MythX

REPORT 6054D3F9F318BF0018E560AD

Created	Fri Mar 19 2021 16:40:25 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	josephwharold@protonmail.com

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
6621dfdb-c374-4765-9f58-5bf661765e60	browser/contracts/SmrToken.sol	27

Analysis 6621dfdb-c374-4765-9f58-5bf661765e60 MythX

Started	Fri Mar 19 2021 16:40:31 GMT+0000 (Coordinated Universal Time)
Finished	Fri Mar 19 2021 17:27:02 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Remythx
Main Source File	Browser/Contracts/SmrToken.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	14	13

ISSUES

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

browser/contracts/SmrToken.sol

Locations

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

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

browser/contracts/SmrToken.sol

Locations

```
87  * Can only be called by the current owner.
88  */
89  function transferOwnership(address newOwner) public onlyOwner {
90  _transferOwnership(newOwner);
91  }
92
93  /**
```

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

browser/contracts/SmrToken.sol

Locations

```
624 * @dev Returns the token decimals.
625 */
626 function decimals() public override view returns (uint8) {
627 return _decimals;
628 }
629
630 /**
```

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

browser/contracts/SmrToken.sol

Locations

```
631 * @dev Returns the token symbol.
632 */
633 function symbol() public override view returns (string memory) {
634 return _symbol;
635 }
636
637 /**
```

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/SmrToken.sol

Locations

```
638 * @dev See {BEP20-totalSupply}.
639 */
640 function totalSupply() public override view returns (uint256) {
641 return _totalSupply;
642 }
643
644 /**
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "burnSupply" 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/SmrToken.sol

Locations

```
645 * @dev See {BEP20-totalSupply}.
646 */
647 function burnSupply() public view returns (uint256) {
648 return _burnSupply;
649 }
650
651 /**
```

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/SmrToken.sol

Locations

```
664 * - the caller must have a balance of at least `amount`.
665 */
666 function transfer(address recipient, uint256 amount) public override returns (bool) {
667 _transfer(_msgSender(), recipient, amount);
668 return true;
669 }
670
671 /**
```

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/SmrToken.sol

Locations

```
672 * @dev See {BEP20-allowance}.
673 */
674 function allowance(address owner, address spender) public override view returns (uint256) {
675 return _allowances[owner][spender];
676 }
677
678 /**
```

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

browser/contracts/SmrToken.sol

Locations

```
683 * - `spender` cannot be the zero address.
684 */
685 function approve(address spender, uint256 amount) public override returns (bool) {
686 _approve(_msgSender(), spender, amount);
687 return true;
688 }
689
690 /**
```

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

browser/contracts/SmrToken.sol

Locations

```
700 * `amount`.
701 */
702 function transferFrom(
703 address sender,
704 address recipient,
705 uint256 amount
706 ) public override returns (bool) {
707 _transfer(sender, recipient, amount);
708 _approve(
709 sender,
710 _msgSender(),
711 _allowances[sender][_msgSender()].sub(amount, 'BEP20: transfer amount exceeds allowance')
712 );
713 return true;
714 }
715
716 /**
```

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/SmrToken.sol

Locations

```
726 * - `spender` cannot be the zero address.
727 */
728 function increaseAllowance(address spender, uint256 addedValue) public returns (bool) {
729 _approve(_msgSender(), spender, _allowances[_msgSender()][spender].add(addedValue));
730 return true;
731 }
732
733 /**
```

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

browser/contracts/SmrToken.sol

Locations

```
745 * `subtractedValue`.
746 */
747 function decreaseAllowance(address spender, uint256 subtractedValue) public returns (bool) {
748 _approve(
749 _msgSender(),
750 spender,
751 _allowances[_msgSender()][spender].sub(subtractedValue, 'BEP20: decreased allowance below zero')
752 );
753 return true;
754 }
755
756 /**
```

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

browser/contracts/SmrToken.sol

Locations

```
762 * - `msg.sender` must be the token owner
763 */
764 function mint(uint256 amount) public onlyOwner returns (bool) {
765 _mint(_msgSender(), amount);
766 return true;
767 }
768
769 /**
```

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/SmrToken.sol

Locations

```
883
884 /// @notice Creates `_amount` token to `_to`. Must only be called by the owner (MasterSamurai).
885 function mint(address _to, uint256 _amount) public onlyOwner {
886 _mint(_to, _amount);
887 _moveDelegates(address(0), _delegates[_to], _amount);
888 }
889
890 /// @dev overrides transfer function to meet tokenomics of SMR
```

LOW

A floating pragma is set.

SWC-103

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/SmrToken.sol

Locations

```
1
2 pragma solidity ^0.4.0;
3
4 /*
```

LOW

A floating pragma is set.

SWC-103

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/SmrToken.sol

Locations

```
27 }
28
29 pragma solidity ^0.4.0;
30
```

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

browser/contracts/SmrToken.sol

Locations

```
386 }
387
388 pragma solidity ^0.6.2;
389
390 /**
```

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

browser/contracts/SmrToken.sol

Locations

```
198 }
199
200 pragma solidity ^0.6.2;
201
202 /**
```

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

browser/contracts/SmrToken.sol

Locations

```
101 }
102
103 pragma solidity ^0.6.2;
104
105 interface IBEP20 {
```

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/SmrToken.sol

Locations

```
546 }
547
548 pragma solidity ^0.4.0;
549
```

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/SmrToken.sol

Locations

```
1010 require(signatory != address(0), "SMR::delegateBySig: invalid signature");
1011 require(nonce == nonces[signatory]++, "SMR::delegateBySig: invalid nonce");
1012 require(now <= expiry, "SMR::delegateBySig: signature expired");
1013 return _delegate(signatory, delegatee);
1014 }
```

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/SmrToken.sol

Locations

```
1040 returns (uint256)
1041 {
1042 require(blockNumber < block.number, "SMR::getPriorVotes: not yet determined");
1043
1044 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/SmrToken.sol

Locations

```
1113 internal
1114 {
1115 uint32 blockNumber = safe32(block.number, "SMR::_writeCheckpoint: block number exceeds 32 bits");
1116
1117 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/SmrToken.sol

Locations

```
1040 returns (uint256)
1041 {
1042 require(blockNumber < block.number, "SMR::getPriorVotes: not yet determined");
1043
1044 uint32 nCheckpoints = numCheckpoints[account];
```

LOW

SWC-128

Loop over unbounded data structure.

Gas consumption in function "sqrt" in contract "SafeMath" 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/SmrToken.sol

Locations

```
376 z = y;
377 uint256 x = y / 2 + 1;
378 while (x < z) {
379 z = x;
380 x = (y / x + x) / 2;
```

LOW

SWC-128

Potentially unbounded data structure passed to builtin.

Gas consumption in function "delegateBySig" in contract "SmrToken" 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/SmrToken.sol

Locations

```
984 abi.encode(
985 DOMAIN_TYPEHASH,
986 keccak256(bytes(name())),
987 getChainId(),
988 address(this)
```

LOW

SWC-128

Loop over unbounded data structure.

Gas consumption in function "getPriorVotes" in contract "SmrToken" 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/SmrToken.sol

Locations

```
1059 uint32 lower = 0;
1060 uint32 upper = nCheckpoints - 1;
1061 while (upper > lower) {
1062 uint32 center = upper - (upper - lower) / 2; // ceil, avoiding overflow
1063 Checkpoint memory cp = checkpoints[account][center];
```