Carbon
Smart Contract Audit Report

April 20, 2021

IMMUNE BYTES

Introduction

1. About Carbon

Carbon is a decentralized platform that integrates DeFi and NFT which is building on Binance Smart Chain. The platform has planned three phases of products: CarbonSlot& combustion mining, Carbon Cash fluidity mining, and NFT related products.

The Carbon economic models cover tokens: platform governance coin CAR (Carbon), algorithm stability coin Carbon Cash (CARC), and dividend coin Carbon Stock (CARS), Carbon strives to use the latest and most scientific economic model to create a long-term value decentralized DeFi&NFT platform

Visit https://carbonfi.org/ to know more.

2. About ImmuneBytes

ImmuneBytes is a security start-up to provide professional services in the blockchain space. The team has hands-on experience in conducting smart contract audits, penetration testing, and security consulting. ImmuneBytes's security auditors have worked on various A-league projects and have a great understanding of DeFi projects like AAVE, Compound, 0x Protocol, Uniswap, dydx.

The team has been able to secure 15+ blockchain projects by providing security services on different frameworks. ImmuneBytes team helps start-up with a detailed analysis of the system ensuring security and managing the overall project.

Visit http://immunebytes.com/ to know more about the services.

Documentation Details

The Carbon team has provided documentation for the purpose of conducting the audit. The documents are:

1. https://carbonfi.org/static/pdf/Carbon_litepaper_EN_V2.4_20210414.pdf

Audit Process & Methodology

ImmuneBytes team has performed thorough testing of the project starting with analyzing the code design patterns in which we reviewed the smart contract architecture to ensure it is structured and safe use of third-party smart contracts and libraries.

Our team then performed a formal line-by-line inspection of the Smart Contract in order to find any potential issues like Signature Replay Attacks, Unchecked External Calls, External Contract Referencing, Variable Shadowing, Race conditions, Transaction-ordering dependence, timestamp dependence, DoS attacks, and others.

In the Unit testing phase, we run unit tests written by the developer in order to verify the functions work as intended. In Automated Testing, we tested the Smart Contract with our in-house developed tools to identify vulnerabilities and security flaws.

The code was audited by a team of independent auditors which includes -

1. Testing the functionality of the Smart Contract to determine proper logic has been followed throughout.
2. Analyzing the complexity of the code by thorough, manual review of the code, line-by-line.
3. Deploying the code on testnet using multiple clients to run live tests.
4. Analyzing failure preparations to check how the Smart Contract performs in case of bugs and vulnerabilities.
5. Checking whether all the libraries used in the code are on the latest version.
6. Analyzing the security of the on-chain data.

Audit Details

- Project Name: Carbon
- Languages: Solidity(Smart contract), Javascript(Unit Testing)
- Testnet Code (Testnet):
    - CarbonSlotAMMPancake: 0x70949fFB406F066d235EF424672Af10aCE68610f
    - CarbonSlotV2: 0x5380cfCf1994633709DC9D2c12ebc3307D39fda1
    - CarbonToken: 0xba66cB2068b8337c64F9d15E511B77A56A863006

Audit Goals

The focus of the audit was to verify that the smart contract system is secure, resilient, and working according to its specifications. The audit activities can be grouped into the following three categories:

1. Security: Identifying security-related issues within each contract and within the system of contracts.
2. Sound Architecture: Evaluation of the architecture of this system through the lens of established smart contract best practices and general software best practices.
3. Code Correctness and Quality: A full review of the contract source code. The primary areas of focus include:
    a. Correctness
    b. Readability
    c. Sections of code with high complexity
    d. Quantity and quality of test coverage

Security Level References

Every issue in this report was assigned a severity level from the following:

High severity issues will bring problems and should be fixed.

Medium severity issues could potentially bring problems and should eventually be fixed.

Low severity issues are minor details and warnings that can remain unfixed but would be better fixed at some point in the future.

Issues	High	Medium	Low
Open	-	2	12
Closed	-	-	-

High severity issues

Medium severity issues

1. There are multiple instances in the contracts where division is done before multiplication. As there are no floating points in solidity and to prevent overflow/underflow it is advised that multiplication be done before division and SafeMath(which is imported but never used) be used for all arithmetic operations. For eg.

```
259             uint256 x3Reward = ((levelPrice[level]) / 100) * 40;
260             sendETHDividends(referrerAddress, userAddress, 1, level, x3Reward);
261
262             uint256 toStaticPool = ((levelPrice[level]) / 100) * 10;
263             staticPool += toStaticPool;
```

2. The **payBackETH()** function, in the CarbonSlotAMMPancake contract, is declared internal and not called inside any other function which makes the function redundant and uncallable. No need to typecast **msg.sender,** you can use **transfer** on it directly.

```
81      function payBackETH() internal onlyOperator {
82          address(uint160(msg.sender)).transfer(address(this).balance);
83      }
```

Low severity issues

1. It is a good practice to lock the solidity version for a live deployment (use **0.6.2** instead of **>=0.6.2**). Contracts should be deployed with the same compiler version and flags that they have been tested the most with. Locking the pragma helps ensure that contracts do not accidentally get deployed using, for example, the latest compiler which may have higher risks of undiscovered bugs. Contracts may also be deployed by others and the pragma indicates the compiler version intended by the original authors.

```
1   // SPDX-License-Identifier: MIT
2   pragma solidity >=0.6.2;
```

2. We can omit **carbonTokenAddress** in both CarbonSlotV2 and CarbonSlotAMMPancake contracts. Wherever the address is needed we can use **address(carbonToken)**

```
48      //test
49      IBEP20 public carbonToken;
50      address public carbonTokenAddress;
```

3. Variable declared but never used : -

```
54      uint256 public mineRate = 1000;
```

4. Reinvest event is declared but never used/emitted

```
84      event Reinvest(
85          address indexed user,
86          address indexed currentReferrer,
87          address indexed caller,
88          uint8 matrix,
89          uint8 level
90      );
```

5. AMMCalled event is declared but never used/emitted

```
125     event AMMCalled(uint256 ethAmount, uint256[] amounts);
```

6. CARSwapped event is declared but never used/emitted

```
21      event CARSwapped(uint256 carAmount, uint256 ethAmount);
```

7. Use the **emit** keyword before emitting an event

```
68              CARSwappedV2(ethAmount, amounts);
```

8. The **bytesToAddress()** function is not used anywhere in the CarbonSlotV2 contract. Additionally, it is a private function so which makes it redundant and unusable.

```
506     function bytesToAddress(bytes memory bys)
507         private
508         pure
509         returns (address addr)
510     {
511         assembly {
512             addr := mload(add(bys, 20))
513         }
514     }
```

