QuillAudits

Audit Report
May, 2022

For

Table of Content

Executive Summary

Project Name	Carpe Diem Savings
Overview	Carpe Diem Savings is a blockchain based decentralized savings account for your cryptocurrency assets. Carpe Diem Savings is a decentralised framework that combines different types of savings accounts into one model and applies them to existing cryptocurrencies using blockchain-based smart contract technology.
Timeline	6th April, 2022 to 9th May, 2022
Method	Manual Review, Functional Testing, Automated Testing etc.
Scope of Audit	The scope of this audit was to analyse Carpe Diem Savings codebase for quality, security, and correctness.
Codebase	*https://github.com/CarpeDiemSavings/contracts/tree/main/contracts*
Commit hash	9aaf7559d66756b8d6fd11a06929a2a5977dd8f8
Fixed In	66e9a0801eaa4dacc06479aca67b33cb93fb755c

High
Medium
Low
Informational

	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	3	0	0
Partially Resolved Issues	0	0	0	1
Resolved Issues	0	1	3	1

Types of Severities

High

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

Medium

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

Low

Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Types of Issues

Open

Security vulnerabilities identified that must be resolved and are currently unresolved.

Resolved

These are the issues identified in the initial audit and have been successfully fixed.

Acknowledged

Vulnerabilities which have been acknowledged but are yet to be resolved.

Partially Resolved

Considerable efforts have been invested to reduce the risk/impact of the security issue, but are not completely resolved.

Checked Vulnerabilities

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- Exception Disorder
- Gasless Send
- Use of tx.origin
- Compiler version not fixed
- Address hardcoded
- Divide before multiply
- Integer overflow/underflow
- Dangerous strict equalities
- Tautology or contradiction
- Return values of low-level calls
- Missing Zero Address Validation
- Private modifier
- Revert/require functions
- Using block.timestamp
- Multiple Sends
- Using SHA3
- Using suicide
- Using throw
- Using inline assembly

Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis

In this step, we have analysed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis

Static analysis of smart contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis

Manual analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analysed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption

In this step, we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit

Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis.

Manual Testing

High Severity Issues

No issues found

Medium Severity Issues

1. state variable are written after the external call

If the token safeTransferFrom or safeTransfer is malicious then the reentrency can be created in Carpediem.upgradeStake, Carpediem.distributePenalty, Carpediem.deposit.

We recommend to follow CEI pattern to remove reentrency or use Openzepplin ReentrancyGuard.

Status

Fixed

2. Perform multiplication on the result of division

In Carpediem._getPenalty on line 343 and 346 the solidity integer division might truncate. As a result, performing multiplication before division can sometimes avoid loss of precision and raise various issues.

It is always recommended to usually re-arrange arithmetic to perform multiplication before division.

Status

Acknowledged

3. block.timestamp has been used on various places

In the case of block.timestamp, developers often attempt to use it to trigger time-dependent events. As Ethereum is decentralized, nodes can synchronize time only to some degree. Moreover, malicious miners can alter the timestamp of their blocks, especially if they can gain advantages by doing so. However, miners can't set a timestamp smaller than the previous one (otherwise the block will be rejected), nor can they set the timestamp too far ahead in the future. Taking all of the above into consideration, developers can't rely on the preciseness of the provided timestamp.

It is always recommended to follow the 15-second rule allows one to achieve a more reliable estimate of time.

Reference: *SWC116*

Status

Acknowledged

4. No rescueFund method is implemented in all the contracts

There should be a rescueFund method to rescue the tokens funds which are mistakenly sent and are not part of the contracts.

The behaviour of the rescueFund should only be operated by the owner and nobody else.

Status

Acknowledged

Low Severity Issues

5. Used locked pragma version

The pragma versions used in the contract are not locked. Consider using the latest versions among 0.8.13 for deploying the contracts and libraries, as it does not compile for any other version and can be confusing for a developer. Solidity source files indicate the versions of the compiler they can be compiled with.

```
pragma solidity ^0.8.0; // bad: compiles between 0.8.0 and 0.8.10
pragma solidity 0.8.0; // good : compiles w 0.8.0 only but not the latest version
pragma solidity 0.8.10; // best: compiles w 0.8.10
```

Status

Fixed

6. Owner should be multisig

We recommend to use multisig account address (gnosis-safe) for owner such that the pool creating is not been malicious in future and the decentralization is achieved in the system.

Status

Fixed

7. Contracts Have Non-indexed Events

No parameters are indexed in the events of most contracts.
For example, StakeUpgraded, Withdraw and StakeRemoved events of Carpediem don't index the depositor/who of the contract.

It is recommended to index the relevant event parameters to allow integrators and dApps to quickly search for these and simplify UIs.

Status

Fixed

Informational Issues

8. Gas optimizations

Carpediem.sol

1. On line no 87 and 88 the reinitialize of variable value to 0 is done which is a gas wastage. As by default, the value of uint256 variable is 0. We recommend removing line 87 and 88.
2. On line no 252, 52, 48 no need to calculate the length of address as it's a static initialization of length 3. We can directly use 3.
3. Most of the constants can be made private instead of public.

CarpediemFacory.sol

4. On line no 49 and 52 the calculation of length is not needed as it's static , and we can directly use 5 and 3.
5. The constant percentBase should be private instead of public.

Status

Partially Fixed

9. Netspec comments are missing

We recommend to follow *netspec* spec doc for more readability and better understanding of code.

