MythX

REPORT 6050350B04997600189F2942

Created	Tue Mar 16 2021 04:33:15 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	admin@pineappledefi.com

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
c3fda7ad-ceae-4f8e-a243-a25f65983655	browser/contracts/PineappleToken.sol	24

Analysis c3fda7ad-ceae-4f8e-a243-a25f65983655 MythX

Started	Tue Mar 16 2021 04:33:19 GMT+0000 (Coordinated Universal Time)
Finished	Tue Mar 16 2021 04:35:41 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Remythx
Main Source File	Browser/Contracts/PineappleToken.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	13	11

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

browser/contracts/PineappleToken.sol

Locations

```
78  * thereby removing any functionality that is only available to the owner.
79  */
80  function renounceOwnership() public virtual onlyOwner {
81  emit OwnershipTransferred(_owner, address(0));
82  _owner = address(0);
83  }
84
85  /**
```

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

browser/contracts/PineappleToken.sol

Locations

```
87  * Can only be called by the current owner.
88  */
89  function transferOwnership(address newOwner) public virtual onlyOwner {
90  require(newOwner != address(0), "Ownable: new owner is the zero address");
91  emit OwnershipTransferred(_owner, newOwner);
92  _owner = newOwner;
93  }
94  }
```

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

browser/contracts/PineappleToken.sol

Locations

```
427  * name.
428  */
429  function symbol() public override view returns (string memory) {
430  return _symbol;
431  }
432
433  /**
```

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

browser/contracts/PineappleToken.sol

Locations

```
434  * @dev Returns the number of decimals used to get its user representation.
435  */
436  function decimals() public override view returns (uint8) {
437  return _decimals;
438  }
439
440  /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "totalSupply" 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

browser/contracts/PineappleToken.sol

Locations

```
441 * @dev See {BEP20-totalSupply}.
442 */
443 function totalSupply() public override view returns (uint256) {
444 return _totalSupply;
445 }
446
447 /**
```

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

browser/contracts/PineappleToken.sol

Locations

```
460 * - the caller must have a balance of at least `amount`.
461 */
462 function transfer(address recipient, uint256 amount) public override returns (bool) {
463 _transfer(_msgSender(), recipient, amount);
464 return true;
465 }
466
467 /**
```

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

browser/contracts/PineappleToken.sol

Locations

```
468 * @dev See {BEP20-allowance}.
469 */
470 function allowance(address owner, address spender) public override view returns (uint256) {
471 return _allowances[owner][spender];
472 }
473
474 /**
```

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

browser/contracts/PineappleToken.sol

Locations

```
479 * - `spender` cannot be the zero address.
480 */
481 function approve(address spender, uint256 amount) public override returns (bool) {
482 _approve(_msgSender(), spender, amount);
483 return true;
484 }
485
486 /**
```

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

browser/contracts/PineappleToken.sol

Locations

```
496 * `amount`.
497 */
498 function transferFrom (address sender, address recipient, uint256 amount) public override returns (bool) {
499 _transfer(sender, recipient, amount);
500 _approve(
501 sender,
502 _msgSender(),
503 _allowances[sender][_msgSender()].sub(amount, 'BEP20: transfer amount exceeds allowance')
504 );
505 return true;
506 }
507
508 /**
```

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

browser/contracts/PineappleToken.sol

Locations

```
518 * - `spender` cannot be the zero address.
519 */
520 function increaseAllowance(address spender, uint256 addedValue) public returns (bool) {
521 _approve(_msgSender(), spender, _allowances[_msgSender()][spender].add(addedValue));
522 return true;
523 }
524
525 /**
```

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

browser/contracts/PineappleToken.sol

Locations

```
537 * `subtractedValue`.
538 */
539 function decreaseAllowance(address spender, uint256 subtractedValue) public returns (bool) {
540 _approve(_msgSender(), spender, _allowances[_msgSender()][spender].sub(subtractedValue, 'BEP20: decreased allowance below zero'));
541 return true;
542 }
543
544 /**
```

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

browser/contracts/PineappleToken.sol

Locations

