MythX

REPORT 609D7D5C706B4A0018FB7343

Created	Thu May 13 2021 19:26:20 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	609d7b6f8bfa12ed16f28fb0

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
5285387b-311b-4c70-a81b-248db680243b	/contracts/masterchef.sol	19

Analysis 5285387b-311b-4c70-a81b-248db680243b

MythX

Started	Thu May 13 2021 19:26:30 GMT+0000 (Coordinated Universal Time)
Finished	Thu May 13 2021 19:28:45 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Vscode-Extension
Main Source File	/Contracts/Masterchef.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	11	8

ISSUES

MEDIUM — SWC-000

Function could be marked as external.

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

```
function add(uint256 _allocPoint, IBEP20 _lpToken, uint16 _depositFeeBP, uint256 _harvestInterval, bool _withUpdate) public onlyOwner {
require(_depositFeeBP <= 10000, "add: invalid deposit fee basis points");
require(_harvestInterval <= MAXIMUM_HARVEST_INTERVAL, "add: invalid harvest interval");
if (_withUpdate) {
massUpdatePools();
}
uint256 lastRewardBlock = block.number > startBlock ? block.number : startBlock;
totalAllocPoint = totalAllocPoint.add(_allocPoint);
poolInfo.push(PoolInfo({
lpToken: _lpToken,
allocPoint: _allocPoint,
lastRewardBlock: lastRewardBlock,
accCougarPerShare: 0,
depositFeeBP: _depositFeeBP,
harvestInterval: _harvestInterval
}));
}

// Update the given pool's COUGAR allocation point and deposit fee. Can only be called by the owner.
function set(uint256 _pid, uint256 _allocPoint, uint16 _depositFeeBP, uint256 _harvestInterval, bool _withUpdate) public onlyOwner {
require(_depositFeeBP <= 10000, "set: invalid deposit fee basis points");
require(_harvestInterval <= MAXIMUM_HARVEST_INTERVAL, "set: invalid harvest interval");
if (_withUpdate) {
```

MEDIUM

SWC-000

Function could be marked as external.

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

```
function set(uint256 _pid, uint256 _allocPoint, uint16 _depositFeeBP, uint256 _harvestInterval, bool _withUpdate) public onlyOwner {
require(_depositFeeBP <= 10000, "set: invalid deposit fee basis points");
require(_harvestInterval <= MAXIMUM_HARVEST_INTERVAL, "set: invalid harvest interval");
if (_withUpdate) {
massUpdatePools();
}
totalAllocPoint = totalAllocPoint.sub(poolInfo[_pid].allocPoint).add(_allocPoint);
poolInfo[_pid].allocPoint = _allocPoint;
poolInfo[_pid].depositFeeBP = _depositFeeBP;
poolInfo[_pid].harvestInterval = _harvestInterval;
}

// Return reward multiplier over the given _from to _to block.
function getMultiplier(uint256 _from, uint256 _to) public pure returns (uint256) {
return _to.sub(_from).mul(BONUS_MULTIPLIER);
}

// View function to see pending COUGARs on frontend.
function pendingCougar(uint256 _pid, address _user) external view returns (uint256) {
PoolInfo storage pool = poolInfo[_pid];
```

MEDIUM

SWC-000

Function could be marked as external.