Status

Fixed

Binance Testnet Contract

0x28e8dA57Acc2D99cE6B92297e6cb979eE158DcbD
0xc53b4Ae5fB089ff273b76DC1Ba598c8d1c04C4c6

Automated Tests

Slither

```
        - extraShares = _buyShares(_amount) (CarpediemFactory_flat.sol#710)
                - (success,returndata) = target.call{value: value}(data) (CarpediemFactory_flat.sol#199)
        State variables written after the call(s):
        - stakes[msg.sender][_stakeId] = StakeInfo(stakeInfo.amount + _amount,stakeInfo.duration,stakeInfo.startTs,stakeInfo.shares + extraShares,stakeInfo.lBonusShares + lBonusShares,bBonusShares,lambda,getReward(
msg.sender,_stakeId)) (CarpediemFactory_flat.sol#727-736)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities

CarpeDiem._getPenalty(address,uint256,uint256) (CarpediemFactory_flat.sol#900-919) performs a multiplication on the result of a division:
        -lateWeeks = (blockTimestamp - (startTs + duration)) / WEEK (CarpediemFactory_flat.sol#911)
        -(_reward * PENALTY_PERCENT_PER_WEEK * lateWeeks) / PERCENT_BASE (CarpediemFactory_flat.sol#913-914)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

Reentrancy in CarpeDiem.distributePenalty() (CarpediemFactory_flat.sol#815-836):
        External calls:
        - poolToken.safeTransfer(DEAD_WALLET,(_commissionAccumulator * burnPercent) / (PERCENT_BASE - stakersPercent)) (CarpediemFactory_flat.sol#829-833)
        State variables written after the call(s):
        - commissionAccumulator = 0 (CarpediemFactory_flat.sol#835)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

Reentrancy in CarpeDiem.deposit(uint256,uint32) (CarpediemFactory_flat.sol#675-699):
        External calls:
        - shares = _buyShares(_amount) (CarpediemFactory_flat.sol#679)
                - token.safeTransferFrom(msg.sender,address(this),_amount) (CarpediemFactory_flat.sol#870)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (CarpediemFactory_flat.sol#462)
                - (success,returndata) = target.call{value: value}(data) (CarpediemFactory_flat.sol#199)
        External calls sending eth:
        - shares = _buyShares(_amount) (CarpediemFactory_flat.sol#679)
                - (success,returndata) = target.call{value: value}(data) (CarpediemFactory_flat.sol#199)
        State variables written after the call(s):
        - stakes[msg.sender].push(StakeInfo(_amount,_duration,uint32(block.timestamp),shares,lBonusShares,bBonusShares,lambda,0)) (CarpediemFactory_flat.sol#685-696)
        - totalShares += shares + lBonusShares + bBonusShares (CarpediemFactory_flat.sol#684)
Reentrancy in CarpeDiem.upgradeStake(uint256,uint256) (CarpediemFactory_flat.sol#701-744):
        External calls:
        - extraShares = _buyShares(_amount) (CarpediemFactory_flat.sol#710)
                - token.safeTransferFrom(msg.sender,address(this),_amount) (CarpediemFactory_flat.sol#870)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (CarpediemFactory_flat.sol#462)
                - (success,returndata) = target.call{value: value}(data) (CarpediemFactory_flat.sol#199)
        External calls sending eth:
        - extraShares = _buyShares(_amount) (CarpediemFactory_flat.sol#710)
                - (success,returndata) = target.call{value: value}(data) (CarpediemFactory_flat.sol#199)
        State variables written after the call(s):
        - totalShares += (extraShares + bBonusShares + lBonusShares - stakeInfo.bBonusShares) (CarpediemFactory_flat.sol#724)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in CarpeDiem.deposit(uint256,uint32) (CarpediemFactory_flat.sol#675-699):
        External calls:
        - shares = _buyShares(_amount) (CarpediemFactory_flat.sol#679)
                - token.safeTransferFrom(msg.sender,address(this),_amount) (CarpediemFactory_flat.sol#870)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (CarpediemFactory_flat.sol#462)
                - (success,returndata) = target.call{value: value}(data) (CarpediemFactory_flat.sol#199)
        External calls sending eth:
        - shares = _buyShares(_amount) (CarpediemFactory_flat.sol#679)
                - (success,returndata) = target.call{value: value}(data) (CarpediemFactory_flat.sol#199)
        Event emitted after the call(s):
```

```
Reentrancy in CarpeDiem.deposit(uint256,uint32) (CarpediemFactory_flat.sol#675-699):
	External calls:
	- shares = _buyShares(_amount) (CarpediemFactory_flat.sol#679)
		- token.safeTransferFrom(msg.sender,address(this),_amount) (CarpediemFactory_flat.sol#870)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (CarpediemFactory_flat.sol#462)
		- (success,returndata) = target.call{value: value}(data) (CarpediemFactory_flat.sol#199)
	External calls sending eth:
	- shares = _buyShares(_amount) (CarpediemFactory_flat.sol#679)
		- (success,returndata) = target.call{value: value}(data) (CarpediemFactory_flat.sol#199)
	Event emitted after the call(s):
	- Deposit(msg.sender,stakes[msg.sender].length - 1,_amount,_duration) (CarpediemFactory_flat.sol#690)
	- SharesCharged(totalShares,totalShares + shares + lBonusShares + bBonusShares) (CarpediemFactory_flat.sol#683)
Reentrancy in CarpeDiem.removeDeadStake(address,uint256) (CarpediemFactory_flat.sol#776-813):
	External calls:
	- token.safeTransfer(_user,stakeInfo.amount) (CarpediemFactory_flat.sol#811)
	Event emitted after the call(s):
	- StakeRemoved(_user,_stakeId,stakeInfo.amount) (CarpediemFactory_flat.sol#812)
Reentrancy in CarpeDiem.upgradeStake(uint256,uint256) (CarpediemFactory_flat.sol#701-744):
