite loop possibility
getMultiplier	Passed
pendingPayb	Passed
massUpdatePools	Infinite loop possibility
updatePool	Passed
deposit	Passed
withdraw	Passed
enterStaking	Passed
leaveStaking	Passed
timeToLeaveStaking	Passed
currentSupplierAllowance	Passed

Function Names	Testing results
getPoolInvervalReturn	Passed
emergencyWithdraw	Passed
safePaybTransfer	Passed

File 2: PaybMockToken.sol

Function Names	Testing results
constructor	Passed

File 3: PaybVirtualToken.sol

Function Names	Testing results
mint	Owner must be MasterVault contract
burn	Owner must be MasterVault contract
safePaybTransfer	Passed

Automated Testing

Slither

```
INFO:Detectors:
PaybVirtualToken.safePaybTransfer(address,uint256) (MasterVault.sol#959-966) ignores return value by payb.transfer(_to,paybBal) (MasterVa
ult.sol#962)
PaybVirtualToken.safePaybTransfer(address,uint256) (MasterVault.sol#959-966) ignores return value by payb.transfer(_to,_amount) (MasterVa
ult.sol#964)
MasterVault.updatePool(uint256) (MasterVault.sol#1181-1197) ignores return value by payb.transferFrom(address(masterSupplier),address(pay
bVirtual),paybReward) (MasterVault.sol#1194)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
MasterVault.pendingPayb(uint256,address) (MasterVault.sol#1159-1170) performs a multiplication on the result of a division:
        -paybReward = multiplier.mul(paybPerBlock).mul(pool.allocPoint).div(totalAllocPoint) (MasterVault.sol#1166)
        -accPaybPerShare = accPaybPerShare.add(paybReward.mul(1e12).div(lpSupply)) (MasterVault.sol#1167)
MasterVault.updatePool(uint256) (MasterVault.sol#1181-1197) performs a multiplication on the result of a division:
        -paybReward = multiplier.mul(paybPerBlock).mul(pool.allocPoint).div(totalAllocPoint) (MasterVault.sol#1192)
        -pool.accPaybPerShare = pool.accPaybPerShare.add(paybReward.mul(1e12).div(lpSupply)) (MasterVault.sol#1195)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
MasterVault.updatePool(uint256) (MasterVault.sol#1181-1197) uses a dangerous strict equality:
        - lpSupply == 0 (MasterVault.sol#1187)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in MasterVault.add(uint256,IBEP20,bool) (MasterVault.sol#1105-1125):
        External calls:
        - massUpdatePools() (MasterVault.sol#1107)
                - payb.transferFrom(address(masterSupplier),address(paybVirtual),paybReward) (MasterVault.sol#1194)
        State variables written after the call(s):
        - poolInfo.push(PoolInfo(_lpToken,_allocPoint,lastRewardBlock,0)) (MasterVault.sol#1118-1123)
        - updateStakingPool() (MasterVault.sol#1124)
                - poolInfo[0].allocPoint = points (MasterVault.sol#1149)
        - totalAllocPoint = totalAllocPoint.add(_allocPoint) (MasterVault.sol#1117)
        - updateStakingPool() (MasterVault.sol#1124)
                - totalAllocPoint = totalAllocPoint.sub(poolInfo[0].allocPoint).add(points) (MasterVault.sol#1148)
Reentrancy in MasterVault.deposit(uint256,uint256) (MasterVault.sol#1200-1217):
        External calls:
        - updatePool(_pid) (MasterVault.sol#1204)
                - payb.transferFrom(address(masterSupplier),address(paybVirtual),paybReward) (MasterVault.sol#1194)
        - safePaybTransfer(msg.sender,pending) (MasterVault.sol#1208)
                - paybVirtual.safePaybTransfer(_to,_amount) (MasterVault.sol#1318)
                - payb.transferFrom(address(masterSupplier),address(paybVirtual),paybReward) (MasterVault.sol#1194)
        - safePaybTransfer(msg.sender,pending) (MasterVault.sol#1208)
                - paybVirtual.safePaybTransfer(_to,_amount) (MasterVault.sol#1318)
        - pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount) (MasterVault.sol#1212)
        State variables written after the call(s):
        - user.amount = user.amount.add(_amount) (MasterVault.sol#1213)
        - user.rewardDebt = user.amount.mul(pool.accPaybPerShare).div(1e12) (MasterVault.sol#1215)
Reentrancy in MasterVault.emergencyWithdraw(uint256) (MasterVault.sol#1304-1314):
        External calls:
        - pool.lpToken.safeTransfer(address(msg.sender),user.amount) (MasterVault.sol#1310)
        State variables written after the call(s):
        - user.amount = 0 (MasterVault.sol#1312)
        - user.rewardDebt = 0 (MasterVault.sol#1313)
Reentrancy in MasterVault.enterStaking(uint256) (MasterVault.sol#1239-1258):
        External calls:
        - updatePool(0) (MasterVault.sol#1242)
                - payb.transferFrom(address(masterSupplier),address(paybVirtual),paybReward) (MasterVault.sol#1194)
        - safePaybTransfer(msg.sender,pending) (MasterVault.sol#1246)
                - paybVirtual.safePaybTransfer(_to,_amount) (MasterVault.sol#1318)
        - pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount) (MasterVault.sol#1251)
        State variables written after the call(s):
        - user.amount = user.amount.add(_amount) (MasterVault.sol#1252)
        - user.rewardDebt = user.amount.mul(pool.accPaybPerShare).div(1e12) (MasterVault.sol#1254)
Reentrancy in MasterVault.leaveStaking(uint256) (MasterVault.sol#1261-1282):
        External calls:
        - updatePool(0) (MasterVault.sol#1265)
                - payb.transferFrom(address(masterSupplier),address(paybVirtual),paybReward) (MasterVault.sol#1194)
        - safePaybTransfer(msg.sender,pending) (MasterVault.sol#1268)
                - paybVirtual.safePaybTransfer(_to,_amount) (MasterVault.sol#1318)
        State variables written after the call(s):
        - user.amount = user.amount.sub(_amount) (MasterVault.sol#1274)
Reentrancy in MasterVault.leaveStaking(uint256) (MasterVault.sol#1261-1282):
        External calls:
        - updatePool(0) (MasterVault.sol#1265)
                - payb.transferFrom(address(masterSupplier),address(paybVirtual),paybReward) (MasterVault.sol#1194)
        - safePaybTransfer(msg.sender,pending) (MasterVault.sol#1268)
                - paybVirtual.safePaybTransfer(_to,_amount) (MasterVault.sol#1318)
        - pool.lpToken.safeTransfer(address(msg.sender),userReturn) (MasterVault.sol#1275)
```

```
        - pool.lpToken.safeTransfer(address(msg.sender),userReturn) (MasterVault.sol#1275)
        - pool.lpToken.safeTransfer(address(this.treasury()),fee) (MasterVault.sol#1276)
        State variables written after the call(s):
        - user.rewardDebt = user.amount.mul(pool.accPaybPerShare).div(1e12) (MasterVault.sol#1278)
Reentrancy in MasterVault.set(uint256,uint256,bool) (MasterVault.sol#1128-1138):
        External calls:
        - massUpdatePools() (MasterVault.sol#1130)
                - payb.transferFrom(address(masterSupplier),address(paybVirtual),paybReward) (MasterVault.sol#1194)
        State variables written after the call(s):
        - poolInfo[_pid].allocPoint = _allocPoint (MasterVault.sol#1133)
        - updateStakingPool() (MasterVault.sol#1136)
                - poolInfo[0].allocPoint = points (MasterVault.sol#1149)
        - totalAllocPoint = totalAllocPoint.sub(prevAllocPoint).add(_allocPoint) (MasterVault.sol#1135)
        - updateStakingPool() (MasterVault.sol#1136)
                - totalAllocPoint = totalAllocPoint.sub(poolInfo[0].allocPoint).add(points) (MasterVault.sol#1148)
Reentrancy in MasterVault.updatePool(uint256) (MasterVault.sol#1181-1197):
        External calls:
        - payb.transferFrom(address(masterSupplier),address(paybVirtual),paybReward) (MasterVault.sol#1194)
        State variables written after the call(s):
        - pool.accPaybPerShare = pool.accPaybPerShare.add(paybReward.mul(1e12).div(lpSupply)) (MasterVault.sol#1195)
        - pool.lastRewardBlock = block.number (MasterVault.sol#1196)
Reentrancy in MasterVault.withdraw(uint256,uint256) (MasterVault.sol#1220-1236):
        External calls:
        - updatePool(_pid) (MasterVault.sol#1225)
                - payb.transferFrom(address(masterSupplier),address(paybVirtual),paybReward) (MasterVault.sol#1194)
        - safePaybTransfer(msg.sender,pending) (MasterVault.sol#1228)
                - paybVirtual.safePaybTransfer(_to,_amount) (MasterVault.sol#1318)
        State variables written after the call(s):
        - user.amount = user.amount.sub(_amount) (MasterVault.sol#1231)
Reentrancy in MasterVault.withdraw(uint256,uint256) (MasterVault.sol#1220-1236):
        External calls:
        - updatePool(_pid) (MasterVault.sol#1225)
                - payb.transferFrom(address(masterSupplier),address(paybVirtual),paybReward) (MasterVault.sol#1194)
        - safePaybTransfer(msg.sender,pending) (MasterVault.sol#1228)
                - paybVirtual.safePaybTransfer(_to,_amount) (MasterVault.sol#1318)
