MythX

REPORT 60D5A843A876B70019CAD2A8

Created	Fri Jun 25 2021 09:56:19 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	60d58d6043f2c39d6f12de1d

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
65fc44b2-df17-44fc-8d31-50c487dcb3f5	contracts/MasterChef.sol	48

Analysis 65fc44b2-df17-44fc-8d31-50c487dcb3f5 MythX

Started	Fri Jun 25 2021 09:56:27 GMT+0000 (Coordinated Universal Time)
Finished	Fri Jun 25 2021 10:41:31 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Remythx
Main Source File	Contracts/MasterChef.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	34	14

ISSUES

MEDIUM Function could be marked as external.

SWC-000 The function definition of "renounceOwnership" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file
contracts/MasterChef.sol
Locations

```
252 * thereby removing any functionality that is only available to the owner.
253 */
254 function renounceOwnership() public virtual onlyOwner {
255 emit OwnershipTransferred(_owner, address(0));
256 _owner = address(0);
257 }
258
259 /**
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "transferOwnership" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/MasterChef.sol

Locations

```
261 * Can only be called by the current owner.
262 */
263 function transferOwnership(address newOwner) public virtual onlyOwner {
264 require(newOwner != address(0), "Ownable: new owner is the zero address");
265 emit OwnershipTransferred(_owner, newOwner);
266 _owner = newOwner;
267 }
268 }
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "symbol" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/MasterChef.sol

Locations

```
540 }
541
542 function symbol() public override view returns (string memory) {
543 return _symbol;
544 }
545
546 function decimals() public override view returns (uint8) {
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "decimals" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/MasterChef.sol

Locations

```
544 }
545
546 function decimals() public override view returns (uint8) {
547 return _decimals;
548 }
549
550 function totalSupply() public override view returns (uint256) {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transfer" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/MasterChef.sol

Locations

```
556 }
557
558 function transfer(address recipient, uint256 amount) public override returns (bool) {
559 _transfer(_msgSender(), recipient, amount);
560 return true;
561 }
562
563 function allowance(address owner, address spender) public override view returns (uint256) {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "approve" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/MasterChef.sol

Locations

```
565 }
566
567 function approve(address spender, uint256 amount) public override returns (bool) {
568 _approve(_msgSender(), spender, amount);
569 return true;
570 }
571
572 function transferFrom (address sender, address recipient, uint256 amount) public override returns (bool) {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "transferFrom" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/MasterChef.sol

Locations

```
570 }
571
572 function transferFrom (address sender, address recipient, uint256 amount) public override returns (bool) {
573 _transfer(sender, recipient, amount);
574 _approve(
575 sender,
576 _msgSender(),
577 _allowances[sender][_msgSender()].sub(amount, 'BEP20: transfer amount exceeds allowance')
578 );
579 return true;
580 }
581
582 function increaseAllowance(address spender, uint256 addedValue) public returns (bool) {
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "increaseAllowance" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/MasterChef.sol

Locations

```
580 }
581
582 function increaseAllowance(address spender, uint256 addedValue) public returns (bool) {
583 _approve(_msgSender(), spender, _allowances[_msgSender()][spender].add(addedValue));
584 return true;
585 }
586
587 function decreaseAllowance(address spender, uint256 subtractedValue) public returns (bool) {
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "decreaseAllowance" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/MasterChef.sol

Locations

```
585 }
586
587 function decreaseAllowance(address spender, uint256 subtractedValue) public returns (bool) {
588 _approve(_msgSender(), spender, _allowances[_msgSender()][spender].sub(subtractedValue, 'BEP20: decreased allowance below zero'));
589 return true;
590 }
591
592 function mint(uint256 amount) public onlyOwner returns (bool) {
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "mint" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/MasterChef.sol

Locations

```
590 }
591
592 function mint(uint256 amount) public onlyOwner returns (bool) {
593 _mint(_msgSender(), amount);
594 return true;
595 }
596
597 function _transfer (address sender, address recipient, uint256 amount) internal virtual {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "mint" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/MasterChef.sol