9. Instead of updating the **openAMM** variable in the CarbonSlotV2 contract, it's better to fetch it from the CarbonSlotAMMPancake contract to avoid any discrepancies. A way to do that is to include it in the CarbonSlotAMMInterface.

```
520     function refresAMM(bool _isOpen) external onlyOperator {
521         openAMM = _isOpen;
522     }
```

10. Referrer address has been set twice. We can omit line 178.

```
175         User memory user = User({referrer: referrerAddress, partnersCount: 0}); // id: lastUserId,
176
177         users[userAddress] = user;
178         users[userAddress].referrer = referrerAddress;
```

11. **Context.sol** and **IUinswapV2Interface.sol** is imported but not used anywhere in the **CarbonSlotV2** contract.

```
4   import '../Context.sol';
5   import '../IBEP20.sol';
6   import '../IUinswapV2Interface.sol';
7   import '../Libraries.sol';
```

12. **Context.sol** is imported but not used anywhere in the **CarbonSlotAMMPancake** contract.

```
4   import '../Context.sol';
5   import '../IBEP20.sol';
6   import '../IPancakeInterface.sol';
7   import './SlotAMMInterface.sol';
```

Recommendations

1. In the **transferMineCARtoAddress()** function we can skip the check in the if condition because **safeTransfer** takes care of that.

```
436     function transferMineCARtoAddress(address target, uint256 amount) internal {
437         if (carbonToken.balanceOf(address(this)) >= amount) {
438             carbonToken.safeTransfer(target, amount);
439             globalMine = globalMine + amount;
440
441             emit CARMined(target, amount);
442         }
443     }
```

2. In the **etherProceeds()** function we can replace the whole line 579 with: **msg.sender.transfer(address(this).balance);**

```
578     function etherProceeds() external onlyOperator {
579         if (!msg.sender.send(address(this).balance)) revert();
580     }
```

3. Instead of type casting to **uint160** and then type casting to **address** we can simply type cast to **payable** in both cases**.**

```
450                 if (!address(uint160(initNode)).send(amount)) {
451                     address(uint160(initNode)).transfer(amount);
452                 }
```

```
497         if (!address(uint160(receiver)).send(ethValue)) {
498             address(uint160(receiver)).transfer(address(this).balance);
499             return;
500         }
```

4. The following **if** condition can be skipped and only the **transfer** statement is sufficient.

```
450                 if (!address(uint160(initNode)).send(amount)) {
451                     address(uint160(initNode)).transfer(amount);
452                 }
```

5. The second **if** condition should come first if we need the **SentExtraEthDividends** to be emitted if it satisfies the **if** condition. In many cases the second **if** condition won't be

executed.

```
497         if (!address(uint160(receiver)).send(ethValue)) {
498             address(uint160(receiver)).transfer(address(this).balance);
499             return;
500         }
501         if (isExtraDividends) {
502             emit SentExtraEthDividends(_from, receiver, matrix, level);
503         }
```

6. The **findCount** variable can be omitted as it isn't used anywhere.

```
390     function findActivedReferrer(address userAddress, uint8 level)
391         public
392         view
393         returns (address)
394     {
395         uint8 findCount = 0;
396         while (true) {
397             findCount++;
398             if (users[users[userAddress].referrer].activeLevels[level]) {
399                 return users[userAddress].referrer;
400             } else {
401                 userAddress = users[userAddress].referrer;
402             }
403         }
404     }
```

Automated Audit

Solhint Linting Violations

Solhint is an open-source project for linting solidity code, providing both security and style guide validations. It integrates seamlessly into most mainstream IDEs. We used Solhint as a plugin within our VScode for this analysis. Multiple Linting violations were detected by Solhint, it is recommended to use Solhint's npm package to lint the contract.

Contract Library

Contract-library contains the most complete, high-level decompiled representation of all Ethereum smart contracts, with security analysis applied to these in real-time.
We performed analysis using the contract Library on the Ropsten address of the Carbon contracts used during manual testing:

- CarbonSlotV2: 0x5380CFCF1994633709DC9D2C12EBC3307D39FDA1
- CarbonSlotAMMPancake: 0x70949fFB406F066d235EF424672Af10aCE68610f
- CarbonToken: 0xba66cB2068b8337c64F9d15E511B77A56A863006

It raised no warnings in any of the contracts.

Mythril

Mythril is a security analysis tool for EVM bytecode. It detects security vulnerabilities in smart contracts built for Ethereum, Hedera, Quorum, Vechain, Roostock, Tron and other EVM-compatible blockchains. It uses symbolic execution, SMT solving and taint analysis to detect a variety of security vulnerabilities. Mythril raised the following concerns:

1. CarbonSlotV2

2. CarbonSlotAMMPancake

```
luv@luv:~/Downloads/Telegram Desktop/contracts$ myth analyze carbon/SlotAMMPancake.sol
The analysis was completed successfully. No issues were detected.
```

3. CarbonToken

```
luv@luv:~/Downloads/Telegram Desktop/contracts$ myth analyze tokens/CarbonToken.sol
The analysis was completed successfully. No issues were detected.
```

Slither

Slither, an open-source static analysis framework. This tool provides rich information about Ethereum smart contracts and has critical properties. While Slither is built as a security-oriented static analysis framework, it is also used to enhance the user's understanding of smart contracts, assist in code reviews, and detect missing optimizations. Slither detected the following issues:

1. CarbonSlotV2

```
INFO:Detectors:
CarbonSlotV2.transferETHToAMM(uint256) (carbon/SlotV3.sol#445-454) sends eth to arbitrary user
	Dangerous calls:
	- slotAMM.swapETHForCAR{value: amount}() (carbon/SlotV3.sol#447)
	- ! address(uint160(initNode)).send(amount) (carbon/SlotV3.sol#450)
	- address(uint160(initNode)).transfer(amount) (carbon/SlotV3.sol#451)
CarbonSlotV2.sendETHDividends(address,address,uint8,uint8,uint256) (carbon/SlotV3.sol#481-504) sends eth to
arbitrary user
	Dangerous calls:
	- ! address(uint160(receiver)).send(ethValue) (carbon/SlotV3.sol#497)
	- address(uint160(receiver)).transfer(address(this).balance) (carbon/SlotV3.sol#498)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
Reentrancy in CarbonSlotV2.registration(address,address) (carbon/SlotV3.sol#162-208):
	External calls:
	- updateMatrixReferrer(userAddress,actived Referrer,1) (carbon/SlotV3.sol#185)
		- returndata = address(token).functionCall(data,SafeBEP20: low-level call failed)
(Libraries.sol#419-423)
		- carbonToken.safeTransfer(target,amount) (carbon/SlotV3.sol#438)
		- slotAMM.swapETHForCAR{value: amount}() (carbon/SlotV3.sol#447)
		- (success,returndata) = target.call{value: weiValue}(data) (Libraries.sol#180-181)
	External calls sending eth:
	- updateMatrixReferrer(userAddress,actived Referrer,1) (carbon/SlotV3.sol#185)
		- slotAMM.swapETHForCAR{value: amount}() (carbon/SlotV3.sol#447)
		- ! address(uint160(initNode)).send(amount) (carbon/SlotV3.sol#450)
		- address(uint160(initNode)).transfer(amount) (carbon/SlotV3.sol#451)
		- (success,returndata) = target.call{value: weiValue}(data) (Libraries.sol#180-181)
		- ! address(uint160(receiver)).send(ethValue) (carbon/SlotV3.sol#497)
		- address(uint160(receiver)).transfer(address(this).balance) (carbon/SlotV3.sol#498)
	State variables written after the call(s):
	- addrRegisted[userAddress] = true (carbon/SlotV3.sol#197)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
INFO:Detectors:
CarbonSlotV2.updateMatrixReferrer(address,address,uint8) (carbon/SlotV3.sol#248-324) performs a multiplication
on the result of a division:
	-x3Reward = ((levelPrice[level]) / 100) * 40 (carbon/SlotV3.sol#259)
CarbonSlotV2.updateMatrixReferrer(address,address,uint8) (carbon/SlotV3.sol#248-324) performs a multiplication
on the result of a division:
	-toStaticPool = ((levelPrice[level]) / 100) * 10 (carbon/SlotV3.sol#262)
CarbonSlotV2.updateMatrixReferrer(address,address,uint8) (carbon/SlotV3.sol#248-324) performs a multiplication
on the result of a division:
	-x3Reward_scope_0 = ((levelPrice[level]) / 100) * 40 (carbon/SlotV3.sol#275)
```

```
CarbonSlotV2.updateMatrixReferrer(address,address,uint8) (carbon/SlotV3.sol#248-324) performs a multiplication
on the result of a division:
        -toStaticPool_scope_1 = ((levelPrice[level]) / 100) * 10 (carbon/SlotV3.sol#277)
CarbonSlotV2.updateMatrixReferrer(address,address,uint8) (carbon/SlotV3.sol#248-324) performs a multiplication
on the result of a division:
        -x3Reward_scope_2 = ((levelPrice[level]) / 100) * 80 (carbon/SlotV3.sol#296)
CarbonSlotV2.updateMatrixM2Referrer(address,address,uint8,uint256) (carbon/SlotV3.sol#326-383) performs a
multiplication on the result of a division:
        -realX2Reward = (x2Reward / 100) * 90 (carbon/SlotV3.sol#340)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
CarbonSlotV2.findEthReceiver(address,address,uint8).isExtraDividends (carbon/SlotV3.sol#470) is a local variable
never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
CarbonSlotV2.transferETHToAMM(uint256) (carbon/SlotV3.sol#445-454) ignores return value by
slotAMM.swapETHForCAR{value: amount}() (carbon/SlotV3.sol#447)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
Reentrancy in CarbonSlotV2.buyNewLevel(uint8) (carbon/SlotV3.sol#210-246):
        External calls:
        - updateMatrixReferrer(msg.sender,actived ReferrerAddress,level) (carbon/SlotV3.sol#233)
                - returndata = address(token).functionCall(data,SafeBEP20: low-level call failed)
(Libraries.sol#419-423)
                - carbonToken.safeTransfer(target,amount) (carbon/SlotV3.sol#438)
                - slotAMM.swapETHForCAR{value: amount}() (carbon/SlotV3.sol#447)
                - (success,returndata) = target.call{value: weiValue}(data) (Libraries.sol#180-181)
        External calls sending eth:
        - updateMatrixReferrer(msg.sender,actived ReferrerAddress,level) (carbon/SlotV3.sol#233)
                - slotAMM.swapETHForCAR{value: amount}() (carbon/SlotV3.sol#447)
                - ! address(uint160(initNode)).send(amount) (carbon/SlotV3.sol#450)
                - address(uint160(initNode)).transfer(amount) (carbon/SlotV3.sol#451)
                - (success,returndata) = target.call{value: weiValue}(data) (Libraries.sol#180-181)
                - ! address(uint160(receiver)).send(ethValue) (carbon/SlotV3.sol#497)
                - address(uint160(receiver)).transfer(address(this).balance) (carbon/SlotV3.sol#498)
        State variables written after the call(s):
        - globalInvest = globalInvest + msg.value (carbon/SlotV3.sol#234)
        - userEpochs[msg.sender].push(epoch) (carbon/SlotV3.sol#243)
Reentrancy in CarbonSlotV2.registration(address,address) (carbon/SlotV3.sol#162-208):
        External calls:
        - updateMatrixReferrer(userAddress,actived Referrer,1) (carbon/SlotV3.sol#185)
                - returndata = address(token).functionCall(data,SafeBEP20: low-level call failed)
(Libraries.sol#419-423)
                - carbonToken.safeTransfer(target,amount) (carbon/SlotV3.sol#438)
                - slotAMM.swapETHForCAR{value: amount}() (carbon/SlotV3.sol#447)
                - (success,returndata) = target.call{value: weiValue}(data) (Libraries.sol#180-181)
        External calls sending eth:
        - updateMatrixReferrer(userAddress,actived Referrer,1) (carbon/SlotV3.sol#185)
                - slotAMM.swapETHForCAR{value: amount}() (carbon/SlotV3.sol#447)
                - ! address(uint160(initNode)).send(amount) (carbon/SlotV3.sol#450)
                - address(uint160(initNode)).transfer(amount) (carbon/SlotV3.sol#451)
                - (success,returndata) = target.call{value: weiValue}(data) (Libraries.sol#180-181)
                - ! address(uint160(receiver)).send(ethValue) (carbon/SlotV3.sol#497)
                - address(uint160(receiver)).transfer(address(this).balance) (carbon/SlotV3.sol#498)
        State variables written after the call(s):
        - globalInvest = globalInvest + msg.value (carbon/SlotV3.sol#199)
        - globalInvestAddrCount += 1 (carbon/SlotV3.sol#200)
```

```
        - userEpochs[userAddress].push(epoch) (carbon/SlotV3.sol#194)
Reentrancy in CarbonSlotV2.transferMineCARtoAddress(address,uint256) (carbon/SlotV3.sol#436-443):
        External calls:
        - carbonToken.safeTransfer(target,amount) (carbon/SlotV3.sol#438)
        State variables written after the call(s):
        - globalMine = globalMine + amount (carbon/SlotV3.sol#439)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in CarbonSlotV2.buyNewLevel(uint8) (carbon/SlotV3.sol#210-246):
        External calls:
        - updateMatrixReferrer(msg.sender,activedReferrerAddress,level) (carbon/SlotV3.sol#233)
                - returndata = address(token).functionCall(data,SafeBEP20: low-level call failed)
(Libraries.sol#419-423)
