MythX

Analysis 2d8365f8-d7ad-45b2-b7de-701d9c464fc6 MythX

Started	Fri Apr 02 2021 03:07:32 GMT+0000 (Coordinated Universal Time)
Finished	Fri Apr 02 2021 03:52:58 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Remythx
Main Source File	`MasterChef.Sol`

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	26	25

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

`SafeMath.sol`

Locations

```
require(b > 0, errorMessage);
uint256 c = a / b;
// assert(a == b * c + a % b); // There is no case in which this doesn't hold

return c;
}

/**
* @dev Returns the remainder of dividing two unsigned integers. (unsigned integer modulo),
* Reverts when dividing by zero.
*
* Counterpart to Solidity's `%` operator. This function uses a `revert`
* opcode (which leaves remaining gas untouched) while Solidity uses an
* invalid opcode to revert (consuming all remaining gas).
*
* Requirements:
*
* - The divisor cannot be zero.
*/
function mod(uint256 a, uint256 b) internal pure returns (uint256) {
return mod(a, b, 'SafeMath: modulo by zero');
}

```

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

`SafeMath.sol`

Locations

```

/**
* @dev Returns the remainder of dividing two unsigned integers. (unsigned integer modulo),
* Reverts with custom message when dividing by zero.
*
* Counterpart to Solidity's `%` operator. This function uses a `revert`
* opcode (which leaves remaining gas untouched) while Solidity uses an
* invalid opcode to revert (consuming all remaining gas).
*
* Requirements:
*
* - The divisor cannot be zero.
*/
function mod(
uint256 a,
uint256 b,
string memory errorMessage
) internal pure returns (uint256) {
require(b != 0, errorMessage);
```

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

`IBEP20.sol`

Locations

```
* @dev Returns the token symbol.
*/
function symbol() external view returns (string memory);

/**
* @dev Returns the token name.
*/
function name() external view returns (string memory);

/**
```

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

`IBEP20.sol`

Locations

```
function name() external view returns (string memory);

/**
* @dev Returns the bep token owner.
*/
// function getOwner() external view returns (address);

/**
* @dev Returns the amount of tokens owned by `account`.
*/
function balanceOf(address account) external 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

`IBEP20.sol`

Locations

```

/**
* @dev Returns the amount of tokens owned by `account`.
*/
function balanceOf(address account) external view returns (uint256);

/**
* @dev Moves `amount` tokens from the caller's account to `recipient`.
*
* Returns a boolean value indicating whether the operation succeeded.
*
* Emits a {Transfer} event.
*/
function transfer(address recipient, uint256 amount) external returns (bool);

/**
/**
* @dev Returns the remaining number of tokens that `spender` will be
* allowed to spend on behalf of `owner` through {transferFrom}. This is
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "owner" 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

Address.sol

Locations

```
// According to EIP-1052, 0x0 is the value returned for not-yet created accounts
// and 0xc5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d85a470 is returned
// for accounts without code, i.e. `keccak256('')`
bytes32 codehash;
bytes32 accountHash = 0xc5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d85a470;
// solhint-disable-next-line no-inline-assembly
assembly {
```

MEDIUM

SWC-000

Function could be marked as external.

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

Address.sol

Locations

```
*
* https://eips.ethereum.org/EIPS/eip-1884[EIP1884] increases the gas cost
* of certain opcodes, possibly making contracts go over the 2300 gas limit
* imposed by `transfer`, making them unable to receive funds via
* `transfer`. {sendValue} removes this limitation.
*
* https://diligence.consensys.net/posts/2019/09/stop-using-soliditys-transfer-now/[Learn more].
*
* IMPORTANT: because control is transferred to `recipient`, care must be
```

MEDIUM

SWC-000

Function could be marked as external.

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

Address.sol

Locations

```
* https://diligence.consensys.net/posts/2019/09/stop-using-soliditys-transfer-now/[Learn more].
*
* IMPORTANT: because control is transferred to `recipient`, care must be
* taken to not create reentrancy vulnerabilities. Consider using
* {ReentrancyGuard} or the
* https://solidity.readthedocs.io/en/v0.5.11/security-considerations.html#use-the-checks-effects-interactions-pattern[checks-effects-interactions pattern].
*/
```

MEDIUM

SWC-000

Function could be marked as external.

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

MYSTERYToken.sol

Locations

```

/// @notice A record of states for signing / validating signatures
mapping (address => uint) public nonces;

/// @notice An event thats emitted when an account changes its delegate
event DelegateChanged(address indexed delegator, address indexed fromDelegate, address indexed toDelegate);
```

MEDIUM

SWC-000

Function could be marked as external.