Locations

```
717
718 /// @notice Creates `_amount` token to `_to`. Must only be called by the owner (MasterChef).
719 function mint(address _to, uint256 _amount) public onlyOwner {
720 _mint(_to, _amount);
721 _moveDelegates(address(0), _delegates[_to], _amount);
722 }
723
724 /// @dev overrides transfer function to meet tokenomics
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "burn" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/MasterChef.sol

Locations

```
757
758 /// @dev Burn by reduce total supply
759 function burn(uint256 amount) public {
760 _burn(_msgSender(), amount);
761 }
762
763 /// @dev BurnFrom by reduce total supply
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "burnFrom" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/MasterChef.sol

Locations

```
762
763 /// @dev BurnFrom by reduce total supply
764 function burnFrom(address account, uint256 amount) public {
765 uint256 decreasedAllowance = allowance(account, _msgSender()).sub(amount, "ERC20: burn amount exceeds allowance");
766
767 _approve(account, _msgSender(), decreasedAllowance);
768 _burn(account, amount);
769 }
770
771 /// @dev Swap and liquify
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "isExcludedFromAntiWhale" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/MasterChef.sol

Locations

```
847 * @dev Returns the address is excluded from antiWhale or not.
848 */
849 function isExcludedFromAntiWhale(address _account) public view returns (bool) {
850 return _excludedFromAntiWhale[_account];
851 }
852
853 // To receive BNB from swapRouter when swapping
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "updateTransferTaxRate" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/MasterChef.sol

Locations

```
858 * Can only be called by the current operator.
859 */
860 function updateTransferTaxRate(uint16 _transferTaxRate) public onlyOperator {
861 require(_transferTaxRate <= MAXIMUM_TRANSFER_TAX_RATE, "ERROR::updateTransferTaxRate: Transfer tax rate must not exceed the maximum rate.");
862 emit TransferTaxRateUpdated(msg.sender, transferTaxRate, _transferTaxRate);
863 transferTaxRate = _transferTaxRate;
864 }
865
866 /**
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "updateBurnRate" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/MasterChef.sol

Locations

```
868 * Can only be called by the current operator.
869 */
870 function updateBurnRate(uint16 _burnRate) public onlyOperator {
871 require(_burnRate <= 100, "ERROR::updateBurnRate: Burn rate must not exceed the maximum rate.");
872 emit BurnRateUpdated(msg.sender, burnRate, _burnRate);
873 burnRate = _burnRate;
874 }
875
876 /**
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "updateMaxTransferAmountRate" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/MasterChef.sol

Locations

```
878 * Can only be called by the current operator.
879 */
880 function updateMaxTransferAmountRate(uint16 _maxTransferAmountRate) public onlyOperator {
881 require(_maxTransferAmountRate <= 10000, "ERROR::updateMaxTransferAmountRate: Max transfer amount rate must not exceed the maximum rate.");
882 emit MaxTransferAmountRateUpdated(msg.sender, maxTransferAmountRate, _maxTransferAmountRate);
883 maxTransferAmountRate = _maxTransferAmountRate;
884 }
885
886 /**
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "updateMinAmountToLiquify" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/MasterChef.sol

Locations

```
888 * Can only be called by the current operator.
889 */
890 function updateMinAmountToLiquify(uint256 _minAmount) public onlyOperator {
891 emit MinAmountToLiquifyUpdated(msg.sender, minAmountToLiquify, _minAmount);
892 minAmountToLiquify = _minAmount;
893 }
894
895 /**
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "setExcludedFromAntiWhale" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/MasterChef.sol

Locations

```
897 * Can only be called by the current operator.
898 */
899 function setExcludedFromAntiWhale(address _account, bool _excluded) public onlyOperator {
900 _excludedFromAntiWhale[_account] = _excluded;
901 }
902
903 /**
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "updateSwapAndLiquifyEnabled" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/MasterChef.sol

Locations

```
905 * Can only be called by the current operator.
906 */
907 function updateSwapAndLiquifyEnabled(bool _enabled) public onlyOperator {
908 emit SwapAndLiquifyEnabledUpdated(msg.sender, _enabled);
909 swapAndLiquifyEnabled = _enabled;
910 }
911
912 /**
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "updateSwapRouter" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/MasterChef.sol