        - pool.lpToken.safeTransfer(address(msg.sender),_amount) (MasterVault.sol#1232)
        State variables written after the call(s):
        - user.rewardDebt = user.amount.mul(pool.accPaybPerShare).div(1e12) (MasterVault.sol#1234)
```

```
                - paybVirtual.safePaybTransfer(_to,_amount) (MasterVault.sol#1318)
        - pool.lpToken.safeTransfer(address(msg.sender),_amount) (MasterVault.sol#1232)
        State variables written after the call(s):
        - user.rewardDebt = user.amount.mul(pool.accPaybPerShare).div(1e12) (MasterVault.sol#1234)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
BEP20.constructor(string,string).name (MasterVault.sol#677) shadows:
        - BEP20.name() (MasterVault.sol#693-695) (function)
        - IBEP20.name() (MasterVault.sol#22) (function)
BEP20.constructor(string,string).symbol (MasterVault.sol#677) shadows:
        - BEP20.symbol() (MasterVault.sol#707-709) (function)
        - IBEP20.symbol() (MasterVault.sol#17) (function)
BEP20.allowance(address,address).owner (MasterVault.sol#741) shadows:
        - Ownable.owner() (MasterVault.sol#593-595) (function)
BEP20._approve(address,address,uint256).owner (MasterVault.sol#913) shadows:
        - Ownable.owner() (MasterVault.sol#593-595) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Ownable.constructor().msgSender (MasterVault.sol#585) lacks a zero-check on :
                - _owner = msgSender (MasterVault.sol#586)
Ownable.authorizeOwnershipTransfer(address).authorizedAddress (MasterVault.sol#620) lacks a zero-check on :
                - _authorizedNewOwner = authorizedAddress (MasterVault.sol#621)
MasterVault.constructor(IBEP20,PaybVirtualToken,uint256,uint256,address)._treasury (MasterVault.sol#1048) lacks a zero-check on :
                - treasury = _treasury (MasterVault.sol#1056)
MasterVault.updateMasterSupplier(address).newSupplier (MasterVault.sol#1072) lacks a zero-check on :
                - masterSupplier = newSupplier (MasterVault.sol#1073)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in MasterVault.add(uint256,IBEP20,bool) (MasterVault.sol#1105-1125):
        External calls:
        - massUpdatePools() (MasterVault.sol#1107)
                - payb.transferFrom(address(masterSupplier),address(paybVirtual),paybReward) (MasterVault.sol#1194)
        State variables written after the call(s):
        - poolExistance[address(_lpToken)] = true (MasterVault.sol#1113)
Reentrancy in MasterVault.enterStaking(uint256) (MasterVault.sol#1239-1258):
        External calls:
        - updatePool(0) (MasterVault.sol#1242)
                - payb.transferFrom(address(masterSupplier),address(paybVirtual),paybReward) (MasterVault.sol#1194)
```

```
        - cooldownPeriodEnd[msg.sender] = block.timestamp.add(this.withdrawalCooldown()) (MasterVault.sol#1250)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in MasterVault.deposit(uint256,uint256) (MasterVault.sol#1200-1217):
        External calls:
        - updatePool(_pid) (MasterVault.sol#1204)
                - payb.transferFrom(address(masterSupplier),address(paybVirtual),paybReward) (MasterVault.sol#1194)
        - safePaybTransfer(msg.sender,pending) (MasterVault.sol#1208)
                - paybVirtual.safePaybTransfer(_to,_amount) (MasterVault.sol#1318)
        - pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount) (MasterVault.sol#1212)
        Event emitted after the call(s):
        - Deposit(msg.sender,_pid,_amount) (MasterVault.sol#1216)
Reentrancy in MasterVault.emergencyWithdraw(uint256) (MasterVault.sol#1304-1314):
        External calls:
        - pool.lpToken.safeTransfer(address(msg.sender),user.amount) (MasterVault.sol#1310)
        Event emitted after the call(s):
        - EmergencyWithdraw(msg.sender,_pid,user.amount) (MasterVault.sol#1311)
Reentrancy in MasterVault.enterStaking(uint256) (MasterVault.sol#1239-1258):
        External calls:
        - updatePool(0) (MasterVault.sol#1242)
                - payb.transferFrom(address(masterSupplier),address(paybVirtual),paybReward) (MasterVault.sol#1194)
        - safePaybTransfer(msg.sender,pending) (MasterVault.sol#1246)
                - paybVirtual.safePaybTransfer(_to,_amount) (MasterVault.sol#1318)
        - pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount) (MasterVault.sol#1251)
        - paybVirtual.mint(msg.sender,_amount) (MasterVault.sol#1256)
        Event emitted after the call(s):
        - Deposit(msg.sender,0,_amount) (MasterVault.sol#1257)
Reentrancy in MasterVault.leaveStaking(uint256) (MasterVault.sol#1261-1282):
        External calls:
        - updatePool(0) (MasterVault.sol#1265)
                - payb.transferFrom(address(masterSupplier),address(paybVirtual),paybReward) (MasterVault.sol#1194)
        - safePaybTransfer(msg.sender,pending) (MasterVault.sol#1268)
                - paybVirtual.safePaybTransfer(_to,_amount) (MasterVault.sol#1318)
        - pool.lpToken.safeTransfer(address(msg.sender),userReturn) (MasterVault.sol#1275)
        - pool.lpToken.safeTransfer(address(this.treasury()),fee) (MasterVault.sol#1276)
        - paybVirtual.burn(msg.sender,_amount) (MasterVault.sol#1280)
        Event emitted after the call(s):
        - Withdraw(msg.sender,0,_amount) (MasterVault.sol#1281)
        - Withdraw(msg.sender,0,_amount) (MasterVault.sol#1281)
Reentrancy in MasterVault.withdraw(uint256,uint256) (MasterVault.sol#1220-1236):
        External calls:
        - updatePool(_pid) (MasterVault.sol#1225)
                - payb.transferFrom(address(masterSupplier),address(paybVirtual),paybReward) (MasterVault.sol#1194)
        - safePaybTransfer(msg.sender,pending) (MasterVault.sol#1228)
                - paybVirtual.safePaybTransfer(_to,_amount) (MasterVault.sol#1318)
        - pool.lpToken.safeTransfer(address(msg.sender),_amount) (MasterVault.sol#1232)
        Event emitted after the call(s):
        - Withdraw(msg.sender,_pid,_amount) (MasterVault.sol#1235)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
MasterVault.leaveStaking(uint256) (MasterVault.sol#1261-1282) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(cooldownPeriodEnd[msg.sender] <= block.timestamp,withdraw: not yet possible) (MasterVault.sol#1271)
MasterVault.emergencyWithdraw(uint256) (MasterVault.sol#1304-1314) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(cooldownPeriodEnd[msg.sender] <= block.timestamp,withdraw: not yet possible) (MasterVault.sol#1308)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (MasterVault.sol#115-124) uses assembly
        - INLINE ASM (MasterVault.sol#122)
Address._verifyCallResult(bool,bytes,string) (MasterVault.sol#260-277) uses assembly
        - INLINE ASM (MasterVault.sol#269-272)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address.functionCall(address,bytes) (MasterVault.sol#168-170) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (MasterVault.sol#193-195) is never used and should be removed
Address.functionDelegateCall(address,bytes) (MasterVault.sol#242-244) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (MasterVault.sol#252-258) is never used and should be removed
Address.functionStaticCall(address,bytes) (MasterVault.sol#218-220) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (MasterVault.sol#228-234) is never used and should be removed
Address.sendValue(address,uint256) (MasterVault.sol#142-148) is never used and should be removed
BEP20._burnFrom(address,uint256) (MasterVault.sol#930-937) is never used and should be removed
Context._msgData() (MasterVault.sol#568-571) is never used and should be removed
SafeBEP20.safeApprove(IBEP20,address,uint256) (MasterVault.sol#507-521) is never used and should be removed
SafeBEP20.safeDecreaseAllowance(IBEP20,address,uint256) (MasterVault.sol#532-542) is never used and should be removed
SafeBEP20.safeIncreaseAllowance(IBEP20,address,uint256) (MasterVault.sol#523-530) is never used and should be removed
```

```
Address.functionStaticCall(address,bytes) (MasterVault.sol#218-220) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (MasterVault.sol#228-234) is never used and should be removed
Address.sendValue(address,uint256) (MasterVault.sol#142-148) is never used and should be removed
BEP20._burnFrom(address,uint256) (MasterVault.sol#930-937) is never used and should be removed
Context._msgData() (MasterVault.sol#568-571) is never used and should be removed
SafeBEP20.safeApprove(IBEP20,address,uint256) (MasterVault.sol#507-521) is never used and should be removed
SafeBEP20.safeDecreaseAllowance(IBEP20,address,uint256) (MasterVault.sol#532-542) is never used and should be removed
