0x Guard

Smart contracts security assessment

Final report

Tariff: Standard

PulsePuma

August 2023

0xguard.com

hello@0xguard.com

Contents

Introduction

The report has been prepared for **PulsePuma**.

The audited PulsePuma project is Masterchef-like farm minting ERC20 token as reward.

Contracts were provided in .sol files:

SHA-1(masterchef.sol) = a35afb1564fc7a6f6355978ccc99bb17cc9a52a0

SHA-1(puma.sol) = f9fea465bfb1689d0e8780c0ec9fd77e3b6e7ee1

The updated code was deployed to the Pulse chain:

PumaToken 0xe206676C1d0C3CBA48d2Df32C05015cC7b837ADc

Masterchef 0xE29b9DA164285bDdeef0fD57153dAaeCb5fC29ba

Name	PulsePuma
Audit date	2023-08-02 - 2023-08-08
Language	Solidity
Platform	Pulse Chain

Contracts checked

Name	Address
PumaToken	0xe206676C1d0C3CBA48d2Df32C05015cC7b837ADc
MasterChef	0xE29b9DA164285bDdeef0fD57153dAaeCb5fC29ba
SafeMath	

Procedure

We perform our audit according to the following procedure:

Automated analysis

- Scanning the project's smart contracts with several publicly available automated Solidity analysis tools
- Manual verification (reject or confirm) all the issues found by the tools

Manual audit

- Manually analyze smart contracts for security vulnerabilities
- Smart contracts' logic check

Known vulnerabilities checked

Title	Check result
Unencrypted Private Data On-Chain	passed
Code With No Effects	passed
Message call with hardcoded gas amount	passed
Typographical Error	passed
DoS With Block Gas Limit	passed
Presence of unused variables	passed
Incorrect Inheritance Order	passed
Requirement Violation	passed
Weak Sources of Randomness from Chain Attributes	passed
Shadowing State Variables	passed

Incorrect Constructor Name	passed
Block values as a proxy for time	passed
Authorization through tx.origin	passed
DoS with Failed Call	passed
Delegatecall to Untrusted Callee	passed
Use of Deprecated Solidity Functions	passed
Assert Violation	passed
State Variable Default Visibility	passed
Reentrancy	passed
Unprotected SELFDESTRUCT Instruction	passed
Unprotected Ether Withdrawal	passed
Unchecked Call Return Value	passed
Floating Pragma	not passed
Outdated Compiler Version	passed
Integer Overflow and Underflow	passed
Function Default Visibility	passed

Classification of issue severity

High severity	High severity issues can cause a significant or full loss of funds, change of contract ownership, major interference with contract logic. Such issues require immediate attention.
Medium severity	Medium severity issues do not pose an immediate risk, but can be detrimental to the client's reputation if exploited. Medium severity issues may lead to a contract failure and can be fixed by modifying the contract state or redeployment. Such issues require attention.

Low severity Low severity issues do not cause significant destruction to the contract's functionality. Such issues are recommended to be taken into consideration.

Issues

High severity issues

1. Delegates are not transferred (PumaToken)
Status: Fixed

Balance checkpoints tracking relies on the internal `_moveDelegates` function, which must be called with every token movement. However, `transfer` function inherited from the standard ERC20 implementation doesn't support this functionality. See more in this Medium post.

Recommendation: Import ERC20 implementation from OpenZeppelin and include `_moveDelegates` call into the `ERC20._afterTokenTransfer` hook.

2. Owner excessive rights (MasterChef)
Status: Partially fixed

The owner has an ability to set an arbitrary emission rate, effectively resulting in PUMA token can be minted under owner's control.

Pool's deposit fee can be set up to 100%, owner's frontrunning transaction can be used to fully seize incoming deposit.

Recommendation: Limit emission rate from above, use MultiSig&Timelock for ownership.

Medium severity issues

1. Duplicating pools (MasterChef)

Status: Fixed

Pools with the same `lpToken` address are allowed. Let us assume there's 2 duplicating pools, one of which is disabled by its allocation set to 0. Then the `updatePool` function accounts disabled pool in `lpSupply = pool.lpToken.balanceOf(address(this))` line, reducing user's reward.

Recommendation: Include pool's balance to the PoolInfo structure and use it in the `updatePool`.

Low severity issues

1. Tokens with transfer fees aren't supported (MasterChef)

Status: Open

Actual transferred amount is not checked in the `deposit` function.

Recommendation: The owner must not add pools for tokens with taxable transfers.

2. Wrong pragma version (SafeMath)

Status: Open

SafeMath library from the OpenZeppelin repository is used with a wrong compiler version. Solidity compiler supports safe arithmetic operations after 0.8 release, thus newer versions of SafeMath library don't include additional safety checks.

Conclusion

PulsePuma PumaToken, MasterChef, SafeMath contracts were audited. 2 high, 1 medium, 2 low severity issues were found.

1 high, 1 medium severity issues have been fixed in the update.