                - carbonToken.safeTransfer(target,amount) (carbon/SlotV3.sol#438)
                - slotAMM.swapETHForCAR{value: amount}() (carbon/SlotV3.sol#447)
                - (success,returndata) = target.call{value: weiValue}(data) (Libraries.sol#180-181)
        External calls sending eth:
        - updateMatrixReferrer(msg.sender,activedReferrerAddress,level) (carbon/SlotV3.sol#233)
                - slotAMM.swapETHForCAR{value: amount}() (carbon/SlotV3.sol#447)
                - ! address(uint160(initNode)).send(amount) (carbon/SlotV3.sol#450)
                - address(uint160(initNode)).transfer(amount) (carbon/SlotV3.sol#451)
                - (success,returndata) = target.call{value: weiValue}(data) (Libraries.sol#180-181)
                - ! address(uint160(receiver)).send(ethValue) (carbon/SlotV3.sol#497)
                - address(uint160(receiver)).transfer(address(this).balance) (carbon/SlotV3.sol#498)
        Event emitted after the call(s):
        - Upgrade(msg.sender,activedReferrerAddress,1,level) (carbon/SlotV3.sol#245)
Reentrancy in CarbonSlotV2.registration(address,address) (carbon/SlotV3.sol#162-208):
        External calls:
        - updateMatrixReferrer(userAddress,activedReferrer,1) (carbon/SlotV3.sol#185)
                - returndata = address(token).functionCall(data,SafeBEP20: low-level call failed)
(Libraries.sol#419-423)
                - carbonToken.safeTransfer(target,amount) (carbon/SlotV3.sol#438)
                - slotAMM.swapETHForCAR{value: amount}() (carbon/SlotV3.sol#447)
                - (success,returndata) = target.call{value: weiValue}(data) (Libraries.sol#180-181)
        External calls sending eth:
        - updateMatrixReferrer(userAddress,activedReferrer,1) (carbon/SlotV3.sol#185)
                - slotAMM.swapETHForCAR{value: amount}() (carbon/SlotV3.sol#447)
                - ! address(uint160(initNode)).send(amount) (carbon/SlotV3.sol#450)
                - address(uint160(initNode)).transfer(amount) (carbon/SlotV3.sol#451)
                - (success,returndata) = target.call{value: weiValue}(data) (Libraries.sol#180-181)
                - ! address(uint160(receiver)).send(ethValue) (carbon/SlotV3.sol#497)
                - address(uint160(receiver)).transfer(address(this).balance) (carbon/SlotV3.sol#498)
        Event emitted after the call(s):
        - Registration(userAddress,referrerAddress,userAddress,referrerAddress) (carbon/SlotV3.sol#202-207)
Reentrancy in CarbonSlotV2.transferMineCARtoAddress(address,uint256) (carbon/SlotV3.sol#436-443):
        External calls:
        - carbonToken.safeTransfer(target,amount) (carbon/SlotV3.sol#438)
        Event emitted after the call(s):
        - CARMined(target,amount) (carbon/SlotV3.sol#441)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Address.isContract(address) (Libraries.sol#102-113) uses assembly
        - INLINE ASM (Libraries.sol#109-111)
Address._functionCallWithValue(address,bytes,uint256,string) (Libraries.sol#171-198) uses assembly
        - INLINE ASM (Libraries.sol#190-193)
CarbonSlotV2.registration(address,address) (carbon/SlotV3.sol#162-208) uses assembly
        - INLINE ASM (carbon/SlotV3.sol#170-172)
```

```
CarbonSlotV2.bytesToAddress(bytes) (carbon/SlotV3.sol#506-514) uses assembly
        - INLINE ASM (carbon/SlotV3.sol#511-513)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Different versions of Solidity is used in :
        - Version used: ['>=0.6.0', '>=0.6.2', '^0.6.0']
        - >=0.6.0 (Context.sol#3)
        - >=0.6.0 (IBEP20.sol#2)
        - >=0.6.2 (IUinswapV2Interface.sol#1)
        - >=0.6.0 (Interfaces.sol#2)
        - ^0.6.0 (Libraries.sol#2)
        - >=0.6.2 (carbon/SlotAMMInterface.sol#2)
        - >=0.6.2 (carbon/SlotV3.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Pragma version>=0.6.0 (Context.sol#3) allows old versions
Pragma version>=0.6.0 (IBEP20.sol#2) allows old versions
Pragma version>=0.6.2 (IUinswapV2Interface.sol#1) allows old versions
Pragma version>=0.6.0 (Interfaces.sol#2) allows old versions
Pragma version^0.6.0 (Libraries.sol#2) allows old versions
Pragma version>=0.6.2 (carbon/SlotAMMInterface.sol#2) allows old versions
Pragma version>=0.6.2 (carbon/SlotV3.sol#2) allows old versions
solc-0.6.2 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (Libraries.sol#115-127):
        - (success) = recipient.call{value: amount}() (Libraries.sol#122)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (Libraries.sol#171-198):
        - (success,returndata) = target.call{value: weiValue}(data) (Libraries.sol#180-181)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (IUinswapV2Interface.sol#36) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (IUinswapV2Interface.sol#37) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (IUinswapV2Interface.sol#54) is not in mixedCase
Function IUniswapV2Router01.WETH() (IUinswapV2Interface.sol#74) is not in mixedCase
Parameter CarbonSlotV2.refreshinitNode(address)._initNode (carbon/SlotV3.sol#456) is not in mixedCase
Parameter CarbonSlotV2.findEthReceiver(address,address,uint8)._from (carbon/SlotV3.sol#466) is not in
mixedCase
Parameter CarbonSlotV2.sendETHDividends(address,address,uint8,uint8,uint256)._from (carbon/SlotV3.sol#483)
is not in mixedCase
Parameter CarbonSlotV2.refreshOpen(bool)._open (carbon/SlotV3.sol#516) is not in mixedCase
Parameter CarbonSlotV2.refresAMM(bool)._isOpen (carbon/SlotV3.sol#520) is not in mixedCase
Parameter CarbonSlotV2.setAMMInterface(address)._ammAddress (carbon/SlotV3.sol#524) is not in mixedCase
Parameter CarbonSlotV2.queryUserEpochInfo(address,uint256)._epochIndex (carbon/SlotV3.sol#559) is not in
mixedCase
Parameter CarbonSlotV2.refreshTokenAddr(address)._addr (carbon/SlotV3.sol#582) is not in mixedCase
Parameter CarbonSlotV2.queryUserTotalMine(address)._addr (carbon/SlotV3.sol#595) is not in mixedCase
Parameter CarbonSlotV2.queryUserTotalReward(address)._addr (carbon/SlotV3.sol#599) is not in mixedCase
Parameter CarbonSlotV2.queryUserX3LevelReward(address,uint8)._addr (carbon/SlotV3.sol#603) is not in
mixedCase
Parameter CarbonSlotV2.queryUserX2LevelReward(address,uint8)._addr (carbon/SlotV3.sol#611) is not in
mixedCase
Parameter CarbonSlotV2.queryUserX3LevelMine(address,uint8)._addr (carbon/SlotV3.sol#619) is not in
mixedCase
Parameter CarbonSlotV2.queryUserContribution(address)._addr (carbon/SlotV3.sol#627) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformity-to-solidity-naming-conventions
INFO:Detectors:
```

```
Reentrancy in CarbonSlotV2.buyNewLevel(uint8) (carbon/SlotV3.sol#210-246):
	External calls:
	- updateMatrixReferrer(msg.sender,activedReferrerAddress,level) (carbon/SlotV3.sol#233)
		- ! address(uint160(initNode)).send(amount) (carbon/SlotV3.sol#450)
		- address(uint160(initNode)).transfer(amount) (carbon/SlotV3.sol#451)
		- ! address(uint160(receiver)).send(ethValue) (carbon/SlotV3.sol#497)
		- address(uint160(receiver)).transfer(address(this).balance) (carbon/SlotV3.sol#498)
	External calls sending eth:
	- updateMatrixReferrer(msg.sender,activedReferrerAddress,level) (carbon/SlotV3.sol#233)
		- slotAMM.swapETHForCAR{value: amount}() (carbon/SlotV3.sol#447)
		- ! address(uint160(initNode)).send(amount) (carbon/SlotV3.sol#450)
		- address(uint160(initNode)).transfer(amount) (carbon/SlotV3.sol#451)