SafeBEP20.safeIncreaseAllowance(IBEP20,address,uint256) (MasterVault.sol#523-530) is never used and should be removed
SafeMath.div(uint256,uint256,string) (MasterVault.sol#453-456) is never used and should be removed
SafeMath.mod(uint256,uint256) (MasterVault.sol#415-418) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (MasterVault.sol#473-476) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (MasterVault.sol#287-291) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (MasterVault.sol#323-326) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (MasterVault.sol#333-336) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (MasterVault.sol#308-316) is never used and should be removed
SafeMath.trySub(uint256,uint256) (MasterVault.sol#298-301) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (MasterVault.sol#142-148):
        - (success) = recipient.call{value: amount}() (MasterVault.sol#146)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (MasterVault.sol#203-210):
        - (success,returndata) = target.call{value: value}(data) (MasterVault.sol#208)
Low level call in Address.functionStaticCall(address,bytes,string) (MasterVault.sol#228-234):
        - (success,returndata) = target.staticcall(data) (MasterVault.sol#232)
Low level call in Address.functionDelegateCall(address,bytes,string) (MasterVault.sol#252-258):
        - (success,returndata) = target.delegatecall(data) (MasterVault.sol#256)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter PaybVirtualToken.mint(address,uint256)._to (MasterVault.sol#942) is not in mixedCase
Parameter PaybVirtualToken.mint(address,uint256)._amount (MasterVault.sol#942) is not in mixedCase
Parameter PaybVirtualToken.burn(address,uint256)._from (MasterVault.sol#946) is not in mixedCase
Parameter PaybVirtualToken.burn(address,uint256)._amount (MasterVault.sol#946) is not in mixedCase
Parameter PaybVirtualToken.safePaybTransfer(address,uint256)._to (MasterVault.sol#959) is not in mixedCase
Parameter PaybVirtualToken.safePaybTransfer(address,uint256)._amount (MasterVault.sol#959) is not in mixedCase
Parameter MasterVault.setWithdrawPeriod(uint256)._withdrawPeriod (MasterVault.sol#1085) is not in mixedCase
Parameter MasterVault.setWithdrawFee(uint256)._withdrawFee (MasterVault.sol#1095) is not in mixedCase
Parameter MasterVault.add(uint256,IBEP20,bool)._allocPoint (MasterVault.sol#1105) is not in mixedCase
Parameter MasterVault.add(uint256,IBEP20,bool)._lpToken (MasterVault.sol#1105) is not in mixedCase
Parameter MasterVault.add(uint256,IBEP20,bool)._lpToken (MasterVault.sol#1105) is not in mixedCase
Parameter MasterVault.add(uint256,IBEP20,bool)._withUpdate (MasterVault.sol#1105) is not in mixedCase
Parameter MasterVault.set(uint256,uint256,bool)._pid (MasterVault.sol#1128) is not in mixedCase
Parameter MasterVault.set(uint256,uint256,bool)._allocPoint (MasterVault.sol#1128) is not in mixedCase
Parameter MasterVault.set(uint256,uint256,bool)._withUpdate (MasterVault.sol#1128) is not in mixedCase
Parameter MasterVault.getMultiplier(uint256,uint256)._from (MasterVault.sol#1154) is not in mixedCase
Parameter MasterVault.getMultiplier(uint256,uint256)._to (MasterVault.sol#1154) is not in mixedCase
Parameter MasterVault.pendingPayb(uint256,address)._pid (MasterVault.sol#1159) is not in mixedCase
Parameter MasterVault.pendingPayb(uint256,address)._user (MasterVault.sol#1159) is not in mixedCase
Parameter MasterVault.updatePool(uint256)._pid (MasterVault.sol#1181) is not in mixedCase
Parameter MasterVault.deposit(uint256,uint256)._pid (MasterVault.sol#1200) is not in mixedCase
Parameter MasterVault.deposit(uint256,uint256)._amount (MasterVault.sol#1200) is not in mixedCase
Parameter MasterVault.withdraw(uint256,uint256)._pid (MasterVault.sol#1220) is not in mixedCase
Parameter MasterVault.withdraw(uint256,uint256)._amount (MasterVault.sol#1220) is not in mixedCase
Parameter MasterVault.enterStaking(uint256)._amount (MasterVault.sol#1239) is not in mixedCase
Parameter MasterVault.leaveStaking(uint256)._amount (MasterVault.sol#1261) is not in mixedCase
Parameter MasterVault.emergencyWithdraw(uint256)._pid (MasterVault.sol#1304) is not in mixedCase
Parameter MasterVault.safePaybTransfer(address,uint256)._to (MasterVault.sol#1317) is not in mixedCase
Parameter MasterVault.safePaybTransfer(address,uint256)._amount (MasterVault.sol#1317) is not in mixedCase
Variable MasterVault.MULTIPLIER (MasterVault.sol#1008) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (MasterVault.sol#569)" inContext (MasterVault.sol#563-572)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
authorizedNewOwner() should be declared external:
        - Ownable.authorizedNewOwner() (MasterVault.sol#608-610)
renounceOwnership(address) should be declared external:
        - Ownable.renounceOwnership(address) (MasterVault.sol#644-649)
name() should be declared external:
        - BEP20.name() (MasterVault.sol#693-695)
decimals() should be declared external:
        - BEP20.decimals() (MasterVault.sol#700-702)
symbol() should be declared external:
        - BEP20.symbol() (MasterVault.sol#707-709)
totalSupply() should be declared external:
        - BEP20.totalSupply() (MasterVault.sol#714-716)
balanceOf(address) should be declared external:
```

```
symbol() should be declared external:
        - BEP20.symbol() (MasterVault.sol#707-709)
totalSupply() should be declared external:
        - BEP20.totalSupply() (MasterVault.sol#714-716)
balanceOf(address) should be declared external:
        - BEP20.balanceOf(address) (MasterVault.sol#721-723)
transfer(address,uint256) should be declared external:
        - BEP20.transfer(address,uint256) (MasterVault.sol#733-736)
allowance(address,address) should be declared external:
        - BEP20.allowance(address,address) (MasterVault.sol#741-743)
approve(address,uint256) should be declared external:
        - BEP20.approve(address,uint256) (MasterVault.sol#752-755)
transferFrom(address,address,uint256) should be declared external:
        - BEP20.transferFrom(address,address,uint256) (MasterVault.sol#769-781)
increaseAllowance(address,uint256) should be declared external:
        - BEP20.increaseAllowance(address,uint256) (MasterVault.sol#795-798)
decreaseAllowance(address,uint256) should be declared external:
        - BEP20.decreaseAllowance(address,uint256) (MasterVault.sol#814-821)
mint(uint256) should be declared external:
        - BEP20.mint(uint256) (MasterVault.sol#831-834)
mint(address,uint256) should be declared external:
        - PaybVirtualToken.mint(address,uint256) (MasterVault.sol#942-944)
burn(address,uint256) should be declared external:
        - PaybVirtualToken.burn(address,uint256) (MasterVault.sol#946-948)
safePaybTransfer(address,uint256) should be declared external:
        - PaybVirtualToken.safePaybTransfer(address,uint256) (MasterVault.sol#959-966)
updateMasterSupplier(address) should be declared external:
        - MasterVault.updateMasterSupplier(address) (MasterVault.sol#1072-1074)
updateMultiplier(uint256) should be declared external:
        - MasterVault.updateMultiplier(uint256) (MasterVault.sol#1076-1078)
add(uint256,IBEP20,bool) should be declared external:
        - MasterVault.add(uint256,IBEP20,bool) (MasterVault.sol#1105-1125)
set(uint256,uint256,bool) should be declared external:
        - MasterVault.set(uint256,uint256,bool) (MasterVault.sol#1128-1138)
deposit(uint256,uint256) should be declared external:
        - MasterVault.deposit(uint256,uint256) (MasterVault.sol#1200-1217)
withdraw(uint256,uint256) should be declared external:
        - MasterVault.withdraw(uint256,uint256) (MasterVault.sol#1220-1236)
enterStaking(uint256) should be declared external:
        - MasterVault.enterStaking(uint256) (MasterVault.sol#1239-1258)
leaveStaking(uint256) should be declared external:
        - MasterVault.leaveStaking(uint256) (MasterVault.sol#1261-1282)
timeToLeaveStaking(address) should be declared external:
        - MasterVault.timeToLeaveStaking(address) (MasterVault.sol#1285-1287)
currentSupplierAllowance() should be declared external:
        - MasterVault.currentSupplierAllowance() (MasterVault.sol#1290-1292)
getPoolInvervalReturn(uint256,uint256) should be declared external:
        - MasterVault.getPoolInvervalReturn(uint256,uint256) (MasterVault.sol#1297-1301)
emergencyWithdraw(uint256) should be declared external:
        - MasterVault.emergencyWithdraw(uint256) (MasterVault.sol#1304-1314)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:MasterVault.sol analyzed (9 contracts with 75 detectors), 118 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

PaybVirtualToken.sol