Disclaimer

Slither output

```
INFO:Detectors:
Ownable is re-used:
        - Ownable (contracts/masterchef.sol#358-408)
        - Ownable (contracts/puma.sol#358-408)
ReentrancyGuard is re-used:
        - ReentrancyGuard (contracts/masterchef.sol#594-649)
        - ReentrancyGuard (contracts/puma.sol#594-649)
Context is re-used:
        - Context (contracts/masterchef.sol#337-345)
        - Context (contracts/puma.sol#337-345)
MasterChef is re-used:
        - MasterChef (contracts/masterchef.sol#1188-1446)
        - MasterChef (contracts/puma.sol#1194-1453)
SafeERC20 is re-used:
        - SafeERC20 (contracts/masterchef.sol#423-480)
        - SafeERC20 (contracts/puma.sol#423-480)
SafeMath is re-used:
        - SafeMath (contracts/masterchef.sol#184-325)
        - SafeMath (contracts/puma.sol#184-325)
IERC20 is re-used:
        - IERC20 (contracts/masterchef.sol#484-573)
        - IERC20 (contracts/puma.sol#484-573)
Address is re-used:
        - Address (contracts/masterchef.sol#10-167)
        - Address (contracts/puma.sol#10-167)
ERC20 is re-used:
        - ERC20 (contracts/masterchef.sol#677-941)
        - ERC20 (contracts/puma.sol#677-941)
PumaToken is re-used:
        - PumaToken (contracts/masterchef.sol#946-1177)
        - PumaToken (contracts/puma.sol#946-1183)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#name-reused
INFO:Detectors:
MasterChef.safePumaTransfer(address,uint256) (contracts/masterchef.sol#1422-1429)
ignores return value by puma.transfer(_to,pumaBal) (contracts/masterchef.sol#1425)
MasterChef.safePumaTransfer(address,uint256) (contracts/masterchef.sol#1422-1429)
ignores return value by puma.transfer(_to,_amount) (contracts/masterchef.sol#1427)
```

```
MasterChef.safePumaTransfer(address,uint256) (contracts/puma.sol#1429-1436) ignores
return value by puma.transfer(_to,pumaBal) (contracts/puma.sol#1432)
MasterChef.safePumaTransfer(address,uint256) (contracts/puma.sol#1429-1436) ignores
return value by puma.transfer(_to,_amount) (contracts/puma.sol#1434)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-
transfer
INFO:Detectors:
MasterChef.pendingPuma(uint256,address) (contracts/masterchef.sol#1295-1306) performs a
multiplication on the result of a division:
        - pumaReward =
multiplier.mul(pumaPerBlock).mul(pool.allocPoint).div(totalAllocPoint) (contracts/
masterchef.sol#1302)
        - accPumaPerShare = accPumaPerShare.add(pumaReward.mul(1e12).div(lpSupply))
(contracts/masterchef.sol#1303)
MasterChef.updatePool(uint256) (contracts/masterchef.sol#1317-1333) performs a
multiplication on the result of a division:
        - pumaReward =
multiplier.mul(pumaPerBlock).mul(pool.allocPoint).div(totalAllocPoint) (contracts/
masterchef.sol#1328)
        - pool.accPumaPerShare =
pool.accPumaPerShare.add(pumaReward.mul(1e12).div(lpSupply)) (contracts/
masterchef.sol#1331)
MasterChef.depositReferral(uint256,uint256,address) (contracts/
masterchef.sol#1363-1390) performs a multiplication on the result of a division:
        - depositFee = _amount.mul(pool.depositFeeBP).div(10000) (contracts/
masterchef.sol#1376)
        - feeAddress1Share = depositFee.mul(70).div(100) (contracts/
masterchef.sol#1377)
MasterChef.pendingPuma(uint256,address) (contracts/puma.sol#1301-1312) performs a
multiplication on the result of a division:
        - pumaReward =
multiplier.mul(pumaPerBlock).mul(pool.allocPoint).div(totalAllocPoint) (contracts/
puma.sol#1308)
        - accPumaPerShare = accPumaPerShare.add(pumaReward.mul(1e12).div(lpSupply))
(contracts/puma.sol#1309)
MasterChef.updatePool(uint256) (contracts/puma.sol#1323-1339) performs a multiplication
on the result of a division:
        - pumaReward =
multiplier.mul(pumaPerBlock).mul(pool.allocPoint).div(totalAllocPoint) (contracts/
puma.sol#1334)
        - pool.accPumaPerShare =
pool.accPumaPerShare.add(pumaReward.mul(1e12).div(lpSupply)) (contracts/puma.sol#1337)
MasterChef.depositReferral(uint256,uint256,address) (contracts/puma.sol#1369-1397)
performs a multiplication on the result of a division:
        - depositFee = _amount.mul(pool.depositFeeBP).div(10000) (contracts/
puma.sol#1383)
        - feeAddress1Share = depositFee.mul(70).div(100) (contracts/puma.sol#1384)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-
multiply
INFO:Detectors:
PumaToken._writeCheckpoint(address,uint32,uint256,uint256) (contracts/
masterchef.sol#1152-1164) uses a dangerous strict equality:
        - nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock ==
blockNumber (contracts/masterchef.sol#1156)
MasterChef.updatePool(uint256) (contracts/masterchef.sol#1317-1333) uses a dangerous
strict equality:
        - lpSupply == 0 || pool.allocPoint == 0 (contracts/masterchef.sol#1323)
PumaToken._writeCheckpoint(address,uint32,uint256,uint256) (contracts/
puma.sol#1152-1170) uses a dangerous strict equality:
        - nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock ==
blockNumber (contracts/puma.sol#1162)
MasterChef.updatePool(uint256) (contracts/puma.sol#1323-1339) uses a dangerous strict
equality:
        - lpSupply == 0 || pool.allocPoint == 0 (contracts/puma.sol#1329)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-
strict-equalities
INFO:Detectors:
Reentrancy in MasterChef.add(uint256,IERC20,uint16,bool) (contracts/
masterchef.sol#1262-1276):
        External calls:
        - massUpdatePools() (contracts/masterchef.sol#1265)
                - puma.mint(devaddr,pumaReward.div(10)) (contracts/masterchef.sol#1329)
                - puma.mint(address(this),pumaReward) (contracts/masterchef.sol#1330)
        State variables written after the call(s):
        - poolInfo.push(PoolInfo(_lpToken,_allocPoint,lastRewardBlock,0,_depositFeeBP))
(contracts/masterchef.sol#1269-1275)
        MasterChef.poolInfo (contracts/masterchef.sol#1230) can be used in cross
function reentrancies:
        - MasterChef.add(uint256,IERC20,uint16,bool) (contracts/
masterchef.sol#1262-1276)
        - MasterChef.massUpdatePools() (contracts/masterchef.sol#1309-1314)
        - MasterChef.pendingPuma(uint256,address) (contracts/masterchef.sol#1295-1306)
        - MasterChef.poolInfo (contracts/masterchef.sol#1230)
        - MasterChef.poolLength() (contracts/masterchef.sol#1256-1258)
        - MasterChef.set(uint256,uint256,uint16,bool) (contracts/
masterchef.sol#1279-1287)
        - MasterChef.updatePool(uint256) (contracts/masterchef.sol#1317-1333)