Locations

```
914 * Can only be called by the current operator.
915 */
916 function updateSwapRouter(address _router) public onlyOperator {
917 swapRouter = IUniswapV2Router02(_router);
918 swapPair = IUniswapV2Factory(swapRouter.factory()).getPair(address(this), swapRouter.WETH());
919 require(swapPair != address(0), "ERROR::updateSwapRouter: Invalid pair address.");
920 emit SwapRouterUpdated(msg.sender, address(swapRouter), swapPair);
921 }
922
923 /**
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "transferOperator" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/MasterChef.sol

Locations

```
932 * Can only be called by the current operator.
933 */
934 function transferOperator(address newOperator) public onlyOperator {
935 require(newOperator != address(0), "ERROR::transferOperator: new operator is the zero address");
936 emit OperatorTransferred(_operator, newOperator);
937 _operator = newOperator;
938 }
939 // Copied and modified from YAM code:
940 // https://github.com/yam-finance/yam-protocol/blob/master/contracts/token/YAMGovernanceStorage.sol
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "add" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/MasterChef.sol

Locations

```
1321 // Add a new lp to the pool. Can only be called by the owner.
1322 // XXX DO NOT add the same LP token more than once. Rewards will be messed up if you do.
1323 function add(uint256 _allocPoint, IBEP20 _lpToken, uint16 _depositFeeBP, uint256 _harvestInterval, uint256 _minDeposit, bool _withUpdate)
1324 public onlyOwner {
1325 require(_depositFeeBP <= MAXIMUM_DEPOSIT_FEE, "add: invalid deposit fee basis points");
1326 require(_harvestInterval <= MAXIMUM_HARVEST_INTERVAL, "add: invalid harvest interval");
1327 require(_minDeposit <= MAXIMUM_MIN_DEPOSIT_FEE, "add: invalid Min Deposit amount");
1328 if (_withUpdate) {
1329 massUpdatePools();
1330 }
1331 uint256 lastRewardBlock = block.number > startBlock ? block.number : startBlock;
1332 totalAllocPoint = totalAllocPoint.add(_allocPoint);
1333 poolExistence[_lpToken] = true;
1334
1335 poolInfo.push(PoolInfo({
1336 lpToken: _lpToken,
1337 allocPoint: _allocPoint,
1338 lastRewardBlock: lastRewardBlock,
1339 accFoxPerShare: 0,
1340 depositFeeBP: _depositFeeBP,
1341 harvestInterval: _harvestInterval,
1342 minDeposit: _minDeposit,
1343 totalToken : 0
1344 }));
1345 }
1346
1347 // Update the given pool's token allocation point and deposit fee. Can only be called by the owner.
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "set" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/MasterChef.sol

Locations

```
1346
1347 // Update the given pool's token allocation point and deposit fee. Can only be called by the owner.
1348 function set(uint256 _pid, uint256 _allocPoint, uint16 _depositFeeBP, uint256 _harvestInterval, uint256 _minDeposit, bool _withUpdate)
1349 public onlyOwner {
1350 require(_depositFeeBP <= MAXIMUM_DEPOSIT_FEE, "set: invalid deposit fee basis points");
1351 require(_harvestInterval <= MAXIMUM_HARVEST_INTERVAL, "set: invalid harvest interval");
1352 require(_minDeposit <= MAXIMUM_MIN_DEPOSIT_FEE, "add: invalid Min Deposit amount");
1353 if (_withUpdate) {
1354 massUpdatePools();
1355 }
1356 totalAllocPoint = totalAllocPoint.sub(poolInfo[_pid].allocPoint).add(_allocPoint);
1357 poolInfo[_pid].allocPoint = _allocPoint;
1358 poolInfo[_pid].depositFeeBP = _depositFeeBP;
1359 poolInfo[_pid].harvestInterval = _harvestInterval;
1360 poolInfo[_pid].minDeposit = _minDeposit;
1361 }
1362
1363 // Return reward multiplier over the given _from to _to block.
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "deposit" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/MasterChef.sol