```
INFO:Detectors:
PaybVirtualToken.safePaybTransfer(address,uint256) (PaybVirtualToken.sol#851-858) ignores return value by payb.transfer(_to,paybBal) (Payb
VirtualToken.sol#854)
PaybVirtualToken.safePaybTransfer(address,uint256) (PaybVirtualToken.sol#851-858) ignores return value by payb.transfer(_to,_amount) (Payb
VirtualToken.sol#856)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
BEP20.constructor(string,string).name (PaybVirtualToken.sol#569) shadows:
        - BEP20.name() (PaybVirtualToken.sol#585-587) (function)
        - IBEP20.name() (PaybVirtualToken.sol#404) (function)
BEP20.constructor(string,string).symbol (PaybVirtualToken.sol#569) shadows:
        - BEP20.symbol() (PaybVirtualToken.sol#599-601) (function)
        - IBEP20.symbol() (PaybVirtualToken.sol#399) (function)
BEP20.allowance(address,address).owner (PaybVirtualToken.sol#633) shadows:
        - Ownable.owner() (PaybVirtualToken.sol#508-510) (function)
BEP20._approve(address,address,uint256).owner (PaybVirtualToken.sol#805) shadows:
        - Ownable.owner() (PaybVirtualToken.sol#508-510) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Address.isContract(address) (PaybVirtualToken.sol#220-229) uses assembly
        - INLINE ASM (PaybVirtualToken.sol#227)
Address._verifyCallResult(bool,bytes,string) (PaybVirtualToken.sol#365-382) uses assembly
        - INLINE ASM (PaybVirtualToken.sol#374-377)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address._verifyCallResult(bool,bytes,string) (PaybVirtualToken.sol#365-382) is never used and should be removed
Address.functionCall(address,bytes) (PaybVirtualToken.sol#273-275) is never used and should be removed
Address.functionCall(address,bytes,string) (PaybVirtualToken.sol#283-285) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (PaybVirtualToken.sol#298-300) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (PaybVirtualToken.sol#308-315) is never used and should be removed
Address.functionDelegateCall(address,bytes) (PaybVirtualToken.sol#347-349) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (PaybVirtualToken.sol#357-363) is never used and should be removed
Address.functionStaticCall(address,bytes) (PaybVirtualToken.sol#323-325) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (PaybVirtualToken.sol#333-339) is never used and should be removed
Address.isContract(address) (PaybVirtualToken.sol#220-229) is never used and should be removed
Address.sendValue(address,uint256) (PaybVirtualToken.sol#247-253) is never used and should be removed
```