        - totalAllocPoint = totalAllocPoint.add(_allocPoint) (contracts/
masterchef.sol#1268)
        MasterChef.totalAllocPoint (contracts/masterchef.sol#1234) can be used in cross
function reentrancies:
        - MasterChef.add(uint256,IERC20,uint16,bool) (contracts/
masterchef.sol#1262-1276)
        - MasterChef.pendingPuma(uint256,address) (contracts/masterchef.sol#1295-1306)
        - MasterChef.set(uint256,uint256,uint16,bool) (contracts/
masterchef.sol#1279-1287)
        - MasterChef.totalAllocPoint (contracts/masterchef.sol#1234)
        - MasterChef.updatePool(uint256) (contracts/masterchef.sol#1317-1333)
Reentrancy in MasterChef.add(uint256,IERC20,uint16,bool) (contracts/
puma.sol#1268-1282):
        External calls:
        - massUpdatePools() (contracts/puma.sol#1271)
                - puma.mint(devaddr,pumaReward.div(10)) (contracts/puma.sol#1335)
                - puma.mint(address(this),pumaReward) (contracts/puma.sol#1336)
        State variables written after the call(s):
        - poolInfo.push(PoolInfo(_lpToken,_allocPoint,lastRewardBlock,0,_depositFeeBP))
(contracts/puma.sol#1275-1281)
        MasterChef.poolInfo (contracts/puma.sol#1236) can be used in cross function
reentrancies:
        - MasterChef.add(uint256,IERC20,uint16,bool) (contracts/puma.sol#1268-1282)
        - MasterChef.massUpdatePools() (contracts/puma.sol#1315-1320)
        - MasterChef.pendingPuma(uint256,address) (contracts/puma.sol#1301-1312)
        - MasterChef.poolInfo (contracts/puma.sol#1236)
        - MasterChef.poolLength() (contracts/puma.sol#1262-1264)
        - MasterChef.set(uint256,uint256,uint16,bool) (contracts/puma.sol#1285-1293)
        - MasterChef.updatePool(uint256) (contracts/puma.sol#1323-1339)
        - totalAllocPoint = totalAllocPoint.add(_allocPoint) (contracts/puma.sol#1274)
        MasterChef.totalAllocPoint (contracts/puma.sol#1240) can be used in cross
function reentrancies:
        - MasterChef.add(uint256,IERC20,uint16,bool) (contracts/puma.sol#1268-1282)
        - MasterChef.pendingPuma(uint256,address) (contracts/puma.sol#1301-1312)
        - MasterChef.set(uint256,uint256,uint16,bool) (contracts/puma.sol#1285-1293)
        - MasterChef.totalAllocPoint (contracts/puma.sol#1240)
        - MasterChef.updatePool(uint256) (contracts/puma.sol#1323-1339)
Reentrancy in MasterChef.deposit(uint256,uint256) (contracts/masterchef.sol#1336-1360):
        External calls:
        - updatePool(_pid) (contracts/masterchef.sol#1339)
                - puma.mint(devaddr,pumaReward.div(10)) (contracts/masterchef.sol#1329)
                - puma.mint(address(this),pumaReward) (contracts/masterchef.sol#1330)
        - safePumaTransfer(msg.sender,pending) (contracts/masterchef.sol#1343)
                - puma.transfer(_to,pumaBal) (contracts/masterchef.sol#1425)
                - puma.transfer(_to,_amount) (contracts/masterchef.sol#1427)
        - pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount)
(contracts/masterchef.sol#1347)
        - pool.lpToken.safeTransfer(feeAddress,depositFee) (contracts/
masterchef.sol#1351)
        State variables written after the call(s):
        - user.amount = user.amount.add(_amount).sub(depositFee) (contracts/
masterchef.sol#1353)
        MasterChef.userInfo (contracts/masterchef.sol#1232) can be used in cross
function reentrancies:
        - MasterChef.pendingPuma(uint256,address) (contracts/masterchef.sol#1295-1306)
        - MasterChef.userInfo (contracts/masterchef.sol#1232)
Reentrancy in MasterChef.deposit(uint256,uint256) (contracts/masterchef.sol#1336-1360):
        External calls:
        - updatePool(_pid) (contracts/masterchef.sol#1339)
                - puma.mint(devaddr,pumaReward.div(10)) (contracts/masterchef.sol#1329)
                - puma.mint(address(this),pumaReward) (contracts/masterchef.sol#1330)
        - safePumaTransfer(msg.sender,pending) (contracts/masterchef.sol#1343)
                - puma.transfer(_to,pumaBal) (contracts/masterchef.sol#1425)
                - puma.transfer(_to,_amount) (contracts/masterchef.sol#1427)
        - pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount)
(contracts/masterchef.sol#1347)
        State variables written after the call(s):
        - user.amount = user.amount.add(_amount) (contracts/masterchef.sol#1355)
        MasterChef.userInfo (contracts/masterchef.sol#1232) can be used in cross
function reentrancies:
        - MasterChef.pendingPuma(uint256,address) (contracts/masterchef.sol#1295-1306)
        - MasterChef.userInfo (contracts/masterchef.sol#1232)
Reentrancy in MasterChef.deposit(uint256,uint256) (contracts/puma.sol#1342-1366):
        External calls:
        - updatePool(_pid) (contracts/puma.sol#1345)
                - puma.mint(devaddr,pumaReward.div(10)) (contracts/puma.sol#1335)
                - puma.mint(address(this),pumaReward) (contracts/puma.sol#1336)
        - safePumaTransfer(msg.sender,pending) (contracts/puma.sol#1349)
                - puma.transfer(_to,pumaBal) (contracts/puma.sol#1432)
                - puma.transfer(_to,_amount) (contracts/puma.sol#1434)
        - pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount)
(contracts/puma.sol#1353)
        - pool.lpToken.safeTransfer(feeAddress,depositFee) (contracts/puma.sol#1357)
        State variables written after the call(s):
        - user.amount = user.amount.add(_amount).sub(depositFee) (contracts/
puma.sol#1359)
        MasterChef.userInfo (contracts/puma.sol#1238) can be used in cross function
reentrancies:
        - MasterChef.pendingPuma(uint256,address) (contracts/puma.sol#1301-1312)
        - MasterChef.userInfo (contracts/puma.sol#1238)
        - user.rewardDebt = user.amount.mul(pool.accPumaPerShare).div(1e12) (contracts/
puma.sol#1364)
        MasterChef.userInfo (contracts/puma.sol#1238) can be used in cross function
reentrancies:
        - MasterChef.pendingPuma(uint256,address) (contracts/puma.sol#1301-1312)
        - MasterChef.userInfo (contracts/puma.sol#1238)
Reentrancy in MasterChef.deposit(uint256,uint256) (contracts/puma.sol#1342-1366):
        External calls:
        - updatePool(_pid) (contracts/puma.sol#1345)
                - puma.mint(devaddr,pumaReward.div(10)) (contracts/puma.sol#1335)
                - puma.mint(address(this),pumaReward) (contracts/puma.sol#1336)
        - safePumaTransfer(msg.sender,pending) (contracts/puma.sol#1349)
                - puma.transfer(_to,pumaBal) (contracts/puma.sol#1432)