	External calls:
	- extraShares = _buyShares(_amount) (CarpediemFactory_flat.sol#710)
		- token.safeTransferFrom(msg.sender,address(this),_amount) (CarpediemFactory_flat.sol#870)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (CarpediemFactory_flat.sol#462)
		- (success,returndata) = target.call{value: value}(data) (CarpediemFactory_flat.sol#199)
	External calls sending eth:
	- extraShares = _buyShares(_amount) (CarpediemFactory_flat.sol#710)
		- (success,returndata) = target.call{value: value}(data) (CarpediemFactory_flat.sol#199)
	Event emitted after the call(s):
	- SharesCharged(totalShares,totalShares + extraShares + bBonusShares + lBonusShares - stakeInfo.bBonusShares) (CarpediemFactory_flat.sol#722-723)
	- StakeUpgraded(msg.sender,_stakeId,_amount,stakeInfo.startTs + stakeInfo.duration - blockTimestamp) (CarpediemFactory_flat.sol#738-743)
Reentrancy in CarpeDiem.withdraw(uint256) (CarpediemFactory_flat.sol#746-774):
	External calls:
	- token.safeTransfer(msg.sender,stakeInfo.amount + reward - penalty) (CarpediemFactory_flat.sol#772)
	Event emitted after the call(s):
	- Withdraw(msg.sender,_stakeId,stakeInfo.amount,reward,penalty) (CarpediemFactory_flat.sol#773)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

CarpeDiem.upgradeStake(uint256,uint256) (CarpediemFactory_flat.sol#701-744) uses timestamp for comparisons
	Dangerous comparisons:
	- require(bool,string)(block.timestamp < stakeInfo.duration + stakeInfo.startTs,stake matured) (CarpediemFactory_flat.sol#706-709)
CarpeDiem.withdraw(uint256) (CarpediemFactory_flat.sol#746-774) uses timestamp for comparisons
	Dangerous comparisons:
	- totalShares == 0 (CarpediemFactory_flat.sol#763)
CarpeDiem.removeDeadStake(address,uint256) (CarpediemFactory_flat.sol#776-813) uses timestamp for comparisons
	Dangerous comparisons:
	- require(bool,string)(uint32(block.timestamp) >= stakeInfo.startTs + stakeInfo.duration + 31536000,stakeAlive) (CarpediemFactory_flat.sol#780-783)
	- totalShares == 0 (CarpediemFactory_flat.sol#801)
CarpeDiem.getReward(address,uint256) (CarpediemFactory_flat.sol#847-863) uses timestamp for comparisons
	Dangerous comparisons:
	- poolLambda - stakeInfo.lastLambda > 0 (CarpediemFactory_flat.sol#854)
CarpeDiem._getBonusL(uint256,uint32) (CarpediemFactory_flat.sol#887-898) uses timestamp for comparisons
	Dangerous comparisons:
	- _duration < poolLBonus (CarpediemFactory_flat.sol#893)
CarpeDiem._getPenalty(address,uint256,uint256) (CarpediemFactory_flat.sol#900-919) uses timestamp for comparisons
	Dangerous comparisons:
	- startTs + duration <= blockTimestamp (CarpediemFactory_flat.sol#909)
	- startTs + duration + WEEK > blockTimestamp (CarpediemFactory_flat.sol#910)
	- lateWeeks >= MAX_PENALTY_DURATION (CarpediemFactory_flat.sol#912)
CarpeDiem._changeSharesPrice(uint256,uint256) (CarpediemFactory_flat.sol#921-930) uses timestamp for comparisons
	Dangerous comparisons:
	- _profit > (oldPrice * _shares) / (MULTIPLIER) (CarpediemFactory_flat.sol#923)
	- newPrice > MAX_PRICE (CarpediemFactory_flat.sol#926)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Address.isContract(address) (CarpediemFactory_flat.sol#94-104) uses assembly
	- INLINE ASM (CarpediemFactory_flat.sol#100-102)
Address.verifyCallResult(bool,bytes,string) (CarpediemFactory_flat.sol#263-283) uses assembly
	- INLINE ASM (CarpediemFactory_flat.sol#275-278)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
	- Version used: ['0.8.7', '^0.8.0']
	- ^0.8.0 (CarpediemFactory_flat.sol#6)
	- ^0.8.0 (CarpediemFactory_flat.sol#71)
	- ^0.8.0 (CarpediemFactory_flat.sol#290)
	- ^0.8.0 (CarpediemFactory_flat.sol#374)
	- ^0.8.0 (CarpediemFactory_flat.sol#474)
	- ^0.8.0 (CarpediemFactory_flat.sol#500)
	- ^0.8.0 (CarpediemFactory_flat.sol#571)
	- 0.8.7 (CarpediemFactory_flat.sol#936)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Address.functionCall(address,bytes) (CarpediemFactory_flat.sol#147-149) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (CarpediemFactory_flat.sol#176-182) is never used and should be removed
Address.functionDelegateCall(address,bytes) (CarpediemFactory_flat.sol#236-238) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (CarpediemFactory_flat.sol#246-255) is never used and should be removed
Address.functionStaticCall(address,bytes) (CarpediemFactory_flat.sol#209-211) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (CarpediemFactory_flat.sol#219-228) is never used and should be removed
Address.sendValue(address,uint256) (CarpediemFactory_flat.sol#122-127) is never used and should be removed
Context._msgData() (CarpediemFactory_flat.sol#491-493) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (CarpediemFactory_flat.sol#414-427) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (CarpediemFactory_flat.sol#438-449) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (CarpediemFactory_flat.sol#429-436) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (CarpediemFactory_flat.sol#6) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (CarpediemFactory_flat.sol#71) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (CarpediemFactory_flat.sol#290) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (CarpediemFactory_flat.sol#374) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (CarpediemFactory_flat.sol#474) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (CarpediemFactory_flat.sol#500) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (CarpediemFactory_flat.sol#571) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.7 (CarpediemFactory_flat.sol#936) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.7 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (CarpediemFactory_flat.sol#122-127):