```
Address.functionStaticCall(address,bytes) (PaybVirtualToken.sol#323-325) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (PaybVirtualToken.sol#333-339) is never used and should be removed
Address.isContract(address) (PaybVirtualToken.sol#220-229) is never used and should be removed
Address.sendValue(address,uint256) (PaybVirtualToken.sol#247-253) is never used and should be removed
BEP20._burnFrom(address,uint256) (PaybVirtualToken.sol#822-829) is never used and should be removed
Context._msgData() (PaybVirtualToken.sol#485-488) is never used and should be removed
SafeMath.div(uint256,uint256) (PaybVirtualToken.sol#120-123) is never used and should be removed
SafeMath.div(uint256,uint256,string) (PaybVirtualToken.sol#175-178) is never used and should be removed
SafeMath.mod(uint256,uint256) (PaybVirtualToken.sol#137-140) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (PaybVirtualToken.sol#195-198) is never used and should be removed
SafeMath.mul(uint256,uint256) (PaybVirtualToken.sol#101-106) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (PaybVirtualToken.sol#9-13) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (PaybVirtualToken.sol#45-48) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (PaybVirtualToken.sol#55-58) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (PaybVirtualToken.sol#30-38) is never used and should be removed
SafeMath.trySub(uint256,uint256) (PaybVirtualToken.sol#20-23) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (PaybVirtualToken.sol#247-253):
        - (success) = recipient.call{value: amount}() (PaybVirtualToken.sol#251)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (PaybVirtualToken.sol#308-315):
        - (success,returndata) = target.call{value: value}(data) (PaybVirtualToken.sol#313)
Low level call in Address.functionStaticCall(address,bytes,string) (PaybVirtualToken.sol#333-339):
        - (success,returndata) = target.staticcall(data) (PaybVirtualToken.sol#337)
Low level call in Address.functionDelegateCall(address,bytes,string) (PaybVirtualToken.sol#357-363):
        - (success,returndata) = target.delegatecall(data) (PaybVirtualToken.sol#361)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter PaybVirtualToken.mint(address,uint256)._to (PaybVirtualToken.sol#834) is not in mixedCase
Parameter PaybVirtualToken.mint(address,uint256)._amount (PaybVirtualToken.sol#834) is not in mixedCase
Parameter PaybVirtualToken.burn(address,uint256)._from (PaybVirtualToken.sol#838) is not in mixedCase
Parameter PaybVirtualToken.burn(address,uint256)._amount (PaybVirtualToken.sol#838) is not in mixedCase
Parameter PaybVirtualToken.safePaybTransfer(address,uint256)._to (PaybVirtualToken.sol#851) is not in mixedCase
Parameter PaybVirtualToken.safePaybTransfer(address,uint256)._amount (PaybVirtualToken.sol#851) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
```