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

MYSTERYToken.sol

Locations

```

/// @notice An event thats emitted when an account changes its delegate
event DelegateChanged(address indexed delegator, address indexed fromDelegate, address indexed toDelegate);

/// @notice An event thats emitted when a delegate account's vote balance changes
event DelegateVotesChanged(address indexed delegate, uint previousBalance, uint newBalance);
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "allowance" 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

MYSTERYToken.sol

Locations

```

/**
* @notice Delegates votes from signatory to `delegatee`
* @param delegatee The address to delegate votes to
* @param nonce The contract state required to match the signature
* @param expiry The time at which to expire the signature
* @param v The recovery byte of the signature
* @param r Half of the ECDSA signature pair
```

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

MYSTERYToken.sol

Locations

```
* @param expiry The time at which to expire the signature
* @param v The recovery byte of the signature
* @param r Half of the ECDSA signature pair
* @param s Half of the ECDSA signature pair
*/
function delegateBySig(
address delegatee,
uint nonce,
uint expiry,
uint8 v,
bytes32 r,
```

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

MYSTERYToken.sol

Locations

```
DELEGATION_TYPEHASH,
delegatee,
nonce,
expiry
)
);

bytes32 digest = keccak256(
abi.encodePacked(
"\x19\x01",
domainSeparator,
structHash
)
);

address signatory = ecrecover(digest, v, r, s);
require(signatory != address(0), "MYSTERY::delegateBySig: invalid signature");
require(nonce == nonces[signatory]++, "MYSTERY::delegateBySig: invalid nonce");
require(now <= expiry, "MYSTERY::delegateBySig: signature expired");
return _delegate(signatory, delegatee);
```

MEDIUM

SWC-000

Function could be marked as external.

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

MYSTERYToken.sol

Locations

```
* @notice Gets the current votes balance for `account`
* @param account The address to get votes balance
* @return The number of current votes for `account`
*/
function getCurrentVotes(address account)
external
view
returns (uint256)
{
uint32 nCheckpoints = numCheckpoints[account];
return nCheckpoints > 0 ? checkpoints[account][nCheckpoints - 1].votes : 0;
}
```

MEDIUM

SWC-000

Function could be marked as external.

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

MYSTERYToken.sol

Locations

```
* @return The number of votes the account had as of the given block
*/
function getPriorVotes(address account, uint blockNumber)
external
view
returns (uint256)
{
require(blockNumber < block.number, "MYSTERY::getPriorVotes: not yet determined");

uint32 nCheckpoints = numCheckpoints[account];
if (nCheckpoints == 0) {
return 0;
}
```

MEDIUM

SWC-113

Multiple calls are executed in the same transaction.

This call is executed following another call within the same transaction. It is possible that the call never gets executed if a prior call fails permanently. This might be caused intentionally by a malicious callee. If possible, refactor the code such that each transaction only executes one external call or make sure that all callees can be trusted (i.e. they're part of your own codebase).

Source file

`MasterChef.sol`

Locations

```
uint256 multiplier = getMultiplier(pool.lastRewardBlock, block.number);
uint256 MYSTERYReward = multiplier.mul(MYSTERYPerBlock).mul(pool.allocPoint).div(totalAllocPoint);
accMYSTERYPerShare = accMYSTERYPerShare.add(MYSTERYReward.mul(1e12).div(lpSupply));
}
return user.amount.mul(accMYSTERYPerShare).div(1e12).sub(user.rewardDebt);
}
```

LOW

SWC-120

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

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

`SafeMath.sol`

Locations

```
* Reverts when dividing by zero.
*
* Counterpart to Solidity's `%` operator. This function uses a `revert`
* opcode (which leaves remaining gas untouched) while Solidity uses an
* invalid opcode to revert (consuming all remaining gas).
*
```

LOW

SWC-120

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

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

`SafeMath.sol`

Locations

```
*
* Counterpart to Solidity's `%` operator. This function uses a `revert`
* opcode (which leaves remaining gas untouched) while Solidity uses an
* invalid opcode to revert (consuming all remaining gas).
*
```

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

`SafeMath.sol`

Locations

```
x = (y / x + x) / 2;
}
} else if (y != 0) {
z = 1;
}
```

LOW

SWC-123

Requirement violation.

A requirement was violated in a nested call and the call was reverted as a result. Make sure valid inputs are provided to the nested call (for instance, via passed arguments).

Source file

`MasterChef.sol`

Locations

```
uint256 multiplier = getMultiplier(pool.lastRewardBlock, block.number);
uint256 MYSTERYReward = multiplier.mul(MYSTERYPerBlock).mul(pool.allocPoint).div(totalAllocPoint);
accMYSTERYPerShare = accMYSTERYPerShare.add(MYSTERYReward.mul(1e12).div(lpSupply));
}
return user.amount.mul(accMYSTERYPerShare).div(1e12).sub(user.rewardDebt);
}
```