		- (success,returndata) = target.call{value: weiValue}(data) (Libraries.sol#180-181)
		- ! address(uint160(receiver)).send(ethValue) (carbon/SlotV3.sol#497)
		- address(uint160(receiver)).transfer(address(this).balance) (carbon/SlotV3.sol#498)
	State variables written after the call(s):
	- globalInvest = globalInvest + msg.value (carbon/SlotV3.sol#234)
	- userEpochs[msg.sender].push(epoch) (carbon/SlotV3.sol#243)
	Event emitted after the call(s):
	- Upgrade(msg.sender,activedReferrerAddress,1,level) (carbon/SlotV3.sol#245)
Reentrancy in CarbonSlotV2.registration(address,address) (carbon/SlotV3.sol#162-208):
	External calls:
	- updateMatrixReferrer(userAddress,activedReferrer,1) (carbon/SlotV3.sol#185)
		- ! address(uint160(initNode)).send(amount) (carbon/SlotV3.sol#450)
		- address(uint160(initNode)).transfer(amount) (carbon/SlotV3.sol#451)
		- ! address(uint160(receiver)).send(ethValue) (carbon/SlotV3.sol#497)
		- address(uint160(receiver)).transfer(address(this).balance) (carbon/SlotV3.sol#498)
	External calls sending eth:
	- updateMatrixReferrer(userAddress,activedReferrer,1) (carbon/SlotV3.sol#185)
		- slotAMM.swapETHForCAR{value: amount}() (carbon/SlotV3.sol#447)
		- ! address(uint160(initNode)).send(amount) (carbon/SlotV3.sol#450)
		- address(uint160(initNode)).transfer(amount) (carbon/SlotV3.sol#451)
		- (success,returndata) = target.call{value: weiValue}(data) (Libraries.sol#180-181)
		- ! address(uint160(receiver)).send(ethValue) (carbon/SlotV3.sol#497)
		- address(uint160(receiver)).transfer(address(this).balance) (carbon/SlotV3.sol#498)
	State variables written after the call(s):
	- addrRegisted[userAddress] = true (carbon/SlotV3.sol#197)
	- globalInvest = globalInvest + msg.value (carbon/SlotV3.sol#199)
	- globalInvestAddrCount += 1 (carbon/SlotV3.sol#200)
	- userEpochs[userAddress].push(epoch) (carbon/SlotV3.sol#194)
	Event emitted after the call(s):
	- Registration(userAddress,referrerAddress,userAddress,referrerAddress) (carbon/SlotV3.sol#202-207)
Reentrancy in CarbonSlotV2.sendETHDividends(address,address,uint8,uint8,uint256)
(carbon/SlotV3.sol#481-504):
	External calls:
	- ! address(uint160(receiver)).send(ethValue) (carbon/SlotV3.sol#497)
	Event emitted after the call(s):
	- SentExtraEthDividends(_from,receiver,matrix,level) (carbon/SlotV3.sol#502)
Reentrancy in CarbonSlotV2.updateMatrixM2Referrer(address,address,uint8,uint256)
(carbon/SlotV3.sol#326-383):
	External calls:
	- sendETHDividends(referrerAddress,userAddress,2,level,realX2Reward) (carbon/SlotV3.sol#341-347)
		- ! address(uint160(receiver)).send(ethValue) (carbon/SlotV3.sol#497)
		- address(uint160(receiver)).transfer(address(this).balance) (carbon/SlotV3.sol#498)
	State variables written after the call(s):
	- staticPool += restReward (carbon/SlotV3.sol#350)
	- totalPlatformContribute += (restReward * 1000) (carbon/SlotV3.sol#353)
```

```
        - userPlatformContributeMap[referrerAddress] += (restReward * 1000) (carbon/SlotV3.sol#352)
Reentrancy in CarbonSlotV2.updateMatrixM2Referrer(address,address,uint8,uint256)
(carbon/SlotV3.sol#326-383):
        External calls:
        - sendETHDividends(referrerAddress,userAddress,2,level,x2Reward) (carbon/SlotV3.sol#335)
                - ! address(uint160(receiver)).send(ethValue) (carbon/SlotV3.sol#497)
                - address(uint160(receiver)).transfer(address(this).balance) (carbon/SlotV3.sol#498)
        - sendETHDividends(referrerAddress,userAddress,2,level,realX2Reward) (carbon/SlotV3.sol#341-347)
                - ! address(uint160(receiver)).send(ethValue) (carbon/SlotV3.sol#497)
                - address(uint160(receiver)).transfer(address(this).balance) (carbon/SlotV3.sol#498)
        - sendETHDividends(referrerAddress,userAddress,2,level,x2Reward) (carbon/SlotV3.sol#358)
                - ! address(uint160(receiver)).send(ethValue) (carbon/SlotV3.sol#497)
                - address(uint160(receiver)).transfer(address(this).balance) (carbon/SlotV3.sol#498)
        - updateMatrixM2Referrer(referrerAddress,activedReferrerAddress,level,x2Reward)
(carbon/SlotV3.sol#363-368)
                - ! address(uint160(receiver)).send(ethValue) (carbon/SlotV3.sol#497)
                - address(uint160(receiver)).transfer(address(this).balance) (carbon/SlotV3.sol#498)
        State variables written after the call(s):
        - users[referrerAddress].matrix[level].x2ReinvestCount ++ (carbon/SlotV3.sol#380)
        - users[referrerAddress].matrix[level].x2referrals = new address[](0) (carbon/SlotV3.sol#381)
        Event emitted after the call(s):
        - BurnOut(referrerAddress,userAddress,userAddress,2,level) (carbon/SlotV3.sol#376)
Reentrancy in CarbonSlotV2.updateMatrixReferrer(address,address,uint8) (carbon/SlotV3.sol#248-324):
        External calls:
        - sendETHDividends(referrerAddress,userAddress,1,level,x3Reward) (carbon/SlotV3.sol#260)
                - ! address(uint160(receiver)).send(ethValue) (carbon/SlotV3.sol#497)
                - address(uint160(receiver)).transfer(address(this).balance) (carbon/SlotV3.sol#498)
        State variables written after the call(s):
        - staticPool += toStaticPool (carbon/SlotV3.sol#263)
Reentrancy in CarbonSlotV2.updateMatrixReferrer(address,address,uint8) (carbon/SlotV3.sol#248-324):
        External calls:
        - sendETHDividends(referrerAddress,userAddress,1,level,x3Reward_scope_0) (carbon/SlotV3.sol#276)
                - ! address(uint160(receiver)).send(ethValue) (carbon/SlotV3.sol#497)
                - address(uint160(receiver)).transfer(address(this).balance) (carbon/SlotV3.sol#498)
        State variables written after the call(s):
        - staticPool += toStaticPool_scope_1 (carbon/SlotV3.sol#278)
Reentrancy in CarbonSlotV2.updateMatrixReferrer(address,address,uint8) (carbon/SlotV3.sol#248-324):
        External calls:
        - sendETHDividends(referrerAddress,userAddress,1,level,x3Reward_scope_0) (carbon/SlotV3.sol#276)
                - ! address(uint160(receiver)).send(ethValue) (carbon/SlotV3.sol#497)
                - address(uint160(receiver)).transfer(address(this).balance) (carbon/SlotV3.sol#498)
        - updateMatrixM2Referrer(referrerAddress,activedReferrerAddress,level,(levelPrice[level] -
x3Reward_scope_0 - toStaticPool_scope_1)) (carbon/SlotV3.sol#282-287)
                - ! address(uint160(receiver)).send(ethValue) (carbon/SlotV3.sol#497)
                - address(uint160(receiver)).transfer(address(this).balance) (carbon/SlotV3.sol#498)
        State variables written after the call(s):
        - updateMatrixM2Referrer(referrerAddress,activedReferrerAddress,level,(levelPrice[level] -
x3Reward_scope_0 - toStaticPool_scope_1)) (carbon/SlotV3.sol#282-287)
                - matrixLevelReward[receiver][matrix][level] = matrixLevelReward[receiver][matrix][level] +
ethValue (carbon/SlotV3.sol#491-493)
        - updateMatrixM2Referrer(referrerAddress,activedReferrerAddress,level,(levelPrice[level] -
x3Reward_scope_0 - toStaticPool_scope_1)) (carbon/SlotV3.sol#282-287)
                - matrixReward[receiver][matrix] = matrixReward[receiver][matrix] + ethValue
(carbon/SlotV3.sol#494-496)
        - updateMatrixM2Referrer(referrerAddress,activedReferrerAddress,level,(levelPrice[level] -
x3Reward_scope_0 - toStaticPool_scope_1)) (carbon/SlotV3.sol#282-287)
                - staticPool += restReward (carbon/SlotV3.sol#350)