	- (success) = recipient.call{value: amount}() (CarpediemFactory_flat.sol#125)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (CarpediemFactory_flat.sol#190-201):
	- (success,returndata) = target.call{value: value}(data) (CarpediemFactory_flat.sol#199)
Low level call in Address.functionStaticCall(address,bytes,string) (CarpediemFactory_flat.sol#219-228):
	- (success,returndata) = target.staticcall(data) (CarpediemFactory_flat.sol#226)
Low level call in Address.functionDelegateCall(address,bytes,string) (CarpediemFactory_flat.sol#246-255):
	- (success,returndata) = target.delegatecall(data) (CarpediemFactory_flat.sol#253)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter CarpeDiem.getStakesLength(address)._staker (CarpediemFactory_flat.sol#671) is not in mixedCase
Parameter CarpeDiem.deposit(uint256,uint32)._amount (CarpediemFactory_flat.sol#675) is not in mixedCase
Parameter CarpeDiem.deposit(uint256,uint32)._duration (CarpediemFactory_flat.sol#675) is not in mixedCase
Parameter CarpeDiem.upgradeStake(uint256,uint256)._stakeId (CarpediemFactory_flat.sol#701) is not in mixedCase
Parameter CarpeDiem.upgradeStake(uint256,uint256)._amount (CarpediemFactory_flat.sol#701) is not in mixedCase
Parameter CarpeDiem.withdraw(uint256)._stakeId (CarpediemFactory_flat.sol#746) is not in mixedCase
Parameter CarpeDiem.removeDeadStake(address,uint256)._user (CarpediemFactory_flat.sol#776) is not in mixedCase
Parameter CarpeDiem.removeDeadStake(address,uint256)._stakeId (CarpediemFactory_flat.sol#776) is not in mixedCase
Parameter CarpeDiem.getPenalty(address,uint256)._user (CarpediemFactory_flat.sol#838) is not in mixedCase
Parameter CarpeDiem.getPenalty(address,uint256)._stakeId (CarpediemFactory_flat.sol#838) is not in mixedCase
```

```
Pragma version^0.8.0 (CarpediemFactory_flat.sol#290) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (CarpediemFactory_flat.sol#374) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (CarpediemFactory_flat.sol#474) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (CarpediemFactory_flat.sol#500) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (CarpediemFactory_flat.sol#571) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version0.8.7 (CarpediemFactory_flat.sol#936) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.7 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (CarpediemFactory_flat.sol#122-127):
        - (success) = recipient.call{value: amount}() (CarpediemFactory_flat.sol#125)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (CarpediemFactory_flat.sol#190-201):
        - (success,returndata) = target.call{value: value}(data) (CarpediemFactory_flat.sol#199)
Low level call in Address.functionStaticCall(address,bytes,string) (CarpediemFactory_flat.sol#219-228):
        - (success,returndata) = target.staticcall(data) (CarpediemFactory_flat.sol#226)
Low level call in Address.functionDelegateCall(address,bytes,string) (CarpediemFactory_flat.sol#246-255):
        - (success,returndata) = target.delegatecall(data) (CarpediemFactory_flat.sol#253)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter CarpeDiem.getStakesLength(address)._staker (CarpediemFactory_flat.sol#671) is not in mixedCase
Parameter CarpeDiem.deposit(uint256,uint32)._amount (CarpediemFactory_flat.sol#675) is not in mixedCase
Parameter CarpeDiem.deposit(uint256,uint32)._duration (CarpediemFactory_flat.sol#675) is not in mixedCase
Parameter CarpeDiem.upgradeStake(uint256,uint256)._stakeId (CarpediemFactory_flat.sol#701) is not in mixedCase
Parameter CarpeDiem.upgradeStake(uint256,uint256)._amount (CarpediemFactory_flat.sol#701) is not in mixedCase
Parameter CarpeDiem.withdraw(uint256)._stakeId (CarpediemFactory_flat.sol#746) is not in mixedCase
Parameter CarpeDiem.removeDeadStake(address,uint256)._user (CarpediemFactory_flat.sol#776) is not in mixedCase
Parameter CarpeDiem.removeDeadStake(address,uint256)._stakeId (CarpediemFactory_flat.sol#776) is not in mixedCase
Parameter CarpeDiem.getPenalty(address,uint256)._user (CarpediemFactory_flat.sol#838) is not in mixedCase
Parameter CarpeDiem.getPenalty(address,uint256)._stakeId (CarpediemFactory_flat.sol#838) is not in mixedCase
Parameter CarpeDiem.getReward(address,uint256)._user (CarpediemFactory_flat.sol#847) is not in mixedCase
Parameter CarpeDiem.getReward(address,uint256)._stakeId (CarpediemFactory_flat.sol#847) is not in mixedCase
Parameter CarpediemFactory.poolsByTokenLength(address)._token (CarpediemFactory_flat.sol#962) is not in mixedCase
Parameter CarpediemFactory.createPool(address,uint256,uint256,uint256,uint256,uint256,uint16[5],address[3])._token (CarpediemFactory_flat.sol#967) is not in mixedCase
Parameter CarpediemFactory.createPool(address,uint256,uint256,uint256,uint256,uint256,uint16[5],address[3])._initialPrice (CarpediemFactory_flat.sol#968) is not in mixedCase
Parameter CarpediemFactory.createPool(address,uint256,uint256,uint256,uint256,uint256,uint16[5],address[3])._bBonusAmount (CarpediemFactory_flat.sol#969) is not in mixedCase
Parameter CarpediemFactory.createPool(address,uint256,uint256,uint256,uint256,uint256,uint16[5],address[3])._lBonusPeriod (CarpediemFactory_flat.sol#970) is not in mixedCase
