MythX

REPORT 60668A8AB857B00011F186FA

Created	Fri Apr 02 2021 03:07:54 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	mysteryfarm063@gmail.com

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
0911f61e-f770-4816-b2b2-0fa0361db315	Presale.sol	27

Analysis `0911f61e-f770-4816-b2b2-0fa0361db315` MythX

Started	Fri Apr 02 2021 03:08:02 GMT+0000 (Coordinated Universal Time)
Finished	Fri Apr 02 2021 03:53:03 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Remythx
Main Source File	`Presale.Sol`

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	16	11

ISSUES

MEDIUM **Function could be marked as external.**

SWC-000

The function definition of "name" 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
`Presale.sol`
Locations

```
387 * @dev Returns the name of the token.
388 */
389 function name() public view returns (string memory) {
390 return _name;
391 }
392
393 /**
```

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
`Presale.sol`
Locations

```
395 * name.
396 */
397 function symbol() public view returns (string memory) {
398 return _symbol;
399 }
400
401 /**
```

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

Presale.sol

Locations

```
412 * {IBEP20-balanceOf} and {IBEP20-transfer}.
413 */
414 function decimals() public view returns (uint8) {
415 return _decimals;
416 }
417
418 /**
```

MEDIUM

Function could be marked as external.

SWC-000

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

Presale.sol

Locations

```
419 * @dev See {IBEP20-totalSupply}.
420 */
421 function totalSupply() public view override returns (uint256) {
422 return _totalSupply;
423 }
424
425 /**
```

MEDIUM

Function could be marked as external.

SWC-000

The function definition of "balanceOf" 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

Presale.sol

Locations

```
426 * @dev See {IBEP20-balanceOf}.
427 */
428 function balanceOf(address account) public view override returns (uint256) {
429 return _balances[account];
430 }
431
432 /**
```

MEDIUM

Function could be marked as external.

SWC-000

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

Presale.sol

Locations

```
438 * - the caller must have a balance of at least `amount`.
439 */
440 function transfer(address recipient, uint256 amount) public virtual override returns (bool) {
441 _transfer(_msgSender(), recipient, amount);
442 return true;
443 }
444
445 /**
```

MEDIUM

Function could be marked as external.

SWC-000

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

Presale.sol

Locations

```
446 * @dev See {IBEP20-allowance}.
447 */
448 function allowance(address owner, address spender) public view virtual override returns (uint256) {
449 return _allowances[owner][spender];
450 }
451
452 /**
```

MEDIUM

Function could be marked as external.

SWC-000

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

Presale.sol

Locations

```
457 * - `spender` cannot be the zero address.
458 */
459 function approve(address spender, uint256 amount) public virtual override returns (bool) {
460 _approve(_msgSender(), spender, amount);
461 return true;
462 }
463
464 /**
```

MEDIUM

Function could be marked as external.

SWC-000

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

Presale.sol

Locations

```
474 * `amount`.
475 */
476 function transferFrom(address sender, address recipient, uint256 amount) public virtual override returns (bool) {
477 _transfer(sender, recipient, amount);
478 _approve(sender, _msgSender(), _allowances[sender][_msgSender()].sub(amount, "BEP20: transfer amount exceeds allowance"));
479 return true;
480 }
481
482 /**
```

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

Presale.sol

Locations

```
492 * - `spender` cannot be the zero address.
493 */
494 function increaseAllowance(address spender, uint256 addedValue) public virtual returns (bool) {
495 _approve(_msgSender(), spender, _allowances[_msgSender()][spender].add(addedValue));
496 return true;
497 }
498
499 /**
```

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

Presale.sol

Locations

```
511 * `subtractedValue`.
512 */
513 function decreaseAllowance(address spender, uint256 subtractedValue) public virtual returns (bool) {
514 _approve(_msgSender(), spender, _allowances[_msgSender()][spender].sub(subtractedValue, "BEP20: decreased allowance below zero"));
515 return true;
516 }
517
518 /**
```

MEDIUM

Function could be marked as external.

