

REPORT 60C0BB5B72369300185C706C

Created	Wed Jun 09 2021 13:00:11 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	60a253128bfa12714ff2900a

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
15956e6b-599f-4815-acf6-3693bb2fc7d4	contracts/SiberianToken.sol	33

Started	Wed Jun 09 2021 13:00:17 GMT+0000 (Coordinated Universal Time)
Finished	Wed Jun 09 2021 13:15:19 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Client Tool	Remythx
Main Source File	Contracts/SiberianToken.sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	21	12

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

contracts/SiberianToken.sol

Locations

```
806 * thereby removing any functionality that is only available to the owner.
807 */
808 function renounceOwnership() public virtual onlyOwner {
809     emit OwnershipTransferred(_owner, address(0));
810     _owner = address(0);
811 }
812
813 /**
```

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

contracts/SiberianToken.sol

Locations

```
815 | * Can only be called by the current owner.
816 | */
817 | function transferOwnership(address newOwner) public virtual onlyOwner {
818 |     require(newOwner != address(0), "Ownable: new owner is the zero address");
819 |     emit OwnershipTransferred(_owner, newOwner);
820 |     _owner = newOwner;
821 | }
822 | }
```

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

contracts/SiberianToken.sol

Locations

```
901 | * @dev Returns the token decimals.
902 | */
903 | function decimals() public override view returns (uint8) {
904 |     return _decimals;
905 | }
906 |
907 | /**
```

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

contracts/SiberianToken.sol

Locations

```
908 | * @dev Returns the token symbol.
909 | */
910 | function symbol() public override view returns (string memory) {
911 |     return _symbol;
912 | }
913 |
914 | /**
```

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

contracts/SiberianToken.sol

Locations

```
934 * - the caller must have a balance of at least `amount`.
935 */
936 function transfer(address recipient, uint256 amount) public override returns (bool) {
937     transfer(msgSender(), recipient, amount);
938     return true;
939 }
940
941 /**
```

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

contracts/SiberianToken.sol

Locations

```
942 * @dev See {BEP20-allowance}.
943 */
944 function allowance(address owner, address spender) public override view returns (uint256) {
945     return _allowances[owner][spender];
946 }
947
948 /**
```

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

contracts/SiberianToken.sol

Locations

```
953 * - `spender` cannot be the zero address.
954 */
955 function approve(address spender, uint256 amount) public override returns (bool) {
956     approve(msgSender(), spender, amount);
957     return true;
958 }
959
960 /**
```

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

contracts/SiberianToken.sol

Locations

```
970 * `amount`.
971 */
972 function transferFrom
973 address sender
974 address recipient
975 uint256 amount
976 public override returns (bool) {
977     transfer(sender, recipient, amount);
978     approve(
979         sender,
980         msgSender(),
981         allowances[sender][msgSender()].sub(amount, "BEP20: transfer amount exceeds allowance")
982     );
983     return true;
984 }
985
986 /**
```

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

contracts/SiberianToken.sol

Locations

```
996 * - `spender` cannot be the zero address.
997 */
998 function increaseAllowance(address spender, uint256 addedValue) public returns (bool) {
999     approve(msgSender(), spender, _allowances[msgSender()][spender].add(addedValue));
1000     return true;
1001 }
1002
1003 /**
```

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

contracts/SiberianToken.sol

Locations

```
1015 | * `subtractedValue`.
1016 | */
1017 | function decreaseAllowance(address spender, uint256 subtractedValue) public returns (bool) {
1018 |     approve(
1019 |         msgSender(),
1020 |         spender,
1021 |         _allowances[msgSender()][spender].sub(subtractedValue, "BEP20: decreased allowance below zero")
1022 |     );
1023 |     return true;
1024 | }
1025 |
1026 | /**
```

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

contracts/SiberianToken.sol

Locations

```
1032 | * - `msg.sender` must be the token owner
1033 | */
1034 | function mint(uint256 amount) public onlyOwner returns (bool) {
1035 |     _mint(msgSender(), amount);
1036 |     return true;
1037 | }
1038 |
1039 | /**
```

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

contracts/SiberianToken.sol

Locations

```
1232 |
1233 | /// @notice Creates `_amount` token to `_to`. Must only be called by the owner (MasterChef).
1234 | function mint(address _to, uint256 _amount) public onlyOwner {
1235 |     _mint(_to, _amount);
1236 |     _moveDelegates(address(0), _delegates[_to], _amount);
1237 | }
1238 |
1239 | /// @dev overrides transfer function to meet tokenomics of SIBERIAN
```

MEDIUM Function could be marked as external.