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

```
UserInfo storage user = userInfo[_pid][msg.sender];
updatePool(_pid);
if (_amount > 0 && address(cougarReferral) != address(0) && _referrer != address(0) && _referrer != msg.sender) {
cougarReferral.recordReferral(msg.sender, _referrer);
}
payOrLockupPendingCougar(_pid);
if (_amount > 0) {
pool.lpToken.safeTransferFrom(address(msg.sender), address(this), _amount);
if (address(pool.lpToken) == address(cougar)) {
uint256 transferTax = _amount.mul(cougar.transferTaxRate()).div(10000);
_amount = _amount.sub(transferTax);
}
if (pool.depositFeeBP > 0) {
uint256 depositFee = _amount.mul(pool.depositFeeBP).div(10000);
pool.lpToken.safeTransfer(feeAddress, depositFee);
user.amount = user.amount.add(_amount).sub(depositFee);
} else {
user.amount = user.amount.add(_amount);
}
}
user.rewardDebt = user.amount.mul(pool.accCougarPerShare).div(1e12);
emit Deposit(msg.sender, _pid, _amount);
}

// Withdraw LP tokens from MasterChef.
function withdraw(uint256 _pid, uint256 _amount) public nonReentrant {
PoolInfo storage pool = poolInfo[_pid];
UserInfo storage user = userInfo[_pid][msg.sender];
require(user.amount >= _amount, "withdraw: not good");
updatePool(_pid);
payOrLockupPendingCougar(_pid);
```

MEDIUM

SWC-000

Function could be marked as external.

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

```
UserInfo storage user = userInfo[_pid][msg.sender];
require(user.amount >= _amount, "withdraw: not good");
updatePool(_pid);
payOrLockupPendingCougar(_pid);
if (_amount > 0) {
user.amount = user.amount.sub(_amount);
pool.lpToken.safeTransfer(address(msg.sender), _amount);
}
user.rewardDebt = user.amount.mul(pool.accCougarPerShare).div(1e12);
emit Withdraw(msg.sender, _pid, _amount);
}

// Withdraw without caring about rewards. EMERGENCY ONLY.
function emergencyWithdraw(uint256 _pid) public nonReentrant {
PoolInfo storage pool = poolInfo[_pid];
UserInfo storage user = userInfo[_pid][msg.sender];
uint256 amount = user.amount;
user.amount = 0;
user.rewardDebt = 0;
```

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

```
uint256 amount = user.amount;
user.amount = 0;
user.rewardDebt = 0;
user.rewardLockedUp = 0;
user.nextHarvestUntil = 0;
pool.lpToken.safeTransfer(address(msg.sender), amount);
emit EmergencyWithdraw(msg.sender, _pid, amount);
}

// Pay or lockup pending COUGARs.
function payOrLockupPendingCougar(uint256 _pid) internal {
PoolInfo storage pool = poolInfo[_pid];
UserInfo storage user = userInfo[_pid][msg.sender];

if (user.nextHarvestUntil == 0) {
user.nextHarvestUntil = block.timestamp.add(pool.harvestInterval);
}
```

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

```
}

function setFeeAddress(address _feeAddress) public {
require(msg.sender == feeAddress, "setFeeAddress: FORBIDDEN");
require(_feeAddress != address(0), "setFeeAddress: ZERO");
feeAddress = _feeAddress;
}

// Pancake has to add hidden dummy pools in order to alter the emission, here we make it simple and transparent to all.
function updateEmissionRate(uint256 _cougarPerBlock) public onlyOwner {
massUpdatePools();
```

MEDIUM

SWC-000

Function could be marked as external.

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

```
}

// Pancake has to add hidden dummy pools in order to alter the emission, here we make it simple and transparent to all.
function updateEmissionRate(uint256 _cougarPerBlock) public onlyOwner {
massUpdatePools();
emit EmissionRateUpdated(msg.sender, cougarPerBlock, _cougarPerBlock);
cougarPerBlock = _cougarPerBlock;
}
```

MEDIUM

SWC-000

Function could be marked as external.