```
INFO:Detectors:
Redundant expression "this (PaybVirtualToken.sol#486)" inContext (PaybVirtualToken.sol#480-489)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (PaybVirtualToken.sol#527-530)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (PaybVirtualToken.sol#536-540)
name() should be declared external:
        - BEP20.name() (PaybVirtualToken.sol#585-587)
decimals() should be declared external:
        - BEP20.decimals() (PaybVirtualToken.sol#592-594)
symbol() should be declared external:
        - BEP20.symbol() (PaybVirtualToken.sol#599-601)
totalSupply() should be declared external:
        - BEP20.totalSupply() (PaybVirtualToken.sol#606-608)
balanceOf(address) should be declared external:
        - BEP20.balanceOf(address) (PaybVirtualToken.sol#613-615)
transfer(address,uint256) should be declared external:
        - BEP20.transfer(address,uint256) (PaybVirtualToken.sol#625-628)
allowance(address,address) should be declared external:
        - BEP20.allowance(address,address) (PaybVirtualToken.sol#633-635)
approve(address,uint256) should be declared external:
        - BEP20.approve(address,uint256) (PaybVirtualToken.sol#644-647)
transferFrom(address,address,uint256) should be declared external:
        - BEP20.transferFrom(address,address,uint256) (PaybVirtualToken.sol#661-673)
increaseAllowance(address,uint256) should be declared external:
        - BEP20.increaseAllowance(address,uint256) (PaybVirtualToken.sol#687-690)
decreaseAllowance(address,uint256) should be declared external:
        - BEP20.decreaseAllowance(address,uint256) (PaybVirtualToken.sol#706-713)
mint(uint256) should be declared external:
        - BEP20.mint(uint256) (PaybVirtualToken.sol#723-726)
mint(address,uint256) should be declared external:
        - PaybVirtualToken.mint(address,uint256) (PaybVirtualToken.sol#834-836)
burn(address,uint256) should be declared external:
        - PaybVirtualToken.burn(address,uint256) (PaybVirtualToken.sol#838-840)
safePaybTransfer(address,uint256) should be declared external:
```