SWC-000

The function definition of "isExcludedFromAntiWhale" 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/SiberianToken.sol

Locations

```
1349 | * @dev Returns the address is excluded from antiWhale or not.
1350 | */
1351 | function isExcludedFromAntiWhale(address _account) public view returns (bool) {
1352 |     return _excludedFromAntiWhale[_account];
1353 | }
1354 |
1355 | // To receive BNB from siberianSwapRouter when swapping
```

MEDIUM Function could be marked as external.

SWC-000

The function definition of "updateTransferTaxRate" 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/SiberianToken.sol

Locations

```
1360 | * Can only be called by the current operator.
1361 | */
1362 | function updateTransferTaxRate(uint16 _transferTaxRate) public onlyOperator {
1363 |     require(_transferTaxRate <= MAXIMUM_TRANSFER_TAX_RATE, "SIBERIAN::updateTransferTaxRate: Transfer tax rate must not exceed the maximum rate.");
1364 |     emit TransferTaxRateUpdated(msg.sender, transferTaxRate, _transferTaxRate);
1365 |     transferTaxRate = _transferTaxRate;
1366 | }
1367 |
1368 | /**
```

MEDIUM Function could be marked as external.

SWC-000

The function definition of "updateBurnRate" 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/SiberianToken.sol

Locations

```
1370 | * Can only be called by the current operator.
1371 | */
1372 | function updateBurnRate(uint16 _burnRate) public onlyOperator {
1373 |     require(_burnRate <= 100, "SIBERIAN::updateBurnRate: Burn rate must not exceed the maximum rate.");
1374 |     emit BurnRateUpdated(msg.sender, burnRate, _burnRate);
1375 |     burnRate = _burnRate;
1376 | }
1377 |
1378 | /**
```

MEDIUM Function could be marked as external.

SWC-000

The function definition of "updateMaxTransferAmountRate" 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/SiberianToken.sol

Locations

```
1380 * Can only be called by the current operator.
1381 */
1382 function updateMaxTransferAmountRate(uint16 _maxTransferAmountRate) public onlyOperator {
1383     require(_maxTransferAmountRate <= 10000, "SIBERIAN::updateMaxTransferAmountRate: Max transfer amount rate must not exceed the maximum rate.");
1384     emit MaxTransferAmountRateUpdated(msg.sender, maxTransferAmountRate, _maxTransferAmountRate);
1385     maxTransferAmountRate = _maxTransferAmountRate;
1386 }
1387
1388 /**
```

MEDIUM Function could be marked as external.

SWC-000

The function definition of "updateMinAmountToLiquify" 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/SiberianToken.sol

Locations

```
1390 * Can only be called by the current operator.
1391 */
1392 function updateMinAmountToLiquify(uint256 _minAmount) public onlyOperator {
1393     emit MinAmountToLiquifyUpdated(msg.sender, minAmountToLiquify, _minAmount);
1394     minAmountToLiquify = _minAmount;
1395 }
1396
1397 /**
```

MEDIUM Function could be marked as external.

SWC-000

The function definition of "setExcludedFromAntiWhale" 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/SiberianToken.sol

Locations

```
1399 * Can only be called by the current operator.
1400 */
1401 function setExcludedFromAntiWhale(address _account, bool _excluded) public onlyOperator {
1402     excludedFromAntiWhale[_account] = _excluded;
1403 }
1404
1405 /**
```


MEDIUM Function could be marked as external.

SWC-000

The function definition of "updateSwapAndLiquifyEnabled" 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/SiberianToken.sol

Locations

```
1407 * Can only be called by the current operator.
1408 */
1409 function updateSwapAndLiquifyEnabled(bool _enabled) public onlyOperator {
1410     emit SwapAndLiquifyEnabledUpdated(msg.sender, _enabled);
1411     swapAndLiquifyEnabled = _enabled;
1412 }
1413
1414 /**
```

MEDIUM Function could be marked as external.