                - puma.transfer(_to,_amount) (contracts/puma.sol#1434)
        - pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount)
(contracts/puma.sol#1353)
        State variables written after the call(s):
        - user.amount = user.amount.add(_amount) (contracts/puma.sol#1361)
        MasterChef.userInfo (contracts/puma.sol#1238) can be used in cross function
reentrancies:
        - MasterChef.pendingPuma(uint256,address) (contracts/puma.sol#1301-1312)
        - MasterChef.userInfo (contracts/puma.sol#1238)
Reentrancy in MasterChef.depositReferral(uint256,uint256,address) (contracts/
masterchef.sol#1363-1390):
        External calls:
        - updatePool(_pid) (contracts/masterchef.sol#1366)
                - puma.mint(devaddr,pumaReward.div(10)) (contracts/masterchef.sol#1329)
                - puma.mint(address(this),pumaReward) (contracts/masterchef.sol#1330)
        - safePumaTransfer(msg.sender,pending) (contracts/masterchef.sol#1370)
                - puma.transfer(_to,pumaBal) (contracts/masterchef.sol#1425)
                - puma.transfer(_to,_amount) (contracts/masterchef.sol#1427)
        - pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount)
(contracts/masterchef.sol#1374)
        - pool.lpToken.safeTransfer(feeAddress,feeAddress1Share) (contracts/
masterchef.sol#1380)
        - pool.lpToken.safeTransfer(referral,feeAddress2Share) (contracts/
masterchef.sol#1381)
        State variables written after the call(s):
        - user.amount = user.amount.add(_amount).sub(depositFee) (contracts/
masterchef.sol#1383)
        MasterChef.userInfo (contracts/masterchef.sol#1232) can be used in cross
function reentrancies:
        - MasterChef.pendingPuma(uint256,address) (contracts/masterchef.sol#1295-1306)
        - MasterChef.userInfo (contracts/masterchef.sol#1232)
Reentrancy in MasterChef.depositReferral(uint256,uint256,address) (contracts/
masterchef.sol#1363-1390):
        External calls:
        - updatePool(_pid) (contracts/masterchef.sol#1366)
                - puma.mint(devaddr,pumaReward.div(10)) (contracts/masterchef.sol#1329)
                - puma.mint(address(this),pumaReward) (contracts/masterchef.sol#1330)
        - safePumaTransfer(msg.sender,pending) (contracts/masterchef.sol#1370)
                - puma.transfer(_to,pumaBal) (contracts/masterchef.sol#1425)
                - puma.transfer(_to,_amount) (contracts/masterchef.sol#1427)
        - pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount)
(contracts/masterchef.sol#1374)
        State variables written after the call(s):
        - user.amount = user.amount.add(_amount) (contracts/masterchef.sol#1385)
        MasterChef.userInfo (contracts/masterchef.sol#1232) can be used in cross
function reentrancies:
        - MasterChef.pendingPuma(uint256,address) (contracts/masterchef.sol#1295-1306)
        - MasterChef.userInfo (contracts/masterchef.sol#1232)
Reentrancy in MasterChef.depositReferral(uint256,uint256,address) (contracts/
puma.sol#1369-1397):
        External calls:
        - updatePool(_pid) (contracts/puma.sol#1373)
                - puma.mint(devaddr,pumaReward.div(10)) (contracts/puma.sol#1335)
                - puma.mint(address(this),pumaReward) (contracts/puma.sol#1336)
        - safePumaTransfer(msg.sender,pending) (contracts/puma.sol#1377)
                - puma.transfer(_to,pumaBal) (contracts/puma.sol#1432)
                - puma.transfer(_to,_amount) (contracts/puma.sol#1434)
        - pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount)
(contracts/puma.sol#1381)
        - pool.lpToken.safeTransfer(feeAddress,feeAddress1Share) (contracts/
puma.sol#1387)
        - pool.lpToken.safeTransfer(referral,feeAddress2Share) (contracts/
puma.sol#1388)
        State variables written after the call(s):
        - user.amount = user.amount.add(_amount).sub(depositFee) (contracts/
puma.sol#1390)
        MasterChef.userInfo (contracts/puma.sol#1238) can be used in cross function
reentrancies:
        - MasterChef.pendingPuma(uint256,address) (contracts/puma.sol#1301-1312)
        - MasterChef.userInfo (contracts/puma.sol#1238)
Reentrancy in MasterChef.depositReferral(uint256,uint256,address) (contracts/
puma.sol#1369-1397):
        External calls:
        - updatePool(_pid) (contracts/puma.sol#1373)
                - puma.mint(devaddr,pumaReward.div(10)) (contracts/puma.sol#1335)
                - puma.mint(address(this),pumaReward) (contracts/puma.sol#1336)
        - safePumaTransfer(msg.sender,pending) (contracts/puma.sol#1377)
                - puma.transfer(_to,pumaBal) (contracts/puma.sol#1432)
                - puma.transfer(_to,_amount) (contracts/puma.sol#1434)
        - pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount)
(contracts/puma.sol#1381)
        State variables written after the call(s):
        - user.amount = user.amount.add(_amount) (contracts/puma.sol#1392)
        MasterChef.userInfo (contracts/puma.sol#1238) can be used in cross function
reentrancies:
        - MasterChef.pendingPuma(uint256,address) (contracts/puma.sol#1301-1312)
        - MasterChef.userInfo (contracts/puma.sol#1238)
Reentrancy in MasterChef.set(uint256,uint256,uint16,bool) (contracts/
masterchef.sol#1279-1287):
        External calls:
        - massUpdatePools() (contracts/masterchef.sol#1282)
                - puma.mint(devaddr,pumaReward.div(10)) (contracts/masterchef.sol#1329)
                - puma.mint(address(this),pumaReward) (contracts/masterchef.sol#1330)
        State variables written after the call(s):
        - poolInfo[_pid].allocPoint = _allocPoint (contracts/masterchef.sol#1285)
        MasterChef.poolInfo (contracts/masterchef.sol#1230) can be used in cross
function reentrancies:
        - MasterChef.add(uint256,IERC20,uint16,bool) (contracts/
masterchef.sol#1262-1276)
        - MasterChef.massUpdatePools() (contracts/masterchef.sol#1309-1314)
        - MasterChef.pendingPuma(uint256,address) (contracts/masterchef.sol#1295-1306)
        - MasterChef.poolInfo (contracts/masterchef.sol#1230)
        - MasterChef.poolLength() (contracts/masterchef.sol#1256-1258)
        - MasterChef.set(uint256,uint256,uint16,bool) (contracts/
masterchef.sol#1279-1287)
        - MasterChef.updatePool(uint256) (contracts/masterchef.sol#1317-1333)
        - poolInfo[_pid].depositFeeBP = _depositFeeBP (contracts/masterchef.sol#1286)