```

```
        - updateMatrixM2Referrer(referrerAddress,activedReferrerAddress,level,(levelPrice[level] -
x3Reward_scope_0 - toStaticPool_scope_1)) (carbon/SlotV3.sol#282-287)
                - totalPlatformContribute += (restReward * 1000) (carbon/SlotV3.sol#353)
        - updateMatrixM2Referrer(referrerAddress,activedReferrerAddress,level,(levelPrice[level] -
x3Reward_scope_0 - toStaticPool_scope_1)) (carbon/SlotV3.sol#282-287)
                - userPlatformContributeMap[referrerAddress] += (restReward * 1000) (carbon/SlotV3.sol#352)
        - updateMatrixM2Referrer(referrerAddress,activedReferrerAddress,level,(levelPrice[level] -
x3Reward_scope_0 - toStaticPool_scope_1)) (carbon/SlotV3.sol#282-287)
                - users[referrerAddress].matrix[level].x2referrals.push(userAddress) (carbon/SlotV3.sol#332)
                - users[referrerAddress].matrix[level].x2ReinvestCount ++ (carbon/SlotV3.sol#380)
                - users[referrerAddress].matrix[level].x2referrals = new address[](0) (carbon/SlotV3.sol#381)
        Event emitted after the call(s):
        - BurnOut(referrerAddress,userAddress,userAddress,2,level) (carbon/SlotV3.sol#376)
                - updateMatrixM2Referrer(referrerAddress,activedReferrerAddress,level,(levelPrice[level] -
x3Reward_scope_0 - toStaticPool_scope_1)) (carbon/SlotV3.sol#282-287)
        - MissedEthReceive(receiver,_from,1,level) (carbon/SlotV3.sol#473)
                - updateMatrixM2Referrer(referrerAddress,activedReferrerAddress,level,(levelPrice[level] -
x3Reward_scope_0 - toStaticPool_scope_1)) (carbon/SlotV3.sol#282-287)
        - SentExtraEthDividends(_from,receiver,matrix,level) (carbon/SlotV3.sol#502)
                - updateMatrixM2Referrer(referrerAddress,activedReferrerAddress,level,(levelPrice[level] -
x3Reward_scope_0 - toStaticPool_scope_1)) (carbon/SlotV3.sol#282-287)
Reentrancy in CarbonSlotV2.updateMatrixReferrer(address,address,uint8) (carbon/SlotV3.sol#248-324):
        External calls:
        - sendETHDividends(referrerAddress,userAddress,1,level,x3Reward_scope_2) (carbon/SlotV3.sol#297)
                - ! address(uint160(receiver)).send(ethValue) (carbon/SlotV3.sol#497)
                - address(uint160(receiver)).transfer(address(this).balance) (carbon/SlotV3.sol#498)
        State variables written after the call(s):
        - addressLevelMine[referrerAddress][level] = addressLevelMine[referrerAddress][level] + mineToken
(carbon/SlotV3.sol#311-313)
        - addressMine[referrerAddress] = addressMine[referrerAddress] + mineToken
(carbon/SlotV3.sol#314-316)
        - staticPool += (levelPrice[level] - x3Reward_scope_2) (carbon/SlotV3.sol#299)
        - users[referrerAddress].matrix[level].blocked = true (carbon/SlotV3.sol#305)
Reentrancy in CarbonSlotV2.updateMatrixReferrer(address,address,uint8) (carbon/SlotV3.sol#248-324):
        External calls:
        - sendETHDividends(referrerAddress,userAddress,1,level,x3Reward_scope_2) (carbon/SlotV3.sol#297)
                - ! address(uint160(receiver)).send(ethValue) (carbon/SlotV3.sol#497)
                - address(uint160(receiver)).transfer(address(this).balance) (carbon/SlotV3.sol#498)
        External calls sending eth:
        - sendETHDividends(referrerAddress,userAddress,1,level,x3Reward_scope_2) (carbon/SlotV3.sol#297)
                - ! address(uint160(receiver)).send(ethValue) (carbon/SlotV3.sol#497)
                - address(uint160(receiver)).transfer(address(this).balance) (carbon/SlotV3.sol#498)
        - transferMineCARtoAddress(referrerAddress,mineToken) (carbon/SlotV3.sol#321)
                - (success,returndata) = target.call{value: weiValue}(data) (Libraries.sol#180-181)
        State variables written after the call(s):
        - transferMineCARtoAddress(referrerAddress,mineToken) (carbon/SlotV3.sol#321)
                - globalMine = globalMine + amount (carbon/SlotV3.sol#439)
        Event emitted after the call(s):
        - CARMined(target,amount) (carbon/SlotV3.sol#441)
                - transferMineCARtoAddress(referrerAddress,mineToken) (carbon/SlotV3.sol#321)
Reentrancy in CarbonSlotV2.withdrawByEpoch(uint256) (carbon/SlotV3.sol#528-549):
        External calls:
        - msg.sender.transfer(canWithdrawAmount) (carbon/SlotV3.sol#546)
        State variables written after the call(s):
        - epoch.totalWithdrawAmount += canWithdrawAmount (carbon/SlotV3.sol#547)
        - epoch.lastWithdrawBlock = block.number (carbon/SlotV3.sol#548)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4
```

```
INFO:Detectors:
CarbonSlotV2.getProfitRate() (carbon/SlotV3.sol#574-576) uses literals with too many digits:
	- 5 * ((globalInvest / 200000000000000000000) + 1) (carbon/SlotV3.sol#575)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
CarbonSlotV2.currentMineRate (carbon/SlotV3.sol#56) should be constant
CarbonSlotV2.mineRate (carbon/SlotV3.sol#54) should be constant
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
owner() should be declared external:
	- Ownable.owner() (Context.sol#35-37)
renounceOwnership() should be declared external:
	- Ownable.renounceOwnership() (Context.sol#54-57)
transferOwnership(address) should be declared external:
	- Ownable.transferOwnership(address) (Context.sol#63-67)
operator() should be declared external:
	- Operator.operator() (Context.sol#84-86)
isOperator() should be declared external:
	- Operator.isOperator() (Context.sol#93-95)
transferOperator(address) should be declared external:
	- Operator.transferOperator(address) (Context.sol#97-99)
usersActiveLevels(address,uint8) should be declared external:
	- CarbonSlotV2.usersActiveLevels(address,uint8) (carbon/SlotV3.sol#406-412)
usersMatrix(address,uint8) should be declared external:
	- CarbonSlotV2.usersMatrix(address,uint8) (carbon/SlotV3.sol#414-434)
getUserEpochLength(address) should be declared external:
	- CarbonSlotV2.getUserEpochLength(address) (carbon/SlotV3.sol#551-557)
queryUserEpochInfo(address,uint256) should be declared external:
	- CarbonSlotV2.queryUserEpochInfo(address,uint256) (carbon/SlotV3.sol#559-572)
queryGlobalMine() should be declared external:
	- CarbonSlotV2.queryGlobalMine() (carbon/SlotV3.sol#587-589)
queryGlobalInvest() should be declared external:
	- CarbonSlotV2.queryGlobalInvest() (carbon/SlotV3.sol#591-593)
queryUserTotalMine(address) should be declared external:
	- CarbonSlotV2.queryUserTotalMine(address) (carbon/SlotV3.sol#595-597)
queryUserTotalReward(address) should be declared external:
	- CarbonSlotV2.queryUserTotalReward(address) (carbon/SlotV3.sol#599-601)
queryUserX3LevelReward(address,uint8) should be declared external:
	- CarbonSlotV2.queryUserX3LevelReward(address,uint8) (carbon/SlotV3.sol#603-609)
queryUserX2LevelReward(address,uint8) should be declared external:
	- CarbonSlotV2.queryUserX2LevelReward(address,uint8) (carbon/SlotV3.sol#611-617)
queryUserX3LevelMine(address,uint8) should be declared external:
	- CarbonSlotV2.queryUserX3LevelMine(address,uint8) (carbon/SlotV3.sol#619-625)
queryUserContribution(address) should be declared external:
	- CarbonSlotV2.queryUserContribution(address) (carbon/SlotV3.sol#627-633)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:carbon/SlotV3.sol analyzed (16 contracts with 46 detectors), 82 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

2. CarbonSlotAMMPancake