SWC-000

The function definition of "updateSiberianSwapRouter" 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/SiberianToken.sol

Locations

```
1416 * Can only be called by the current operator.
1417 */
1418 function updateSiberianSwapRouter(address _router) public onlyOperator {
1419     siberianSwapRouter = IUniswapV2Router02(_router);
1420     siberianSwapPair = IUniswapV2Factory(siberianSwapRouter.factory()).getPair(address(this), siberianSwapRouter.WETH());
1421     require(siberianSwapPair != address(0), "SIBERIAN::updateSiberianSwapRouter: Invalid pair address.");
1422     emit SiberianSwapRouterUpdated(msg.sender, address(siberianSwapRouter), siberianSwapPair);
1423 }
1424
1425 /**
```

MEDIUM Function could be marked as external.

SWC-000

The function definition of "transferOperator" 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/SiberianToken.sol

Locations

```
1434 * Can only be called by the current operator.
1435 */
1436 function transferOperator(address newOperator) public onlyOperator {
1437     require(newOperator != address(0), "SIBERIAN::transferOperator: new operator is the zero address");
1438     emit OperatorTransferred(_operator, newOperator);
1439     _operator = newOperator;
1440 }
1441
1442 // Copied and modified from YAM code:
```

LOW

A floating pragma is set.

SWC-103

The current pragma Solidity directive is "">=0.5.0"". 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

contracts/SiberianToken.sol

Locations

```
7 // File: @uniswap/v2-core/contracts/interfaces/IUniswapV2Factory.sol
8
9 pragma solidity >=0.5.0
10
11 interface IUniswapV2Factory {
```

LOW

A floating pragma is set.

SWC-103

The current pragma Solidity directive is "">=0.5.0"". 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

contracts/SiberianToken.sol

Locations

```
27 // File: @uniswap/v2-core/contracts/interfaces/IUniswapV2Pair.sol
28
29 pragma solidity >=0.5.0
30
31 interface IUniswapV2Pair {
```

LOW

A floating pragma is set.

SWC-103

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

contracts/SiberianToken.sol

Locations

```
82 // File: @uniswap/v2-periphery/contracts/interfaces/IUniswapV2Router01.sol
83
84 pragma solidity >=0.6.2
85
86 interface IUniswapV2Router01 {
```

LOW

A floating pragma is set.

SWC-103

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

contracts/SiberianToken.sol

Locations

```
180 | // File: @uniswap/v2-periphery/contracts/interfaces/IUniswapV2Router02.sol
181 |
182 | pragma solidity >=0.6.2
183 |
```

LOW

A floating pragma is set.

SWC-103

The current pragma Solidity directive is `">=0.6.2<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

contracts/SiberianToken.sol

Locations

```
226 | // File: @openzeppelin/contracts/utils/Address.sol
227 |
228 | pragma solidity >=0.6.2 <0.8.0
229 |
230 | /**
```

LOW

A floating pragma is set.

SWC-103

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

contracts/SiberianToken.sol

Locations

```
416 | // File: @openzeppelin/contracts/math/SafeMath.sol
417 |
418 | pragma solidity >=0.6.0 <0.8.0
419 |
420 | /**
```

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

contracts/SiberianToken.sol

Locations

```
631 | // File: contracts/libs/IBEP20.sol
632 |
633 | pragma solidity >=0.4.0
634 |
635 | interface IBEP20 {
```

LOW

A floating pragma is set.

SWC-103

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

contracts/SiberianToken.sol

Locations

```
731 | // File: @openzeppelin/contracts/utils/Context.sol
732 |
733 | pragma solidity >=0.6.0<0.8.0
734 |
735 | /*
```

LOW

A floating pragma is set.

SWC-103

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

contracts/SiberianToken.sol

Locations

```
756 | // File: @openzeppelin/contracts/access/Ownable.sol
757 |
758 | pragma solidity >=0.6.0<0.8.0
759 |
760 | /**
```

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

contracts/SiberianToken.sol

Locations

```
824 | // File: contracts/libs/BEP20.sol
825 |
826 | pragma solidity >=0.4.0
827 |
```

LOW

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

SWC-120

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/SiberianToken.sol

Locations

```
1574 | returns (uint256)
1575 | {
1576 | require(blockNumber < block.number, "SIBERIAN::getPriorVotes: not yet determined");
1577 |
1578 | uint32 nCheckpoints = numCheckpoints[account];
```

LOW

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

SWC-120

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/SiberianToken.sol

Locations

```
1647 | internal
1648 | {
1649 | uint32 blockNumber = safe32(block.number, "SIBERIAN::writeCheckpoint: block number exceeds 32 bits");
1650 |
1651 | if (nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber) {
```