Parameter CarpediemFactory.createPool(address,uint256,uint256,uint256,uint256,uint256,uint16[5],address[3])._bBonusMaxPercent (CarpediemFactory_flat.sol#971) is not in mixedCase
Parameter CarpediemFactory.createPool(address,uint256,uint256,uint256,uint256,uint256,uint16[5],address[3])._lBonusMaxPercent (CarpediemFactory_flat.sol#972) is not in mixedCase
Parameter CarpediemFactory.createPool(address,uint256,uint256,uint256,uint256,uint256,uint16[5],address[3])._distributionPercents (CarpediemFactory_flat.sol#973) is not in mixedCase
Parameter CarpediemFactory.createPool(address,uint256,uint256,uint256,uint256,uint256,uint16[5],address[3])._distributionAddresses (CarpediemFactory_flat.sol#974) is not in mixedCase
Constant CarpediemFactory.percentBase (CarpediemFactory_flat.sol#946) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable CarpeDiem.bBonusMaxPercent (CarpediemFactory_flat.sol#594) is too similar to CarpeDiem.lBonusMaxPercent (CarpediemFactory_flat.sol#595)
Variable CarpeDiem.upgradeStake(uint256,uint256).bBonusShares (CarpediemFactory_flat.sol#717-720) is too similar to CarpeDiem.deposit(uint256,uint32).lBonusShares (CarpediemFactory_flat.sol#680)
Variable CarpeDiem.deposit(uint256,uint32).bBonusShares (CarpediemFactory_flat.sol#681) is too similar to CarpeDiem.deposit(uint256,uint32).lBonusShares (CarpediemFactory_flat.sol#680)
Variable CarpeDiem.deposit(uint256,uint32).bBonusShares (CarpediemFactory_flat.sol#681) is too similar to CarpeDiem.upgradeStake(uint256,uint256).lBonusShares (CarpediemFactory_flat.sol#713-716)
Variable CarpeDiem.upgradeStake(uint256,uint256).bBonusShares (CarpediemFactory_flat.sol#717-720) is too similar to CarpeDiem.upgradeStake(uint256,uint256).lBonusShares (CarpediemFactory_flat.sol#713-716)
Variable CarpediemFactory.createPool(address,uint256,uint256,uint256,uint256,uint256,uint16[5],address[3])._bBonusMaxPercent (CarpediemFactory_flat.sol#971) is too similar to CarpediemFactory.createPool(address,uin
t256,uint256,uint256,uint256,uint256,uint16[5],address[3])._lBonusMaxPercent (CarpediemFactory_flat.sol#972)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

CarpeDiem.slitherConstructorConstantVariables() (CarpediemFactory_flat.sol#579-931) uses literals with too many digits:
        - DEAD_WALLET = 0x000000000000000000000000000000000000dEaD (CarpediemFactory_flat.sol#582)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (CarpediemFactory_flat.sol#549-551)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (CarpediemFactory_flat.sol#557-560)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
CarpediemFactory_flat.sol analyzed (8 contracts with 75 detectors), 74 result(s) found
ethsec@be29ddd8d48f:/code/contracts$
```

```
ethsec@be29ddd8d48f:/code/contracts$ slither Carpediem_flat.sol

Reentrancy in CarpeDiem.upgradeStake(uint256,uint256) (Carpediem_flat.sol#702-745):
        External calls:
        - extraShares = _buyShares(_amount) (Carpediem_flat.sol#711)
                - token.safeTransferFrom(msg.sender,address(this),_amount) (Carpediem_flat.sol#871)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (Carpediem_flat.sol#462)
                - (success,returndata) = target.call{value: value}(data) (Carpediem_flat.sol#199)
        External calls sending eth:
        - extraShares = _buyShares(_amount) (Carpediem_flat.sol#711)
                - (success,returndata) = target.call{value: value}(data) (Carpediem_flat.sol#199)
        State variables written after the call(s):
        - stakes[msg.sender][_stakeId] = StakeInfo(stakeInfo.amount + _amount,stakeInfo.duration,stakeInfo.startTs,stakeInfo.shares + extraShares,stakeInfo.lBonusShares + lBonusShares,bBonusShares,lambda,getReward(
msg.sender,_stakeId)) (Carpediem_flat.sol#728-737)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities

CarpeDiem._getPenalty(address,uint256,uint256) (Carpediem_flat.sol#901-920) performs a multiplication on the result of a division:
        -lateWeeks = (blockTimestamp - (startTs + duration)) / WEEK (Carpediem_flat.sol#912)
        -(_reward * PENALTY_PERCENT_PER_WEEK * lateWeeks) / PERCENT_BASE (Carpediem_flat.sol#914-915)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

Reentrancy in CarpeDiem.distributePenalty() (Carpediem_flat.sol#816-837):
        External calls:
        - poolToken.safeTransfer(DEAD_WALLET,(_commissionAccumulator * burnPercent) / (PERCENT_BASE - stakersPercent)) (Carpediem_flat.sol#830-834)
        State variables written after the call(s):
        - commissionAccumulator = 0 (Carpediem_flat.sol#836)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

Reentrancy in CarpeDiem.deposit(uint256,uint32) (Carpediem_flat.sol#676-700):
        External calls:
        - shares = _buyShares(_amount) (Carpediem_flat.sol#680)
                - token.safeTransferFrom(msg.sender,address(this),_amount) (Carpediem_flat.sol#871)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (Carpediem_flat.sol#462)
                - (success,returndata) = target.call{value: value}(data) (Carpediem_flat.sol#199)
        External calls sending eth:
        - shares = _buyShares(_amount) (Carpediem_flat.sol#680)
                - (success,returndata) = target.call{value: value}(data) (Carpediem_flat.sol#199)
        State variables written after the call(s):
        - stakes[msg.sender].push(StakeInfo(_amount,_duration,uint32(block.timestamp),shares,lBonusShares,bBonusShares,lambda,0)) (Carpediem_flat.sol#686-697)