        MasterChef.poolInfo (contracts/masterchef.sol#1230) can be used in cross
function reentrancies:
        - MasterChef.add(uint256,IERC20,uint16,bool) (contracts/
masterchef.sol#1262-1276)
        - MasterChef.massUpdatePools() (contracts/masterchef.sol#1309-1314)
        - MasterChef.pendingPuma(uint256,address) (contracts/masterchef.sol#1295-1306)
        - MasterChef.poolInfo (contracts/masterchef.sol#1230)
        - MasterChef.poolLength() (contracts/masterchef.sol#1256-1258)
        - MasterChef.set(uint256,uint256,uint16,bool) (contracts/
masterchef.sol#1279-1287)
        - MasterChef.updatePool(uint256) (contracts/masterchef.sol#1317-1333)
        - totalAllocPoint =
totalAllocPoint.sub(poolInfo[_pid].allocPoint).add(_allocPoint) (contracts/
masterchef.sol#1284)
        MasterChef.totalAllocPoint (contracts/masterchef.sol#1234) can be used in cross
function reentrancies:
        - MasterChef.add(uint256,IERC20,uint16,bool) (contracts/
masterchef.sol#1262-1276)
        - MasterChef.pendingPuma(uint256,address) (contracts/masterchef.sol#1295-1306)
        - MasterChef.set(uint256,uint256,uint16,bool) (contracts/
masterchef.sol#1279-1287)
        - MasterChef.totalAllocPoint (contracts/masterchef.sol#1234)
        - MasterChef.updatePool(uint256) (contracts/masterchef.sol#1317-1333)
Reentrancy in MasterChef.set(uint256,uint256,uint16,bool) (contracts/
puma.sol#1285-1293):
        External calls:
        - massUpdatePools() (contracts/puma.sol#1288)
                - puma.mint(devaddr,pumaReward.div(10)) (contracts/puma.sol#1335)
                - puma.mint(address(this),pumaReward) (contracts/puma.sol#1336)
        State variables written after the call(s):
        - poolInfo[_pid].allocPoint = _allocPoint (contracts/puma.sol#1291)
```

```
	MasterChef.poolInfo (contracts/puma.sol#1236) can be used in cross function
reentrancies:
	- MasterChef.add(uint256,IERC20,uint16,bool) (contracts/puma.sol#1268-1282)
	- MasterChef.massUpdatePools() (contracts/puma.sol#1315-1320)
	- MasterChef.pendingPuma(uint256,address) (contracts/puma.sol#1301-1312)
	- MasterChef.poolInfo (contracts/puma.sol#1236)
	- MasterChef.poolLength() (contracts/puma.sol#1262-1264)
	- MasterChef.set(uint256,uint256,uint16,bool) (contracts/puma.sol#1285-1293)
	- MasterChef.updatePool(uint256) (contracts/puma.sol#1323-1339)
	- poolInfo[_pid].depositFeeBP = _depositFeeBP (contracts/puma.sol#1292)
	MasterChef.poolInfo (contracts/puma.sol#1236) can be used in cross function
reentrancies:
	- MasterChef.add(uint256,IERC20,uint16,bool) (contracts/puma.sol#1268-1282)
	- MasterChef.massUpdatePools() (contracts/puma.sol#1315-1320)
	- MasterChef.pendingPuma(uint256,address) (contracts/puma.sol#1301-1312)
	- MasterChef.poolInfo (contracts/puma.sol#1236)
	- MasterChef.poolLength() (contracts/puma.sol#1262-1264)
	- MasterChef.set(uint256,uint256,uint16,bool) (contracts/puma.sol#1285-1293)
	- MasterChef.updatePool(uint256) (contracts/puma.sol#1323-1339)
	- totalAllocPoint =
totalAllocPoint.sub(poolInfo[_pid].allocPoint).add(_allocPoint) (contracts/
puma.sol#1290)
	MasterChef.totalAllocPoint (contracts/puma.sol#1240) can be used in cross
function reentrancies:
	- MasterChef.add(uint256,IERC20,uint16,bool) (contracts/puma.sol#1268-1282)
	- MasterChef.pendingPuma(uint256,address) (contracts/puma.sol#1301-1312)
	- MasterChef.set(uint256,uint256,uint16,bool) (contracts/puma.sol#1285-1293)
	- MasterChef.totalAllocPoint (contracts/puma.sol#1240)
	- MasterChef.updatePool(uint256) (contracts/puma.sol#1323-1339)
Reentrancy in MasterChef.updateEmissionRate(uint256) (contracts/
masterchef.sol#1442-1445):
	External calls:
	- massUpdatePools() (contracts/masterchef.sol#1443)
		- puma.mint(devaddr,pumaReward.div(10)) (contracts/masterchef.sol#1329)
		- puma.mint(address(this),pumaReward) (contracts/masterchef.sol#1330)
	State variables written after the call(s):
	- pumaPerBlock = _pumaPerBlock (contracts/masterchef.sol#1444)
	MasterChef.pumaPerBlock (contracts/masterchef.sol#1223) can be used in cross
function reentrancies:
	- MasterChef.constructor(PumaToken,address,address,uint256,uint256) (contracts/
masterchef.sol#1242-1254)
```

```
        - MasterChef.pendingPuma(uint256,address) (contracts/masterchef.sol#1295-1306)
        - MasterChef.pumaPerBlock (contracts/masterchef.sol#1223)
        - MasterChef.updateEmissionRate(uint256) (contracts/masterchef.sol#1442-1445)
        - MasterChef.updatePool(uint256) (contracts/masterchef.sol#1317-1333)
Reentrancy in MasterChef.updateEmissionRate(uint256) (contracts/puma.sol#1449-1452):
        External calls:
        - massUpdatePools() (contracts/puma.sol#1450)
                - puma.mint(devaddr,pumaReward.div(10)) (contracts/puma.sol#1335)
                - puma.mint(address(this),pumaReward) (contracts/puma.sol#1336)
        State variables written after the call(s):
        - pumaPerBlock = _pumaPerBlock (contracts/puma.sol#1451)
        MasterChef.pumaPerBlock (contracts/puma.sol#1229) can be used in cross function
reentrancies:
        - MasterChef.constructor(PumaToken,address,address,uint256,uint256) (contracts/
puma.sol#1248-1260)
        - MasterChef.pendingPuma(uint256,address) (contracts/puma.sol#1301-1312)
        - MasterChef.pumaPerBlock (contracts/puma.sol#1229)
        - MasterChef.updateEmissionRate(uint256) (contracts/puma.sol#1449-1452)
        - MasterChef.updatePool(uint256) (contracts/puma.sol#1323-1339)
Reentrancy in MasterChef.updatePool(uint256) (contracts/masterchef.sol#1317-1333):
        External calls:
        - puma.mint(devaddr,pumaReward.div(10)) (contracts/masterchef.sol#1329)
        - puma.mint(address(this),pumaReward) (contracts/masterchef.sol#1330)
        State variables written after the call(s):
        - pool.accPumaPerShare =
pool.accPumaPerShare.add(pumaReward.mul(1e12).div(lpSupply)) (contracts/
masterchef.sol#1331)
        MasterChef.poolInfo (contracts/masterchef.sol#1230) can be used in cross
function reentrancies:
        - MasterChef.add(uint256,IERC20,uint16,bool) (contracts/
masterchef.sol#1262-1276)
        - MasterChef.massUpdatePools() (contracts/masterchef.sol#1309-1314)
        - MasterChef.pendingPuma(uint256,address) (contracts/masterchef.sol#1295-1306)
        - MasterChef.poolInfo (contracts/masterchef.sol#1230)
        - MasterChef.poolLength() (contracts/masterchef.sol#1256-1258)
        - MasterChef.set(uint256,uint256,uint16,bool) (contracts/