Locations

```
1441
1442 // Deposit LP tokens to MasterChef for token allocation.
1443 function deposit(uint256 _pid, uint256 _amount, address _referrer) public nonReentrant {
1444 PoolInfo storage pool = poolInfo[_pid];
1445 UserInfo storage user = userInfo[_pid][msg.sender];
1446
1447 require(fox.balanceOf(msg.sender) >= pool.minDeposit, "Not enough native tokens to use this pool!");
1448
1449 updatePool(_pid);
1450 if (_amount > 0 && address(foxReferral) != address(0) && _referrer != address(0) && _referrer != msg.sender) {
1451 foxReferral.recordReferral(msg.sender, _referrer);
1452 }
1453 payOrLockupPendingFox(_pid);
1454
1455 if(_amount > 0) {
1456 uint256 beforeDeposit = pool.lpToken.balanceOf(address(this));
1457 pool.lpToken.safeTransferFrom(address(msg.sender), address(this), _amount);
1458 uint256 afterDeposit = pool.lpToken.balanceOf(address(this));
1459
1460 // Calculate the correct amount when user deposit
1461 _amount = afterDeposit.sub(beforeDeposit);
1462
1463 if(pool.depositFeeBP > 0){
1464 uint256 depositFee = _amount.mul(pool.depositFeeBP).div(10000);
1465 pool.lpToken.safeTransfer(feeAddress, depositFee);
1466 user.amount = user.amount.add(_amount).sub(depositFee);
1467 pool.totalToken = pool.totalToken.add(_amount).sub(depositFee);
1468 }else{
1469 user.amount = user.amount.add(_amount);
1470 pool.totalToken = pool.totalToken.add(_amount);
1471 }
1472 }
1473 user.rewardDebt = user.amount.mul(pool.accFoxPerShare).div(1e12);
1474 emit Deposit(msg.sender, _pid, _amount);
1475 }
1476
1477 // Withdraw LP tokens from MasterChef.
```

MEDIUM Function could be marked as external.

SWC-000 The function definition of "withdraw" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/MasterChef.sol

Locations

```
1476
1477 // Withdraw LP tokens from MasterChef.
1478 function withdraw(uint256 _pid, uint256 _amount) public nonReentrant {
1479 PoolInfo storage pool = poolInfo[_pid];
1480 UserInfo storage user = userInfo[_pid][msg.sender];
1481
1482 require(fox.balanceOf(msg.sender) >= pool.minDeposit, "Not enough native tokens to use this pool!");
1483 require(pool.totalToken >= _amount, "Withdraw: Pool total tokens not enough");
1484 require(user.amount >= _amount, "withdraw: not good");
1485
1486 updatePool(_pid);
1487
1488 payOrLockupPendingFox(_pid);
1489 if(_amount > 0) {
1490 user.amount = user.amount.sub(_amount);
1491 pool.totalToken = pool.totalToken.sub(_amount);
1492 pool.lpToken.safeTransfer(address(msg.sender), _amount);
1493 }
1494 user.rewardDebt = user.amount.mul(pool.accFoxPerShare).div(1e12);
1495 emit Withdraw(msg.sender, _pid, _amount);
1496 }
1497
1498 // Withdraw without caring about rewards. EMERGENCY ONLY.
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "emergencyWithdraw" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/MasterChef.sol

Locations

```
1497
1498 // Withdraw without caring about rewards. EMERGENCY ONLY.
1499 function emergencyWithdraw(uint256 _pid) public nonReentrant {
1500 PoolInfo storage pool = poolInfo[_pid];
1501 UserInfo storage user = userInfo[_pid][msg.sender];
1502 uint256 amount = user.amount;
1503
1504 require(pool.totalToken >= amount, "EmergencyWithdraw: Pool total tokens not enough");
1505
1506 user.amount = 0;
1507 user.rewardDebt = 0;
1508 user.rewardLockedUp = 0;
1509 user.nextHarvestUntil = 0;
1510 pool.totalToken = pool.totalToken.sub(amount);
1511 pool.lpToken.safeTransfer(address(msg.sender), amount);
1512 emit EmergencyWithdraw(msg.sender, _pid, amount);
1513 }
1514
1515 // Pay or lockup pending FOX.
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "setDevAddress" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/MasterChef.sol

Locations

```
1554
1555 // Update dev address by the previous dev.
1556 function setDevAddress(address _devAddress) public {
1557 require(msg.sender == devAddress, "setDevAddress: FORBIDDEN");
1558 require(_devAddress != address(0), "setDevAddress: ZERO");
1559 devAddress = _devAddress;
1560 }
1561
1562 function setFeeAddress(address _feeAddress) public{
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "setFeeAddress" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/MasterChef.sol

Locations