```
        - PaybVirtualToken.burn(address,uint256) (PaybVirtualToken.sol#838-840)
safePaybTransfer(address,uint256) should be declared external:
        - PaybVirtualToken.safePaybTransfer(address,uint256) (PaybVirtualToken.sol#851-858)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:PaybVirtualToken.sol analyzed (7 contracts with 75 detectors), 59 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

PaybMockToken.sol

```
INFO:Detectors:
BEP20.constructor(string,string).name (PaybMockToken.sol#568) shadows:
        - BEP20.name() (PaybMockToken.sol#584-586) (function)
        - IBEP20.name() (PaybMockToken.sol#403) (function)
BEP20.constructor(string,string).symbol (PaybMockToken.sol#568) shadows:
        - BEP20.symbol() (PaybMockToken.sol#598-600) (function)
        - IBEP20.symbol() (PaybMockToken.sol#398) (function)
BEP20.allowance(address,address).owner (PaybMockToken.sol#632) shadows:
        - Ownable.owner() (PaybMockToken.sol#507-509) (function)
BEP20._approve(address,address,uint256).owner (PaybMockToken.sol#804) shadows:
        - Ownable.owner() (PaybMockToken.sol#507-509) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Address.isContract(address) (PaybMockToken.sol#219-228) uses assembly
        - INLINE ASM (PaybMockToken.sol#226)
Address._verifyCallResult(bool,bytes,string) (PaybMockToken.sol#364-381) uses assembly
        - INLINE ASM (PaybMockToken.sol#373-376)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address._verifyCallResult(bool,bytes,string) (PaybMockToken.sol#364-381) is never used and should be removed
Address.functionCall(address,bytes) (PaybMockToken.sol#272-274) is never used and should be removed
Address.functionCall(address,bytes,string) (PaybMockToken.sol#282-284) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (PaybMockToken.sol#297-299) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (PaybMockToken.sol#307-314) is never used and should be removed
Address.functionDelegateCall(address,bytes) (PaybMockToken.sol#346-348) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (PaybMockToken.sol#356-362) is never used and should be removed
Address.functionStaticCall(address,bytes) (PaybMockToken.sol#322-324) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (PaybMockToken.sol#332-338) is never used and should be removed
Address.isContract(address) (PaybMockToken.sol#219-228) is never used and should be removed
Address.sendValue(address,uint256) (PaybMockToken.sol#246-252) is never used and should be removed
BEP20._burn(address,uint256) (PaybMockToken.sol#782-788) is never used and should be removed
BEP20._burnFrom(address,uint256) (PaybMockToken.sol#821-828) is never used and should be removed
Context._msgData() (PaybMockToken.sol#484-487) is never used and should be removed
SafeMath.div(uint256,uint256) (PaybMockToken.sol#120-123) is never used and should be removed
SafeMath.div(uint256,uint256,string) (PaybMockToken.sol#175-178) is never used and should be removed
SafeMath.mod(uint256,uint256) (PaybMockToken.sol#137-140) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (PaybMockToken.sol#195-198) is never used and should be removed
SafeMath.mul(uint256,uint256) (PaybMockToken.sol#101-106) is never used and should be removed
SafeMath.sub(uint256,uint256) (PaybMockToken.sol#86-89) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (PaybMockToken.sol#9-13) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (PaybMockToken.sol#45-48) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (PaybMockToken.sol#55-58) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (PaybMockToken.sol#30-38) is never used and should be removed
SafeMath.trySub(uint256,uint256) (PaybMockToken.sol#20-23) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (PaybMockToken.sol#246-252):
        - (success) = recipient.call{value: amount}() (PaybMockToken.sol#250)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (PaybMockToken.sol#307-314):
        - (success,returndata) = target.call{value: value}(data) (PaybMockToken.sol#312)
Low level call in Address.functionStaticCall(address,bytes,string) (PaybMockToken.sol#332-338):
        - (success,returndata) = target.staticcall(data) (PaybMockToken.sol#336)
Low level call in Address.functionDelegateCall(address,bytes,string) (PaybMockToken.sol#356-362):
        - (success,returndata) = target.delegatecall(data) (PaybMockToken.sol#360)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Redundant expression "this (PaybMockToken.sol#485)" inContext (PaybMockToken.sol#479-488)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (PaybMockToken.sol#526-529)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (PaybMockToken.sol#535-539)
name() should be declared external:
        - BEP20.name() (PaybMockToken.sol#584-586)
decimals() should be declared external:
        - BEP20.decimals() (PaybMockToken.sol#591-593)
symbol() should be declared external:
        - BEP20.symbol() (PaybMockToken.sol#598-600)
totalSupply() should be declared external:
```

```
totalSupply() should be declared external:
        - BEP20.totalSupply() (PaybMockToken.sol#605-607)
balanceOf(address) should be declared external:
        - BEP20.balanceOf(address) (PaybMockToken.sol#612-614)
transfer(address,uint256) should be declared external:
        - BEP20.transfer(address,uint256) (PaybMockToken.sol#624-627)
allowance(address,address) should be declared external:
        - BEP20.allowance(address,address) (PaybMockToken.sol#632-634)
approve(address,uint256) should be declared external:
        - BEP20.approve(address,uint256) (PaybMockToken.sol#643-646)
transferFrom(address,address,uint256) should be declared external:
        - BEP20.transferFrom(address,address,uint256) (PaybMockToken.sol#660-672)
increaseAllowance(address,uint256) should be declared external:
        - BEP20.increaseAllowance(address,uint256) (PaybMockToken.sol#686-689)
decreaseAllowance(address,uint256) should be declared external:
        - BEP20.decreaseAllowance(address,uint256) (PaybMockToken.sol#705-712)
mint(uint256) should be declared external:
        - BEP20.mint(uint256) (PaybMockToken.sol#722-725)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:PaybMockToken.sol analyzed (7 contracts with 75 detectors), 50 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
root@server:/chetan/gaza/mycontracts#
```