masterchef.sol#1279-1287)
        - MasterChef.updatePool(uint256) (contracts/masterchef.sol#1317-1333)
        - pool.lastRewardBlock = block.number (contracts/masterchef.sol#1332)
        MasterChef.poolInfo (contracts/masterchef.sol#1230) can be used in cross
function reentrancies:
        - MasterChef.add(uint256,IERC20,uint16,bool) (contracts/
masterchef.sol#1262-1276)
        - MasterChef.massUpdatePools() (contracts/masterchef.sol#1309-1314)
        - MasterChef.pendingPuma(uint256,address) (contracts/masterchef.sol#1295-1306)
        - MasterChef.poolInfo (contracts/masterchef.sol#1230)
        - MasterChef.poolLength() (contracts/masterchef.sol#1256-1258)
        - MasterChef.set(uint256,uint256,uint16,bool) (contracts/
masterchef.sol#1279-1287)
        - MasterChef.updatePool(uint256) (contracts/masterchef.sol#1317-1333)
Reentrancy in MasterChef.updatePool(uint256) (contracts/puma.sol#1323-1339):
        External calls:
        - puma.mint(devaddr,pumaReward.div(10)) (contracts/puma.sol#1335)
        - puma.mint(address(this),pumaReward) (contracts/puma.sol#1336)
        State variables written after the call(s):
        - pool.accPumaPerShare =
pool.accPumaPerShare.add(pumaReward.mul(1e12).div(lpSupply)) (contracts/puma.sol#1337)
        MasterChef.poolInfo (contracts/puma.sol#1236) can be used in cross function
reentrancies:
        - MasterChef.add(uint256,IERC20,uint16,bool) (contracts/puma.sol#1268-1282)
        - MasterChef.massUpdatePools() (contracts/puma.sol#1315-1320)
        - MasterChef.pendingPuma(uint256,address) (contracts/puma.sol#1301-1312)
        - MasterChef.poolInfo (contracts/puma.sol#1236)
        - MasterChef.poolLength() (contracts/puma.sol#1262-1264)
        - MasterChef.set(uint256,uint256,uint16,bool) (contracts/puma.sol#1285-1293)
        - MasterChef.updatePool(uint256) (contracts/puma.sol#1323-1339)
        - pool.lastRewardBlock = block.number (contracts/puma.sol#1338)
        MasterChef.poolInfo (contracts/puma.sol#1236) can be used in cross function
reentrancies:
        - MasterChef.add(uint256,IERC20,uint16,bool) (contracts/puma.sol#1268-1282)
        - MasterChef.massUpdatePools() (contracts/puma.sol#1315-1320)
        - MasterChef.pendingPuma(uint256,address) (contracts/puma.sol#1301-1312)
        - MasterChef.poolInfo (contracts/puma.sol#1236)
        - MasterChef.poolLength() (contracts/puma.sol#1262-1264)
        - MasterChef.set(uint256,uint256,uint16,bool) (contracts/puma.sol#1285-1293)
        - MasterChef.updatePool(uint256) (contracts/puma.sol#1323-1339)
Reentrancy in MasterChef.withdraw(uint256,uint256) (contracts/
masterchef.sol#1393-1408):
        External calls:
        - updatePool(_pid) (contracts/masterchef.sol#1397)
                - puma.mint(devaddr,pumaReward.div(10)) (contracts/masterchef.sol#1329)
                - puma.mint(address(this),pumaReward) (contracts/masterchef.sol#1330)
        - safePumaTransfer(msg.sender,pending) (contracts/masterchef.sol#1400)
                - puma.transfer(_to,pumaBal) (contracts/masterchef.sol#1425)
                - puma.transfer(_to,_amount) (contracts/masterchef.sol#1427)
        State variables written after the call(s):
        - user.amount = user.amount.sub(_amount) (contracts/masterchef.sol#1403)
        MasterChef.userInfo (contracts/masterchef.sol#1232) can be used in cross
function reentrancies:
        - MasterChef.pendingPuma(uint256,address) (contracts/masterchef.sol#1295-1306)
        - MasterChef.userInfo (contracts/masterchef.sol#1232)
Reentrancy in MasterChef.withdraw(uint256,uint256) (contracts/
masterchef.sol#1393-1408):
        External calls:
        - updatePool(_pid) (contracts/masterchef.sol#1397)
                - puma.mint(devaddr,pumaReward.div(10)) (contracts/masterchef.sol#1329)
                - puma.mint(address(this),pumaReward) (contracts/masterchef.sol#1330)
        - safePumaTransfer(msg.sender,pending) (contracts/masterchef.sol#1400)
                - puma.transfer(_to,pumaBal) (contracts/masterchef.sol#1425)
                - puma.transfer(_to,_amount) (contracts/masterchef.sol#1427)
        - pool.lpToken.safeTransfer(address(msg.sender),_amount) (contracts/
masterchef.sol#1404)
        State variables written after the call(s):
        - user.rewardDebt = user.amount.mul(pool.accPumaPerShare).div(1e12) (contracts/
masterchef.sol#1406)
        MasterChef.userInfo (contracts/masterchef.sol#1232) can be used in cross
function reentrancies:
        - MasterChef.pendingPuma(uint256,address) (contracts/masterchef.sol#1295-1306)
        - MasterChef.userInfo (contracts/masterchef.sol#1232)
Reentrancy in MasterChef.withdraw(uint256,uint256) (contracts/puma.sol#1400-1415):
        External calls:
        - updatePool(_pid) (contracts/puma.sol#1404)
                - puma.mint(devaddr,pumaReward.div(10)) (contracts/puma.sol#1335)
                - puma.mint(address(this),pumaReward) (contracts/puma.sol#1336)
        - safePumaTransfer(msg.sender,pending) (contracts/puma.sol#1407)
                - puma.transfer(_to,pumaBal) (contracts/puma.sol#1432)
                - puma.transfer(_to,_amount) (contracts/puma.sol#1434)
        State variables written after the call(s):
        - user.amount = user.amount.sub(_amount) (contracts/puma.sol#1410)
        MasterChef.userInfo (contracts/puma.sol#1238) can be used in cross function
reentrancies:
        - MasterChef.pendingPuma(uint256,address) (contracts/puma.sol#1301-1312)
        - MasterChef.userInfo (contracts/puma.sol#1238)
```

```
Reentrancy in MasterChef.withdraw(uint256,uint256) (contracts/puma.sol#1400-1415):
        External calls:
        - updatePool(_pid) (contracts/puma.sol#1404)
                - puma.mint(devaddr,pumaReward.div(10)) (contracts/puma.sol#1335)
                - puma.mint(address(this),pumaReward) (contracts/puma.sol#1336)
        - safePumaTransfer(msg.sender,pending) (contracts/puma.sol#1407)
                - puma.transfer(_to,pumaBal) (contracts/puma.sol#1432)
                - puma.transfer(_to,_amount) (contracts/puma.sol#1434)
        - pool.lpToken.safeTransfer(address(msg.sender),_amount) (contracts/
puma.sol#1411)
        State variables written after the call(s):
        - user.rewardDebt = user.amount.mul(pool.accPumaPerShare).div(1e12) (contracts/
puma.sol#1413)
        MasterChef.userInfo (contracts/puma.sol#1238) can be used in cross function
reentrancies:
        - MasterChef.pendingPuma(uint256,address) (contracts/puma.sol#1301-1312)
        - MasterChef.userInfo (contracts/puma.sol#1238)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-
vulnerabilities-1
INFO:Detectors:
ERC20.constructor(string,string).name (contracts/masterchef.sol#699) shadows:
        - ERC20.name() (contracts/masterchef.sol#715-717) (function)
        - IERC20.name() (contracts/masterchef.sol#503) (function)
ERC20.constructor(string,string).symbol (contracts/masterchef.sol#699) shadows:
        - ERC20.symbol() (contracts/masterchef.sol#723-725) (function)
        - IERC20.symbol() (contracts/masterchef.sol#498) (function)
ERC20.allowance(address,address).owner (contracts/masterchef.sol#764) shadows:
        - Ownable.owner() (contracts/masterchef.sol#375-377) (function)
ERC20._approve(address,address,uint256).owner (contracts/masterchef.sol#923) shadows:
        - Ownable.owner() (contracts/masterchef.sol#375-377) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-
shadowing
INFO:Detectors:
MasterChef.add(uint256,IERC20,uint16,bool) (contracts/masterchef.sol#1262-1276) should
emit an event for:
        - totalAllocPoint = totalAllocPoint.add(_allocPoint) (contracts/
masterchef.sol#1268)
MasterChef.set(uint256,uint256,uint16,bool) (contracts/masterchef.sol#1279-1287) should
emit an event for:
        - totalAllocPoint =
totalAllocPoint.sub(poolInfo[_pid].allocPoint).add(_allocPoint) (contracts/
masterchef.sol#1284)
```

```
MasterChef.updateEmissionRate(uint256) (contracts/masterchef.sol#1442-1445) should emit
an event for:
        - pumaPerBlock = _pumaPerBlock (contracts/masterchef.sol#1444)
MasterChef.add(uint256,IERC20,uint16,bool) (contracts/puma.sol#1268-1282) should emit
an event for:
        - totalAllocPoint = totalAllocPoint.add(_allocPoint) (contracts/puma.sol#1274)
MasterChef.set(uint256,uint256,uint16,bool) (contracts/puma.sol#1285-1293) should emit
an event for:
        - totalAllocPoint =
totalAllocPoint.sub(poolInfo[_pid].allocPoint).add(_allocPoint) (contracts/
puma.sol#1290)
MasterChef.updateEmissionRate(uint256) (contracts/puma.sol#1449-1452) should emit an
event for:
        - pumaPerBlock = _pumaPerBlock (contracts/puma.sol#1451)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-
arithmetic
INFO:Detectors:
MasterChef.constructor(PumaToken,address,address,uint256,uint256)._devaddr (contracts/
masterchef.sol#1244) lacks a zero-check on :
                - devaddr = _devaddr (contracts/masterchef.sol#1250)
MasterChef.constructor(PumaToken,address,address,uint256,uint256)._feeAddress1
(contracts/masterchef.sol#1245) lacks a zero-check on :
                - feeAddress = _feeAddress1 (contracts/masterchef.sol#1251)
MasterChef.dev(address)._devaddr (contracts/masterchef.sol#1432) lacks a zero-check
on :
                - devaddr = _devaddr (contracts/masterchef.sol#1434)
MasterChef.setFeeAddress1(address)._feeAddress1 (contracts/masterchef.sol#1437) lacks a
zero-check on :
                - feeAddress = _feeAddress1 (contracts/masterchef.sol#1439)
MasterChef.constructor(PumaToken,address,address,uint256,uint256)._devaddr (contracts/
puma.sol#1250) lacks a zero-check on :
                - devaddr = _devaddr (contracts/puma.sol#1256)
MasterChef.constructor(PumaToken,address,address,uint256,uint256)._feeAddress1
(contracts/puma.sol#1251) lacks a zero-check on :
                - feeAddress = _feeAddress1 (contracts/puma.sol#1257)
MasterChef.dev(address)._devaddr (contracts/puma.sol#1439) lacks a zero-check on :
                - devaddr = _devaddr (contracts/puma.sol#1441)
MasterChef.setFeeAddress1(address)._feeAddress1 (contracts/puma.sol#1444) lacks a zero-
check on :
                - feeAddress = _feeAddress1 (contracts/puma.sol#1446)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-
```

```
address-validation
INFO:Detectors:
MasterChef.updatePool(uint256) (contracts/masterchef.sol#1317-1333) has external calls
inside a loop: lpSupply = pool.lpToken.balanceOf(address(this)) (contracts/
masterchef.sol#1322)
MasterChef.updatePool(uint256) (contracts/masterchef.sol#1317-1333) has external calls
inside a loop: puma.mint(devaddr,pumaReward.div(10)) (contracts/masterchef.sol#1329)
MasterChef.updatePool(uint256) (contracts/masterchef.sol#1317-1333) has external calls
inside a loop: puma.mint(address(this),pumaReward) (contracts/masterchef.sol#1330)
MasterChef.updatePool(uint256) (contracts/puma.sol#1323-1339) has external calls inside
a loop: lpSupply = pool.lpToken.balanceOf(address(this)) (contracts/puma.sol#1328)
MasterChef.updatePool(uint256) (contracts/puma.sol#1323-1339) has external calls inside
a loop: puma.mint(devaddr,pumaReward.div(10)) (contracts/puma.sol#1335)
MasterChef.updatePool(uint256) (contracts/puma.sol#1323-1339) has external calls inside
a loop: puma.mint(address(this),pumaReward) (contracts/puma.sol#1336)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-
a-loop
INFO:Detectors:
PumaToken.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (contracts/
masterchef.sol#1018-1059) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= expiry,PUMA::delegateBySig: signature
expired) (contracts/masterchef.sol#1057)
PumaToken.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (contracts/
puma.sol#1018-1059) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= expiry,PUMA::delegateBySig: signature
expired) (contracts/puma.sol#1057)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-
timestamp
INFO:Detectors:
Address.isContract(address) (contracts/masterchef.sol#28-37) uses assembly
        - INLINE ASM (contracts/masterchef.sol#35)
Address._verifyCallResult(bool,bytes,string) (contracts/masterchef.sol#149-166) uses
assembly
        - INLINE ASM (contracts/masterchef.sol#158-161)
PumaToken.getChainId() (contracts/masterchef.sol#1171-1175) uses assembly
        - INLINE ASM (contracts/masterchef.sol#1173)
Address.isContract(address) (contracts/puma.sol#28-37) uses assembly
        - INLINE ASM (contracts/puma.sol#35)
Address._verifyCallResult(bool,bytes,string) (contracts/puma.sol#149-166) uses assembly
```

```
        - INLINE ASM (contracts/puma.sol#158-161)
PumaToken.getChainId() (contracts/puma.sol#1177-1181) uses assembly
        - INLINE ASM (contracts/puma.sol#1179)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Address.functionCall(address,bytes) (contracts/puma.sol#81-83) is never used and should
be removed