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

```
}

// Update the cougar referral contract address by the owner
function setCougarReferral(ICougarReferral _cougarReferral) public onlyOwner {
cougarReferral = _cougarReferral;
}

// Update referral commission rate by the owner
function setReferralCommissionRate(uint16 _referralCommissionRate) public onlyOwner {
require(_referralCommissionRate <= MAXIMUM_REFERRAL_COMMISSION_RATE, "setReferralCommissionRate: invalid referral commission rate basis points");
referralCommissionRate = _referralCommissionRate;
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "setCougarReferral" 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

```

// Update referral commission rate by the owner
function setReferralCommissionRate(uint16 _referralCommissionRate) public onlyOwner {
require(_referralCommissionRate <= MAXIMUM_REFERRAL_COMMISSION_RATE, "setReferralCommissionRate: invalid referral commission rate basis points");
referralCommissionRate = _referralCommissionRate;
}
```

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

```
function setReferralCommissionRate(uint16 _referralCommissionRate) public onlyOwner {
require(_referralCommissionRate <= MAXIMUM_REFERRAL_COMMISSION_RATE, "setReferralCommissionRate: invalid referral commission rate basis points");
referralCommissionRate = _referralCommissionRate;
}

// Pay referral commission to the referrer who referred this user.
function payReferralCommission(address _user, uint256 _pending) internal {
if (address(cougarReferral) != address(0) && referralCommissionRate > 0) {
address referrer = cougarReferral.getReferrer(_user);
uint256 commissionAmount = _pending.mul(referralCommissionRate).div(10000);
```

MEDIUM

SWC-128

Loop over unbounded data structure.

Gas consumption in function "massUpdatePools" in contract "MasterChef" depends on the size of data structures or values that may grow unboundedly. If the data structure grows too large, the gas required to execute the code will exceed the block gas limit, effectively causing a denial-of-service condition. Consider that an attacker might attempt to cause this condition on purpose.

Source file

`/contracts/masterchef.sol`

Locations

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

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

```
lpToken: _lpToken,
allocPoint: _allocPoint,
lastRewardBlock: lastRewardBlock,
accCougarPerShare: 0,
depositFeeBP: _depositFeeBP,
harvestInterval: _harvestInterval
```

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

```
allocPoint: _allocPoint,
lastRewardBlock: lastRewardBlock,
accCougarPerShare: 0,
depositFeeBP: _depositFeeBP,
harvestInterval: _harvestInterval
```

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

```
uint256 multiplier = getMultiplier(pool.lastRewardBlock, block.number);
uint256 cougarReward = multiplier.mul(cougarPerBlock).mul(pool.allocPoint).div(totalAllocPoint);
accCougarPerShare = accCougarPerShare.add(cougarReward.mul(1e12).div(lpSupply));
}
uint256 pending = user.amount.mul(accCougarPerShare).div(1e12).sub(user.rewardDebt);
```

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

```
accCougarPerShare = accCougarPerShare.add(cougarReward.mul(1e12).div(lpSupply));
}
uint256 pending = user.amount.mul(accCougarPerShare).div(1e12).sub(user.rewardDebt);
return pending.add(user.rewardLockedUp);
}
```

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

```
if (lpSupply == 0 || pool.allocPoint == 0) {
pool.lastRewardBlock = block.number;
return;
}
uint256 multiplier = getMultiplier(pool.lastRewardBlock, block.number);
uint256 cougarReward = multiplier.mul(cougarPerBlock).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

```
uint256 multiplier = getMultiplier(pool.lastRewardBlock, block.number);
uint256 cougarReward = multiplier.mul(cougarPerBlock).mul(pool.allocPoint).div(totalAllocPoint);
cougar.mint(devAddress, cougarReward.div(10));
cougar.mint(address(this), cougarReward);
pool.accCougarPerShare = pool.accCougarPerShare.add(cougarReward.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

```
cougar.mint(devAddress, cougarReward.div(10));
cougar.mint(address(this), cougarReward);
pool.accCougarPerShare = pool.accCougarPerShare.add(cougarReward.mul(1e12).div(lpSupply));
pool.lastRewardBlock = block.number;
}
```

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

```
function deposit(uint256 _pid, uint256 _amount, address _referrer) public nonReentrant {
PoolInfo storage pool = poolInfo[_pid];
UserInfo storage user = userInfo[_pid][msg.sender];
updatePool(_pid);
if (_amount > 0 && address(cougarReferral) != address(0) && _referrer != address(0) && _referrer != msg.sender) {
```