SWC-000

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

Presale.sol

Locations

```
664 * @dev Returns the address of the current owner.
665 */
666 function owner() public view returns (address) {
667 return _owner;
668 }
669
670 /**
```

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

Presale.sol

Locations

```
683 * thereby removing any functionality that is only available to the owner.
684 */
685 function renounceOwnership() public virtual onlyOwner {
686 emit OwnershipTransferred(_owner, address(0));
687 _owner = address(0);
688 }
689
690 /**
```

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

Presale.sol

Locations

```
692 * Can only be called by the current owner.
693 */
694 function transferOwnership(address newOwner) public virtual onlyOwner {
695 require(newOwner != address(0), "Ownable: new owner is the zero address");
696 emit OwnershipTransferred(_owner, newOwner);
697 _owner = newOwner;
698 }
699 }
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "getDepositAmount" 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

Presale.sol

Locations

```
766 }
767
768 function getDepositAmount() public view returns (uint256) {
769 return totalDepositedEthBalance;
770 }
771
772 function getLeftTimeAmount() public view returns (uint256) {
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "getLeftTimeAmount" 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

Presale.sol

Locations

```
770 }
771
772 function getLeftTimeAmount() public view returns (uint256) {
773 if(now > presaleEndTimestamp) {
774 return 0;
775 } else {
776 return (presaleEndTimestamp - now);
777 }
778 }
779
780 event Deposited(address indexed user, uint256 amount);
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.2"". 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

Presale.sol

Locations

```
3 // File: @openzeppelin/contracts/utils/Address.sol
4
5 pragma solidity ^0.6.2;
6
7 /**
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.0"". 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

Presale.sol

Locations

```
65 // File: @openzeppelin/contracts/GSN/Context.sol
66
67 pragma solidity ^0.6.0;
68
69 /*
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.0"". 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

Presale.sol

Locations

```
95 // File: @openzeppelin/contracts/token/BEP20/IBEP20.sol
96
97 pragma solidity ^0.6.0;
98
99 /**
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.0"". 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

Presale.sol

Locations

```
174 // File: @openzeppelin/contracts/math/SafeMath.sol
175
176 pragma solidity ^0.6.0;
177
178 /**
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.0"". 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

Presale.sol

Locations

```
328 // File: @openzeppelin/contracts/token/BEP20/BEP20.sol
329
330 pragma solidity ^0.6.0;
331
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.0"". 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

Presale.sol

Locations

```
633 // File: @openzeppelin/contracts/access/Ownable.sol
634
635 pragma solidity ^0.6.0;
636
637 /**
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.0"". 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

Presale.sol

Locations

```
702 /// @title PresaleBEP20 Contract
703
704 pragma solidity ^0.6.0;
705
706 interface MYSTERY {
```

LOW

SWC-131

Unused function parameter "from".

The value of the function parameter "from" for the function "_beforeTokenTransfer" of contract "BEP20" does not seem to be used anywhere in "_beforeTokenTransfer".

Source file

Presale.sol

Locations

```
627 * To learn more about hooks, head to xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].
628 */
629 function _beforeTokenTransfer(address from, address to, uint256 amount) internal virtual { }
630 }
```

LOW

SWC-131

Unused function parameter "to".

The value of the function parameter "to" for the function "_beforeTokenTransfer" of contract "BEP20" does not seem to be used anywhere in "_beforeTokenTransfer".

Source file

Presale.sol

Locations

```
627 * To learn more about hooks, head to xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].
628 */
629 function _beforeTokenTransfer(address from, address to, uint256 amount) internal virtual { }
630 }
```

LOW

SWC-131

Unused function parameter "amount".

The value of the function parameter "amount" for the function "_beforeTokenTransfer" of contract "BEP20" does not seem to be used anywhere in "_beforeTokenTransfer".

Source file

Presale.sol

Locations

```
627 * To learn more about hooks, head to xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].
628 */
629 function _beforeTokenTransfer(address from, address to, uint256 amount) internal virtual { }
630 }
```

LOW

SWC-134

Call with hardcoded gas amount.

The highlighted function call forwards a fixed amount of gas. This is discouraged as the gas cost of EVM instructions may change in the future, which could break this contract's assumptions. If this was done to prevent reentrancy attacks, consider alternative methods such as the checks-effects-interactions pattern or reentrancy locks instead.

Source file

Presale.sol

Locations

```
757 function releaseFunds() external onlyOwner {
758 require(now >= presaleEndTimestamp || totalDepositedEthBalance == hardCapEthAmount, "presale is active");
759 presale.transfer(address(this).balance);
760 }
```