Address.functionCallWithValue(address,bytes,uint256) (contracts/puma.sol#106-108) is
never used and should be removed
Address.functionStaticCall(address,bytes) (contracts/puma.sol#131-133) is never used
and should be removed
Address.functionStaticCall(address,bytes,string) (contracts/puma.sol#141-147) is never
used and should be removed
Address.sendValue(address,uint256) (contracts/puma.sol#55-61) is never used and should
be removed
Context._msgData() (contracts/masterchef.sol#342-344) is never used and should be
removed
ERC20._burn(address,uint256) (contracts/puma.sol#902-908) is never used and should be
removed
ERC20._burnFrom(address,uint256) (contracts/masterchef.sol#937-940) is never used and
should be removed
ReentrancyGuard._reentrancyGuardEntered() (contracts/masterchef.sol#646-648) is never
used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (contracts/puma.sol#442-451) is never
used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (contracts/puma.sol#458-461) is
never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (contracts/puma.sol#453-456) is
never used and should be removed
SafeMath.mod(uint256,uint256) (contracts/puma.sol#305-307) is never used and should be
removed
SafeMath.mod(uint256,uint256,string) (contracts/puma.sol#321-324) is never used and
should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (contracts/masterchef.sol#5) allows old versions
Pragma version^0.8.0 (contracts/puma.sol#5) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-
versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (contracts/masterchef.sol#55-61):
        - (success) = recipient.call{value: amount}() (contracts/masterchef.sol#59)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string)
(contracts/masterchef.sol#116-123):
        - (success,returndata) = target.call{value: value}(data) (contracts/
masterchef.sol#121)
Low level call in Address.functionStaticCall(address,bytes,string) (contracts/
masterchef.sol#141-147):
        - (success,returndata) = target.staticcall(data) (contracts/masterchef.sol#145)
Low level call in Address.sendValue(address,uint256) (contracts/puma.sol#55-61):
        - (success) = recipient.call{value: amount}() (contracts/puma.sol#59)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string)
(contracts/puma.sol#116-123):
        - (success,returndata) = target.call{value: value}(data) (contracts/
puma.sol#121)
Low level call in Address.functionStaticCall(address,bytes,string) (contracts/
puma.sol#141-147):
        - (success,returndata) = target.staticcall(data) (contracts/puma.sol#145)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-
calls
INFO:Detectors:
Parameter PumaToken.mint(address,uint256)._to (contracts/masterchef.sol#948) is not in
mixedCase
Parameter PumaToken.mint(address,uint256)._amount (contracts/masterchef.sol#948) is not
in mixedCase
Variable PumaToken._delegates (contracts/masterchef.sol#960) is not in mixedCase
Parameter MasterChef.add(uint256,IERC20,uint16,bool)._allocPoint (contracts/
masterchef.sol#1262) is not in mixedCase
Parameter MasterChef.add(uint256,IERC20,uint16,bool)._lpToken (contracts/
masterchef.sol#1262) is not in mixedCase
Parameter MasterChef.add(uint256,IERC20,uint16,bool)._depositFeeBP (contracts/
masterchef.sol#1262) is not in mixedCase
Parameter MasterChef.add(uint256,IERC20,uint16,bool)._withUpdate (contracts/
masterchef.sol#1262) is not in mixedCase
Parameter MasterChef.set(uint256,uint256,uint16,bool)._pid (contracts/
masterchef.sol#1279) is not in mixedCase
Parameter MasterChef.set(uint256,uint256,uint16,bool)._allocPoint (contracts/
masterchef.sol#1279) is not in mixedCase
Parameter MasterChef.set(uint256,uint256,uint16,bool)._depositFeeBP (contracts/
masterchef.sol#1279) is not in mixedCase
Parameter MasterChef.set(uint256,uint256,uint16,bool)._withUpdate (contracts/
masterchef.sol#1279) is not in mixedCase
```

```
Parameter MasterChef.getMultiplier(uint256,uint256)._from (contracts/
masterchef.sol#1290) is not in mixedCase
Parameter MasterChef.getMultiplier(uint256,uint256)._to (contracts/masterchef.sol#1290)
is not in mixedCase
Parameter MasterChef.pendingPuma(uint256,address)._pid (contracts/masterchef.sol#1295)
is not in mixedCase
Parameter MasterChef.pendingPuma(uint256,address)._user (contracts/masterchef.sol#1295)
is not in mixedCase
Parameter MasterChef.updatePool(uint256)._pid (contracts/masterchef.sol#1317) is not in
mixedCase
Parameter MasterChef.deposit(uint256,uint256)._pid (contracts/masterchef.sol#1336) is
not in mixedCase
Parameter MasterChef.deposit(uint256,uint256)._amount (contracts/masterchef.sol#1336)
is not in mixedCase
Parameter MasterChef.depositReferral(uint256,uint256,address)._pid (contracts/
masterchef.sol#1363) is not in mixedCase
Parameter MasterChef.depositReferral(uint256,uint256,address)._amount (contracts/
masterchef.sol#1363) is not in mixedCase
Parameter MasterChef.withdraw(uint256,uint256)._pid (contracts/masterchef.sol#1393) is
not in mixedCase
Parameter MasterChef.withdraw(uint256,uint256)._amount (contracts/masterchef.sol#1393)
is not in mixedCase
Parameter MasterChef.emergencyWithdraw(uint256)._pid (contracts/masterchef.sol#1411) is
not in mixedCase
Parameter MasterChef.safePumaTransfer(address,uint256)._to (contracts/
masterchef.sol#1422) is not in mixedCase
Parameter MasterChef.safePumaTransfer(address,uint256)._amount (contracts/
masterchef.sol#1422) is not in mixedCase
Parameter MasterChef.dev(address)._devaddr (contracts/masterchef.sol#1432) is not in
mixedCase
Parameter MasterChef.setFeeAddress1(address)._feeAddress1 (contracts/
masterchef.sol#1437) is not in mixedCase
Parameter MasterChef.updateEmissionRate(uint256)._pumaPerBlock (contracts/
masterchef.sol#1442) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-
solidity-naming-conventions
INFO:Detectors:
Variable MasterChef.depositReferral(uint256,uint256,address).feeAddress1Share
(contracts/masterchef.sol#1377) is too similar to
MasterChef.depositReferral(uint256,uint256,address).feeAddress2Share (contracts/
masterchef.sol#1378)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-
too-similar
INFO:Detectors:
MasterChef.puma (contracts/puma.sol#1225) should be immutable
MasterChef.startBlock (contracts/puma.sol#1242) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-
variables-that-could-be-declared-immutable
INFO:Slither:. analyzed (20 contracts with 88 detectors), 129 result(s) found
```

0x Guard