```
INFO:Detectors:
Reentrancy in CarbonSlotAMMPancake.swapETHForCAR() (carbon/SlotAMMPancake.sol#48-70):
	External calls:
	- amounts = uniswapRouter.swapExactETHForTokens{value:
ethAmount}(0,paths,carbonBurnedAddress,block.timestamp) (carbon/SlotAMMPancake.sol#62-67)
	Event emitted after the call(s):
	- CARSwappedV2(ethAmount,amounts) (carbon/SlotAMMPancake.sol#68)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Different versions of Solidity is used in :
	- Version used: ['>=0.6.0', '>=0.6.2']
	- >=0.6.0 (Context.sol#3)
	- >=0.6.0 (IBEP20.sol#2)
	- >=0.6.2 (IPancakeInterface.sol#1)
	- >=0.6.2 (carbon/SlotAMMInterface.sol#2)
	- >=0.6.2 (carbon/SlotAMMPancake.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Pragma version>=0.6.0 (Context.sol#3) allows old versions
Pragma version>=0.6.0 (IBEP20.sol#2) allows old versions
Pragma version>=0.6.2 (IPancakeInterface.sol#1) allows old versions
Pragma version>=0.6.2 (carbon/SlotAMMInterface.sol#2) allows old versions
Pragma version>=0.6.2 (carbon/SlotAMMPancake.sol#2) allows old versions
solc-0.6.2 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Function IPancakeRouter01.WETH() (IPancakeInterface.sol#22) is not in mixedCase
Function IPancakePair.DOMAIN_SEPARATOR() (IPancakeInterface.sol#170) is not in mixedCase
Function IPancakePair.PERMIT_TYPEHASH() (IPancakeInterface.sol#171) is not in mixedCase
Function IPancakePair.MINIMUM_LIQUIDITY() (IPancakeInterface.sol#188) is not in mixedCase
Parameter CarbonSlotAMMPancake.refreshOpen(bool)._open (carbon/SlotAMMPancake.sol#40) is not in
mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformity-to-solidity-naming-conventions
INFO:Detectors:
CarbonSlotAMMPancake.constructor(address,address) (carbon/SlotAMMPancake.sol#25-38) uses literals with too
many digits:
	- _increaseApprove(99999999999900000000000000000000) (carbon/SlotAMMPancake.sol#37)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
owner() should be declared external:
	- Ownable.owner() (Context.sol#35-37)
renounceOwnership() should be declared external:
	- Ownable.renounceOwnership() (Context.sol#54-57)
transferOwnership(address) should be declared external:
	- Ownable.transferOwnership(address) (Context.sol#63-67)
operator() should be declared external:
	- Operator.operator() (Context.sol#84-86)
isOperator() should be declared external:
	- Operator.isOperator() (Context.sol#93-95)
transferOperator(address) should be declared external:
	- Operator.transferOperator(address) (Context.sol#97-99)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:carbon/SlotAMMPancake.sol analyzed (10 contracts with 46 detectors), 20 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