Solidity Static Analysis

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SafeBEP20.safeApprove(contract IBEP20,address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 46:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SafeBEP20.safeIncreaseAllowance(contract IBEP20,address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 62:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SafeBEP20.safeDecreaseAllowance(contract IBEP20,address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 71:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Address._functionCallWithValue(address,bytes,uint256,string): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 134:4:

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases.
Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 33:8:

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases.
Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 152:16:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree.
That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 292:44:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree.
That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 313:53:

Gas costs:

Gas requirement of function BEP20.transferOwnership is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 64:4:

Gas costs:

Gas requirement of function PaybVirtualToken.transferOwnership is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 64:4:

Gas costs:

Gas requirement of function MasterVault.transferOwnership is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 64:4:

This on local calls:

Use of "this" for local functions: Never use "this" to call functions in the same contract, it only consumes more gas than normal local calls.
more
Pos: 292:64:

This on local calls:

Use of "this" for local functions: Never use "this" to call functions in the same contract, it only consumes more gas than normal local calls.
more
Pos: 314:38:

This on local calls:

Use of "this" for local functions: Never use "this" to call functions in the same contract, it only consumes more gas than normal local calls.
more
Pos: 318:46:

ERC20:

ERC20 contract's "decimals" function should have "uint8" as return type
more
Pos: 14:4:

Miscellaneous

Constant/View/Pure functions:

SafeMath.sub(uint256,uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 46:4:

Constant/View/Pure functions:

SafeMath.div(uint256,uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 107:4:

Constant/View/Pure functions:

SafeMath.mod(uint256,uint256) : Is constant but potentially should not be. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 147:4:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 44:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 266:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 306:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 313:12:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 350:12:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.

Pos: 90:16:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.

Pos: 129:20:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.

Pos: 180:24:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.

Pos: 183:20:

Solhint Linter

```
contracts/MasterVault.sol:2:1: Error: Compiler version 0.6.12 does not satisfy the r semver
requirement
contracts/MasterVault.sol:5:8: Error: Use double quotes for string literals
contracts/MasterVault.sol:6:8: Error: Use double quotes for string literals
contracts/MasterVault.sol:7:8: Error: Use double quotes for string literals
contracts/MasterVault.sol:50:20: Error: Variable name must be in mixedCase
contracts/MasterVault.sol:110:47: Error: Use double quotes for string literals
contracts/MasterVault.sol:235:87: Error: Use double quotes for string literals
contracts/MasterVault.sol:243:29: Error: Use double quotes for string literals
contracts/MasterVault.sol:263:29: Error: Use double quotes for string literals
contracts/MasterVault.sol:292:45: Error: Avoid to make time-based decisions in your business
logic
contracts/MasterVault.sol:313:54: Error: Avoid to make time-based decisions in your business
logic
contracts/MasterVault.sol:327:52: Error: Visibility modifier must be first in list of modifiers
contracts/MasterVault.sol:332:46: Error: Visibility modifier must be first in list of modifiers
contracts/MasterVault.sol:339:74: Error: Visibility modifier must be first in list of modifiers
contracts/MasterVault.sol:350:54: Error: Avoid to make time-based decisions in your business
logic
```

Disclaimer

Quillhash audit is not a security warranty, investment advice, or an endorsement of the Paybswap platform. This audit does not provide a security or correctness guarantee of the audited smart contracts. The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the Paybswap Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

Closing Summary

In this report, we have considered the security of Paybswap platform. We performed our audit according to the procedure described above.

The audit showed several medium, low and informational severity issues. In the end, the majority of the issues were fixed and acknowledged by the Auditee. Other issues still remained unfixed as the internal team needs further discussion.

paybswap

QuillAudits

Canada, India, Singapore and United Kingdom

audits.quillhash.com

audits@quillhash.com