```
1560 }
1561
1562 function setFeeAddress(address _feeAddress) public{
1563 require(msg.sender == feeAddress, "setFeeAddress: FORBIDDEN");
1564 require(_feeAddress != address(0), "setFeeAddress: ZERO");
1565 feeAddress = _feeAddress;
1566 }
1567
1568 //Pancake has to add hidden dummy pools inorder to alter the emission, here we make it simple and transparent to all.
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "updateEmissionRate" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/MasterChef.sol

Locations

```
1567
1568 //Pancake has to add hidden dummy pools inorder to alter the emission, here we make it simple and transparent to all.
1569 function updateEmissionRate(uint256 _foxPerBlock) public onlyOwner {
1570 require(_foxPerBlock <= MAX_TOKEN_PER_BLOCK, "FOX per block too high");
1571 massUpdatePools();
1572 emit EmissionRateUpdated(msg.sender, foxPerBlock, _foxPerBlock);
1573 foxPerBlock = _foxPerBlock;
1574 }
1575
1576 function updateAllocPoint(uint256 _pid, uint256 _allocPoint, bool _withUpdate) public onlyOwner {
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "updateAllocPoint" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/MasterChef.sol

Locations

```
1574 }
1575
1576 function updateAllocPoint(uint256 _pid, uint256 _allocPoint, bool _withUpdate) public onlyOwner {
1577 if (_withUpdate) {
1578 massUpdatePools();
1579 }
1580
1581 totalAllocPoint = totalAllocPoint.sub(poolInfo[_pid].allocPoint).add(_allocPoint);
1582 poolInfo[_pid].allocPoint = _allocPoint;
1583 }
1584
1585 function updateMinDeposit(uint256 _pid, uint256 _minDeposit, bool _withUpdate) public onlyOwner {
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "updateMinDeposit" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/MasterChef.sol

Locations

```
1583 }
1584
1585 function updateMinDeposit(uint256 _pid, uint256 _minDeposit, bool _withUpdate) public onlyOwner {
1586 if (_withUpdate) {
1587 massUpdatePools();
1588 }
1589
1590 poolInfo[_pid].minDeposit = _minDeposit;
1591 }
1592
1593 // Update the fox referral contract address by the owner
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "setFoxReferral" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/MasterChef.sol

Locations

```
1592
1593 // Update the fox referral contract address by the owner
1594 function setFoxReferral(IFoxReferral _foxReferral) public onlyOwner {
1595 foxReferral = _foxReferral;
1596 }
1597
1598 // Update referral commission rate by the owner
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "setReferralCommissionRate" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

contracts/MasterChef.sol

Locations

```
1597
1598 // Update referral commission rate by the owner
1599 function setReferralCommissionRate(uint16 _referralCommissionRate) public onlyOwner {
1600 require(_referralCommissionRate <= MAXIMUM_REFERRAL_COMMISSION_RATE, "setReferralCommissionRate: invalid referral commission rate basis points");
1601 referralCommissionRate = _referralCommissionRate;
1602 }
1603
1604 // Pay referral commission to the referrer who referred this user.
```

LOW

SWC-120

Potential use of "block.number" as source of randonmness.

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source file

contracts/MasterChef.sol

Locations

```
1071 returns (uint256)
1072 {
1073 require(blockNumber < block.number, "Error::getPriorVotes: not yet determined");
1074
1075 uint32 nCheckpoints = numCheckpoints[account];
```

LOW

SWC-120

Potential use of "block.number" as source of randonmness.

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source file

contracts/MasterChef.sol

Locations

```
1144 internal
1145 {
1146 uint32 blockNumber = safe32(block.number, "Error::_writeCheckpoint: block number exceeds 32 bits");
1147
1148 if (nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber) {
```

LOW

SWC-120

Potential use of "block.number" as source of randonmness.

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source file

contracts/MasterChef.sol

Locations

```
1329 massUpdatePools();
1330 }
1331 uint256 lastRewardBlock = block.number > startBlock ? block.number : startBlock;
1332 totalAllocPoint = totalAllocPoint.add(_allocPoint);
1333 poolExistence[_lpToken] = true;
```

LOW

SWC-120

Potential use of "block.number" as source of randonmness.