        - totalShares += shares + lBonusShares + bBonusShares (Carpediem_flat.sol#685)
Reentrancy in CarpeDiem.upgradeStake(uint256,uint256) (Carpediem_flat.sol#702-745):
        External calls:
        - extraShares = _buyShares(_amount) (Carpediem_flat.sol#711)
                - token.safeTransferFrom(msg.sender,address(this),_amount) (Carpediem_flat.sol#871)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (Carpediem_flat.sol#462)
                - (success,returndata) = target.call{value: value}(data) (Carpediem_flat.sol#199)
        External calls sending eth:
        - extraShares = _buyShares(_amount) (Carpediem_flat.sol#711)
                - (success,returndata) = target.call{value: value}(data) (Carpediem_flat.sol#199)
        State variables written after the call(s):
        - totalShares += (extraShares + bBonusShares + lBonusShares - stakeInfo.bBonusShares) (Carpediem_flat.sol#725)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in CarpeDiem.deposit(uint256,uint32) (Carpediem_flat.sol#676-700):
        External calls:
        - shares = _buyShares(_amount) (Carpediem_flat.sol#680)
                - token.safeTransferFrom(msg.sender,address(this),_amount) (Carpediem_flat.sol#871)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (Carpediem_flat.sol#462)
                - (success,returndata) = target.call{value: value}(data) (Carpediem_flat.sol#199)
        External calls sending eth:
        - shares = _buyShares(_amount) (Carpediem_flat.sol#680)
                - (success,returndata) = target.call{value: value}(data) (Carpediem_flat.sol#199)
        Event emitted after the call(s):
```

```
Reentrancy in CarpeDiem.upgradeStake(uint256,uint256) (Carpediem_flat.sol#702-745):
	External calls:
	- extraShares = _buyShares(_amount) (Carpediem_flat.sol#711)
		- token.safeTransferFrom(msg.sender,address(this),_amount) (Carpediem_flat.sol#871)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (Carpediem_flat.sol#462)
		- (success,returndata) = target.call{value: value}(data) (Carpediem_flat.sol#199)
	External calls sending eth:
	- extraShares = _buyShares(_amount) (Carpediem_flat.sol#711)
		- (success,returndata) = target.call{value: value}(data) (Carpediem_flat.sol#199)
	State variables written after the call(s):
	- totalShares += (extraShares + bBonusShares + lBonusShares - stakeInfo.bBonusShares) (Carpediem_flat.sol#725)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in CarpeDiem.deposit(uint256,uint32) (Carpediem_flat.sol#676-700):
	External calls:
	- shares = _buyShares(_amount) (Carpediem_flat.sol#680)
		- token.safeTransferFrom(msg.sender,address(this),_amount) (Carpediem_flat.sol#871)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (Carpediem_flat.sol#462)
		- (success,returndata) = target.call{value: value}(data) (Carpediem_flat.sol#199)
	External calls sending eth:
	- shares = _buyShares(_amount) (Carpediem_flat.sol#680)
		- (success,returndata) = target.call{value: value}(data) (Carpediem_flat.sol#199)
	Event emitted after the call(s):
	- Deposit(msg.sender,stakes[msg.sender].length - 1,_amount,_duration) (Carpediem_flat.sol#699)
	- SharesChanged(totalShares,totalShares + shares + lBonusShares + bBonusShares) (Carpediem_flat.sol#684)
Reentrancy in CarpeDiem.removeDeadStake(address,uint256) (Carpediem_flat.sol#777-814):
	External calls:
	- token.safeTransfer(_user,stakeInfo.amount) (Carpediem_flat.sol#812)
	Event emitted after the call(s):
	- StakeRemoved(_user,_stakeId,stakeInfo.amount) (Carpediem_flat.sol#813)
Reentrancy in CarpeDiem.upgradeStake(uint256,uint256) (Carpediem_flat.sol#702-745):
	External calls:
	- extraShares = _buyShares(_amount) (Carpediem_flat.sol#711)
		- token.safeTransferFrom(msg.sender,address(this),_amount) (Carpediem_flat.sol#871)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (Carpediem_flat.sol#462)
		- (success,returndata) = target.call{value: value}(data) (Carpediem_flat.sol#199)
	External calls sending eth:
	- extraShares = _buyShares(_amount) (Carpediem_flat.sol#711)
		- (success,returndata) = target.call{value: value}(data) (Carpediem_flat.sol#199)
	Event emitted after the call(s):
	- SharesChanged(totalShares,totalShares + extraShares + bBonusShares + lBonusShares - stakeInfo.bBonusShares) (Carpediem_flat.sol#723-724)
	- StakeUpgraded(msg.sender,_stakeId,_amount,stakeInfo.startTs + stakeInfo.duration - blockTimestamp) (Carpediem_flat.sol#739-744)
Reentrancy in CarpeDiem.withdraw(uint256) (Carpediem_flat.sol#747-775):
	External calls:
	- token.safeTransfer(msg.sender,stakeInfo.amount + reward - penalty) (Carpediem_flat.sol#773)
	Event emitted after the call(s):
	- Withdraw(msg.sender,_stakeId,stakeInfo.amount,reward,penalty) (Carpediem_flat.sol#774)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

CarpeDiem.upgradeStake(uint256,uint256) (Carpediem_flat.sol#702-745) uses timestamp for comparisons
	Dangerous comparisons:
	- require(bool,string)(block.timestamp < stakeInfo.duration + stakeInfo.startTs,stake matured) (Carpediem_flat.sol#707-710)
CarpeDiem.withdraw(uint256) (Carpediem_flat.sol#747-775) uses timestamp for comparisons
	Dangerous comparisons:
	- totalShares == 0 (Carpediem_flat.sol#764)
CarpeDiem.removeDeadStake(address,uint256) (Carpediem_flat.sol#777-814) uses timestamp for comparisons
	Dangerous comparisons:
	- require(bool,string)(uint32(block.timestamp) >= stakeInfo.startTs + stakeInfo.duration + 31536000,stakeAlive) (Carpediem_flat.sol#781-784)
	- totalShares == 0 (Carpediem_flat.sol#802)
CarpeDiem.getReward(address,uint256) (Carpediem_flat.sol#848-864) uses timestamp for comparisons
	Dangerous comparisons:
```

```
CarpeDiem._getBonusL(uint256,uint32) (Carpediem_flat.sol#888-899) uses timestamp for comparisons
	Dangerous comparisons:
	- _duration < poolLBonus (Carpediem_flat.sol#894)
CarpeDiem._getPenalty(address,uint256,uint256) (Carpediem_flat.sol#901-920) uses timestamp for comparisons
	Dangerous comparisons:
	- startTs + duration <= blockTimestamp (Carpediem_flat.sol#910)
	- startTs + duration + WEEK > blockTimestamp (Carpediem_flat.sol#911)
	- lateWeeks >= MAX_PENALTY_DURATION (Carpediem_flat.sol#913)
CarpeDiem._changeSharesPrice(uint256,uint256) (Carpediem_flat.sol#922-931) uses timestamp for comparisons
	Dangerous comparisons:
	- _profit > (oldPrice * _shares) / (MULTIPLIER) (Carpediem_flat.sol#924)
	- newPrice > MAX_PRICE (Carpediem_flat.sol#927)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

Address.isContract(address) (Carpediem_flat.sol#94-104) uses assembly
	- INLINE ASM (Carpediem_flat.sol#100-102)
Address.verifyCallResult(bool,bytes,string) (Carpediem_flat.sol#263-283) uses assembly
	- INLINE ASM (Carpediem_flat.sol#275-278)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Address.functionCall(address,bytes) (Carpediem_flat.sol#147-149) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (Carpediem_flat.sol#176-182) is never used and should be removed
Address.functionDelegateCall(address,bytes) (Carpediem_flat.sol#236-238) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (Carpediem_flat.sol#246-255) is never used and should be removed
Address.functionStaticCall(address,bytes) (Carpediem_flat.sol#209-211) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (Carpediem_flat.sol#219-228) is never used and should be removed
Address.sendValue(address,uint256) (Carpediem_flat.sol#122-127) is never used and should be removed
Context._msgData() (Carpediem_flat.sol#491-493) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (Carpediem_flat.sol#414-427) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (Carpediem_flat.sol#438-449) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (Carpediem_flat.sol#429-436) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (Carpediem_flat.sol#6) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (Carpediem_flat.sol#71) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (Carpediem_flat.sol#290) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (Carpediem_flat.sol#374) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (Carpediem_flat.sol#474) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (Carpediem_flat.sol#500) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (Carpediem_flat.sol#572) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.7 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (Carpediem_flat.sol#122-127):
	- (success) = recipient.call{value: amount}() (Carpediem_flat.sol#125)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (Carpediem_flat.sol#190-201):
	- (success,returndata) = target.call{value: value}(data) (Carpediem_flat.sol#199)
Low level call in Address.functionStaticCall(address,bytes,string) (Carpediem_flat.sol#219-228):
	- (success,returndata) = target.staticcall(data) (Carpediem_flat.sol#226)
Low level call in Address.functionDelegateCall(address,bytes,string) (Carpediem_flat.sol#246-255):
	- (success,returndata) = target.delegatecall(data) (Carpediem_flat.sol#253)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter CarpeDiem.getStakesLength(address)._staker (Carpediem_flat.sol#672) is not in mixedCase
Parameter CarpeDiem.deposit(uint256,uint32)._amount (Carpediem_flat.sol#676) is not in mixedCase
Parameter CarpeDiem.deposit(uint256,uint32)._duration (Carpediem_flat.sol#676) is not in mixedCase
Parameter CarpeDiem.upgradeStake(uint256,uint256)._stakeId (Carpediem_flat.sol#702) is not in mixedCase
Parameter CarpeDiem.upgradeStake(uint256,uint256)._amount (Carpediem_flat.sol#702) is not in mixedCase
Parameter CarpeDiem.withdraw(uint256)._stakeId (Carpediem_flat.sol#747) is not in mixedCase
Parameter CarpeDiem.removeDeadStake(address,uint256)._user (Carpediem_flat.sol#777) is not in mixedCase
Parameter CarpeDiem.removeDeadStake(address,uint256)._stakeId (Carpediem_flat.sol#777) is not in mixedCase
Parameter CarpeDiem.getPenalty(address,uint256)._user (Carpediem_flat.sol#839) is not in mixedCase
Parameter CarpeDiem.getPenalty(address,uint256)._stakeId (Carpediem_flat.sol#839) is not in mixedCase
```

```
Address.functionDelegateCall(address,bytes) (Carpediem_flat.sol#236-238) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (Carpediem_flat.sol#246-255) is never used and should be removed
Address.functionStaticCall(address,bytes) (Carpediem_flat.sol#209-211) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (Carpediem_flat.sol#219-228) is never used and should be removed
Address.sendValue(address,uint256) (Carpediem_flat.sol#122-127) is never used and should be removed