3. CarbonToken

```
INFO:Detectors:
BEP20.allowance(address,address).owner (tokens/BEP20Burnable.sol#62) shadows:
	- Ownable.owner() (Context.sol#35-37) (function)
BEP20._approve(address,address,uint256).owner (tokens/BEP20Burnable.sol#171) shadows:
	- Ownable.owner() (Context.sol#35-37) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Address.isContract(address) (Libraries.sol#102-113) uses assembly
	- INLINE ASM (Libraries.sol#109-111)
Address._functionCallWithValue(address,bytes,uint256,string) (Libraries.sol#171-198) uses assembly
	- INLINE ASM (Libraries.sol#190-193)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Different versions of Solidity is used in :
	- Version used: ['>=0.6.0', '^0.6.0']
	- >=0.6.0 (Context.sol#3)
	- >=0.6.0 (IBEP20.sol#2)
	- >=0.6.0 (Interfaces.sol#2)
	- ^0.6.0 (Libraries.sol#2)
	- >=0.6.0 (tokens/BEP20Burnable.sol#2)
	- >=0.6.0 (tokens/CarbonToken.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Pragma version>=0.6.0 (Context.sol#3) allows old versions
Pragma version>=0.6.0 (IBEP20.sol#2) allows old versions
Pragma version>=0.6.0 (Interfaces.sol#2) allows old versions
Pragma version^0.6.0 (Libraries.sol#2) allows old versions
Pragma version>=0.6.0 (tokens/BEP20Burnable.sol#2) allows old versions
Pragma version>=0.6.0 (tokens/CarbonToken.sol#2) allows old versions
solc-0.6.2 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (Libraries.sol#115-127):
	- (success) = recipient.call{value: amount}() (Libraries.sol#122)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (Libraries.sol#171-198):
	- (success,returndata) = target.call{value: weiValue}(data) (Libraries.sol#180-181)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
renounceOwnership() should be declared external:
	- Ownable.renounceOwnership() (Context.sol#54-57)
transferOwnership(address) should be declared external:
	- Ownable.transferOwnership(address) (Context.sol#63-67)
operator() should be declared external:
	- Operator.operator() (Context.sol#84-86)
isOperator() should be declared external:
	- Operator.isOperator() (Context.sol#93-95)
transferOperator(address) should be declared external:
	- Operator.transferOperator(address) (Context.sol#97-99)
getOwner() should be declared external:
	- BEP20.getOwner() (tokens/BEP20Burnable.sol#28-30)
name() should be declared external:
	- BEP20.name() (tokens/BEP20Burnable.sol#32-34)
symbol() should be declared external:
	- BEP20.symbol() (tokens/BEP20Burnable.sol#36-38)
decimals() should be declared external:
```

```
        - BEP20.decimals() (tokens/BEP20Burnable.sol#40-42)
totalSupply() should be declared external:
        - BEP20.totalSupply() (tokens/BEP20Burnable.sol#44-46)
transfer(address,uint256) should be declared external:
        - BEP20.transfer(address,uint256) (tokens/BEP20Burnable.sol#52-60)
increaseAllowance(address,uint256) should be declared external:
        - BEP20.increaseAllowance(address,uint256) (tokens/BEP20Burnable.sol#100-111)
decreaseAllowance(address,uint256) should be declared external:
        - BEP20.decreaseAllowance(address,uint256) (tokens/BEP20Burnable.sol#113-127)
mint(address,uint256) should be declared external:
        - CarbonToken.mint(address,uint256) (tokens/CarbonToken.sol#18-28)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:tokens/CarbonToken.sol analyzed (12 contracts with 46 detectors), 28 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Concluding Remarks

While conducting the audits of the Carbon smart contract, it was observed that the contracts contain Medium and Low severity issues, along with several areas of recommendations.

Our auditors suggest that Medium and Low severity issues should be resolved by Carbon developers. Resolving the areas of recommendations are up to the team's discretion. The recommendations given will improve the operations of the smart contract.

Disclaimer

ImmuneBytes's audit does not provide a security or correctness guarantee of the audited smart contract. Securing smart contracts is a multistep process, therefore running a bug bounty program as a complement to this audit is strongly recommended.

Our team does not endorse the Carbon platform or its product neither this audit is investment advice.
Notes:

- Please make sure contracts deployed on the mainnet are the ones audited.
- Check for the code refactor by the team on critical issues.

ImmuneBytes Pvt Ltd.