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source file

contracts/MasterChef.sol

Locations

```
1329 massUpdatePools();
1330 }
1331 uint256 lastRewardBlock = block.number > startBlock ? block.number : startBlock;
1332 totalAllocPoint = totalAllocPoint.add(_allocPoint);
1333 poolExistence[_lpToken] = true;
```

LOW

SWC-120

Potential use of "block.number" as source of randonmness.

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source file

contracts/MasterChef.sol

Locations

```
1373 uint256 lpSupply = pool.totalToken;
1374
1375 if (block.number > pool.lastRewardBlock && lpSupply != 0) {
1376 uint256 multiplier = getMultiplier(pool.lastRewardBlock, block.number);
1377 uint256 foxReward = multiplier.mul(foxPerBlock).mul(pool.allocPoint).div(totalAllocPoint);
```

LOW

SWC-120

Potential use of "block.number" as source of randonmness.

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source file

contracts/MasterChef.sol

Locations

```
1374
1375 if (block.number > pool.lastRewardBlock && lpSupply != 0) {
1376 uint256 multiplier = getMultiplier(pool.lastRewardBlock, block.number);
1377 uint256 foxReward = multiplier.mul(foxPerBlock).mul(pool.allocPoint).div(totalAllocPoint);
1378 accFoxPerShare = accFoxPerShare.add(foxReward.mul(1e12).div(lpSupply));
```

LOW

SWC-120

Potential use of "block.number" as source of randonmness.

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source file

contracts/MasterChef.sol

Locations

```
1396 for (uint256 pid = 0; pid < length; ++pid) {
1397 PoolInfo storage pool = poolInfo[pid];
1398 if (block.number <= pool.lastRewardBlock) {
1399 continue;
1400 }
```

LOW

SWC-120

Potential use of "block.number" as source of randonmness.

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source file

contracts/MasterChef.sol

Locations

```
1401
1402 if (pool.totalToken == 0 || pool.allocPoint == 0) {
1403 pool.lastRewardBlock = block.number;
1404 continue;
1405 }
```

LOW

SWC-120

Potential use of "block.number" as source of randonmness.

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source file

contracts/MasterChef.sol

Locations

```
1405 }
1406
1407 uint256 multiplier = getMultiplier(pool.lastRewardBlock, block.number);
1408 uint256 foxReward = multiplier.mul(foxPerBlock).mul(pool.allocPoint).div(totalAllocPoint);
```

LOW

SWC-120

Potential use of "block.number" as source of randonmness.

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source file

contracts/MasterChef.sol

Locations

```
1409
1410 pool.accFoxPerShare = pool.accFoxPerShare.add(foxReward.mul(1e12).div(pool.totalToken));
1411 pool.lastRewardBlock = block.number;
1412 totalReward.add(foxReward);
1413 }
```

LOW

SWC-120

Potential use of "block.number" as source of randonmness.

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source file

contracts/MasterChef.sol

Locations

```
1422 function updatePool(uint256 _pid) public {
1423 PoolInfo storage pool = poolInfo[_pid];
1424 if (block.number <= pool.lastRewardBlock) {
1425 return;
1426 }
```

LOW

SWC-120

Potential use of "block.number" as source of randonmness.

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source file

contracts/MasterChef.sol

Locations

```
1427
1428 if (pool.totalToken == 0 || pool.allocPoint == 0) {
1429 pool.lastRewardBlock = block.number;
1430 return;
1431 }
```

LOW

SWC-120

Potential use of "block.number" as source of randonmness.

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source file

contracts/MasterChef.sol

Locations

```
1430 return;
1431 }
1432 uint256 multiplier = getMultiplier(pool.lastRewardBlock, block.number);
1433 uint256 foxReward = multiplier.mul(foxPerBlock).mul(pool.allocPoint).div(totalAllocPoint);
```

LOW

SWC-120

Potential use of "block.number" as source of randonmness.

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source file

contracts/MasterChef.sol

Locations

```
1434
1435  pool.accFoxPerShare = pool.accFoxPerShare.add(foxReward.mul(1e12).div(pool.totalToken));
1436  pool.lastRewardBlock = block.number;
1437
1438  fox.mint(devAddress, foxReward.div(10));
```