```
550 * - `msg.sender` must be the token owner
551 */
552 function mint(uint256 amount) public onlyOwner returns (bool) {
553 _mint(_msgSender(), amount);
554 return true;
555 }
556
557 /**
```

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

browser/contracts/PineappleToken.sol

Locations

```
655 contract PineappleToken is BEP20('Pineapple', 'PIN') {
656 /// @notice Creates `_amount` token to `_to`. Must only be called by the owner (MasterChef).
657 function mint(address _to, uint256 _amount) public onlyOwner {
658 _mint(_to, _amount);
659 _moveDelegates(address(0), _delegates[_to], _amount);
660 }
661
662 // Copied and modified from YAM code:
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""=0.6.0<0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

browser/contracts/PineappleToken.sol

Locations

```
3   // SPDX-License-Identifier: MIT
4
5   pragma solidity >=0.6.0 <0.8.0;
6
7   /*
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""=0.6.0<0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

browser/contracts/PineappleToken.sol

Locations

```
28  // File: @openzeppelin/contracts/access/Ownable.sol
29
30  pragma solidity >=0.6.0 <0.8.0;
31
32  /**
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""=0.6.4"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

browser/contracts/PineappleToken.sol

Locations

```
96  // File: contracts/libs/IBEP20.sol
97
98  pragma solidity >=0.6.4;
99
100 interface IBEP20 {
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""=0.6.0<0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

browser/contracts/PineappleToken.sol

Locations

```
191 // File: @openzeppelin/contracts/math/SafeMath.sol
192
193 pragma solidity >=0.6.0 <0.8.0;
194
195 /**
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""=0.4.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

browser/contracts/PineappleToken.sol

Locations

```
351 // File: contracts/libs/BEP20.sol
352
353 pragma solidity >=0.4.0;
354
```

LOW

SWC-116

A control flow decision is made based on The block.timestamp environment variable.

The block.timestamp environment variable is used to determine a control flow decision. 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

browser/contracts/PineappleToken.sol

Locations

```
764 require(signatory != address(0), "PIN::delegateBySig: invalid signature");
765 require(nonce == nonces[signatory]++, "PIN::delegateBySig: invalid nonce");
766 require(now <= expiry, "PIN::delegateBySig: signature expired");
767 return _delegate(signatory, delegatee);
768 }
```

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

browser/contracts/PineappleToken.sol

Locations

```
794 returns (uint256)
795 {
796 require(blockNumber < block.number, "PIN::getPriorVotes: not yet determined");
797
798 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

browser/contracts/PineappleToken.sol

Locations

```
867 internal
868 {
869 uint32 blockNumber = safe32(block.number, "PIN::_writeCheckpoint: block number exceeds 32 bits");
870
871 if (nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber) {
```

LOW

SWC-120

A control flow decision is made based on The block.number environment variable.

The block.number environment variable is used to determine a control flow decision. 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

browser/contracts/PineappleToken.sol

Locations

```
794 returns (uint256)
795 {
796 require(blockNumber < block.number, "PIN::getPriorVotes: not yet determined");
797
798 uint32 nCheckpoints = numCheckpoints[account];
```

LOW

SWC-128

Potentially unbounded data structure passed to builtin.

Gas consumption in function "delegateBySig" in contract "PineappleToken" depends on the size of data structures that may grow unboundedly. Specifically the "1-st" argument to builtin "keccak256" may be able to grow unboundedly causing the builtin to consume more gas than 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

browser/contracts/PineappleToken.sol

Locations

```
738 abi.encode(
739 DOMAIN_TYPEHASH,
740 keccak256(bytes(name())),
741 getChainId(),
742 address(this)
```

LOW

SWC-128

Loop over unbounded data structure.

Gas consumption in function "getPriorVotes" in contract "PineappleToken" 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

browser/contracts/PineappleToken.sol

Locations

```
813 uint32 lower = 0;
814 uint32 upper = nCheckpoints - 1;
815 while (upper > lower) {
816 uint32 center = upper - (upper - lower) / 2; // ceil, avoiding overflow
817 Checkpoint memory cp = checkpoints[account][center];
```