Context._msgData() (Carpediem_flat.sol#491-493) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (Carpediem_flat.sol#414-427) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (Carpediem_flat.sol#438-449) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (Carpediem_flat.sol#429-436) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (Carpediem_flat.sol#6) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (Carpediem_flat.sol#71) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (Carpediem_flat.sol#290) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (Carpediem_flat.sol#374) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (Carpediem_flat.sol#474) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (Carpediem_flat.sol#500) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (Carpediem_flat.sol#572) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.7 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (Carpediem_flat.sol#122-127):
        - (success) = recipient.call{value: amount}() (Carpediem_flat.sol#125)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (Carpediem_flat.sol#190-201):
        - (success,returndata) = target.call{value: value}(data) (Carpediem_flat.sol#199)
Low level call in Address.functionStaticCall(address,bytes,string) (Carpediem_flat.sol#219-228):
        - (success,returndata) = target.staticcall(data) (Carpediem_flat.sol#226)
Low level call in Address.functionDelegateCall(address,bytes,string) (Carpediem_flat.sol#246-255):
        - (success,returndata) = target.delegatecall(data) (Carpediem_flat.sol#253)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Parameter CarpeDiem.getStakesLength(address)._staker (Carpediem_flat.sol#672) is not in mixedCase
Parameter CarpeDiem.deposit(uint256,uint32)._amount (Carpediem_flat.sol#676) is not in mixedCase
Parameter CarpeDiem.deposit(uint256,uint32)._duration (Carpediem_flat.sol#676) is not in mixedCase
Parameter CarpeDiem.upgradeStake(uint256,uint256)._stakeId (Carpediem_flat.sol#702) is not in mixedCase
Parameter CarpeDiem.upgradeStake(uint256,uint256)._amount (Carpediem_flat.sol#702) is not in mixedCase
Parameter CarpeDiem.withdraw(uint256)._stakeId (Carpediem_flat.sol#747) is not in mixedCase
Parameter CarpeDiem.removeDeadStake(address,uint256)._user (Carpediem_flat.sol#777) is not in mixedCase
Parameter CarpeDiem.removeDeadStake(address,uint256)._stakeId (Carpediem_flat.sol#777) is not in mixedCase
Parameter CarpeDiem.getPenalty(address,uint256)._user (Carpediem_flat.sol#839) is not in mixedCase
Parameter CarpeDiem.getPenalty(address,uint256)._stakeId (Carpediem_flat.sol#839) is not in mixedCase
Parameter CarpeDiem.getReward(address,uint256)._user (Carpediem_flat.sol#848) is not in mixedCase
Parameter CarpeDiem.getReward(address,uint256)._stakeId (Carpediem_flat.sol#848) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable CarpeDiem.bBonusMaxPercent (Carpediem_flat.sol#595) is too similar to CarpeDiem.lBonusMaxPercent (Carpediem_flat.sol#596)
Variable CarpeDiem.deposit(uint256,uint32).bBonusShares (Carpediem_flat.sol#682) is too similar to CarpeDiem.deposit(uint256,uint32).lBonusShares (Carpediem_flat.sol#681)
Variable CarpeDiem.deposit(uint256,uint32).bBonusShares (Carpediem_flat.sol#682) is too similar to CarpeDiem.upgradeStake(uint256,uint256).lBonusShares (Carpediem_flat.sol#714-717)
Variable CarpeDiem.upgradeStake(uint256,uint256).bBonusShares (Carpediem_flat.sol#718-721) is too similar to CarpeDiem.deposit(uint256,uint32).lBonusShares (Carpediem_flat.sol#681)
Variable CarpeDiem.upgradeStake(uint256,uint256).bBonusShares (Carpediem_flat.sol#718-721) is too similar to CarpeDiem.upgradeStake(uint256,uint256).lBonusShares (Carpediem_flat.sol#714-717)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

CarpeDiem.slitherConstructorConstantVariables() (Carpediem_flat.sol#580-932) uses literals with too many digits:
        - DEAD_WALLET = 0x000000000000000000000000000000000000dEaD (Carpediem_flat.sol#583)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (Carpediem_flat.sol#549-551)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (Carpediem_flat.sol#557-560)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
Carpediem_flat.sol analyzed (7 contracts with 75 detectors), 61 result(s) found
ethsec@be29ddd8d48f:/code/contracts$
```

Results

Major issues are not found and one which is raised are already been categorized above according to their level of severity.

Closing Summary

In this report, we have considered the security of the Carpe Diem Savings. We performed our audit according to the procedure described above.

Some issues of Medium, Low and informational severity were found, Some suggestions and best practices are also provided in order to improve the code quality and security posture. At the end, the Carpe Diem Team fixed certain issues and acknowledged others.

Disclaimer

QuillAudits smart contract audit is not a security warranty, investment advice, or an endorsement of the Carpe Diem Savings Platform. This audit does not provide a security or correctness guarantee of the audited smart contracts.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the Carpe Diem Savings put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

About QuillAudits

QuillAudits is a secure smart contracts audit platform designed by QuillHash Technologies. We are a team of dedicated blockchain security experts and smart contract auditors determined to ensure that Smart Contract-based Web3 projects can avail the latest and best security solutions to operate in a trustworthy and risk-free ecosystem.

500+
Audits Completed

$15B
Secured

500K
Lines of Code Audited

Follow Our Journey

Audit Report
May, 2022

For

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com