// HALBORN

REEF Protocol

Smart Contract Security Audit

Prepared by: **Halborn**
Date of Engagement: December 3 - 10, 2020
Visit: Halborn.com

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	11/27/2020	Nishit Majithia
0.2	Document Edits	11/30/2020	Nishit Majithia
1.0	Final Version	12/07/2020	Nishit Majithia
1.1	Remediations	12/10/2020	Nishit Majithia

CONTACTS

CONTACT	COMPANY	EMAIL
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Nishit Majithia	Halborn	Nishit.Majithia@halborn.com

1.1 INTRODUCTION

The Reef Finance team engaged Halborn to conduct a security assessment on their smart contracts that implement the protocol on the Ethereum blockchain. Reef is a decentralized, non-custodial protocol; assets are controlled by the users themselves and not stored on the platform. Reef empowers the users to keep storage of their own private keys and cryptocurrency assets, while working in the background akin to a DeFi Operating System, optimizing yield for the users without compromising security. The security assessment was scoped to the smart contract ReefBalancer, ReefBasket, ReefLiquidityBond, ReefMooniswapV1, ReefToken, ReefUniswap and ReefVaultsBasket. An audit of the security risk and implications regarding the changes introduced by the development team at Reef Protocol prior to its production release shortly following the assessments deadline.

Overall, the smart contract code is extremely well documented, follows a high-quality software development standard, contains many utilities and automation scripts to support continuous deployment / testing / integration, and does NOT contain any obvious exploitation vectors that Halborn was able to leverage within the timeframe of testing allotted.

Though the outcome of this security audit is satisfactory; due to time and resource constraints, only testing and verification of essential properties were performed to achieve objectives and deliverables set in the scope. It is important to remark the use of the best practices for secure smart contract development. Halborn recommends performing further testing to validate extended safety and correctness in context to the whole set of contracts. External threats, such as economic attacks, oracle attacks, and inter-contract functions and calls should be

TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of the smart contract audit. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of smart contracts and can quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of the audit:Research into architecture, purpose, and use of Governance Token.

- Research into architecture and purpose
- Smart Contract manual code read and walkthrough ·
- Graphing out functionality and contract logic/connectivity/functions (solgraph)
- Manual Assessment of use and safety for the critical solidity variables and functions in scope to identify any arithmetic related vulnerability classes.
- Scanning of solidity files for vulnerabilities, security hotspots, or bugs. (MythX)
- Static Analysis of security for scoped contract, and imported functions. (Slither)
- Testnet deployment (Truffle, Ganache, Infura)
- Smart Contract Fuzzing and dynamic state exploitation (Echidna) Symbolic Execution / EVM bytecode security assessment (limited time)

1.2 SCOPE

IN-SCOPE:

- ReefBalancer.sol
- ReefBasket.sol
- ReefLiquidityBond.sol
- ReefMooniswapV1.sol
- ReefToken.sol
- ReefUniswap.sol
- ReefVaultsBasket.sol

Commit ID: 45579c600e9bd49b4d0a0c495f158535c98ee691

OUT-OF-SCOPE:

- ReefFarming.sol
- ReefStaking.sol
- External contracts, External Oracles, other smart contracts in the repository or imported by Reef protocol contracts, economic attacks

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW
0	0	0	2

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
MISSING REENTRANCY PROTECTION	Low	12/08/2020
USE OF BLOCK.TIMESTAMP	Informational	-
IGNORE RETURN VALUES	Informational	12/08/2020
TAUTOLOGY EXPRESSIONS	Informational	12/08/2020
STATIC ANALYSIS	Low	-
ERC CONFORMAL CHECKER	Informational	-

Github Commit Remediation Hash: f5a3ccbd9c1a383b2ce704b73489d011b3a7b015

FINDINGS & TECH DETAILS

3.1 MISSING REENTRANCY PROTECTION

- LOW

Description

To protect against cross-function reentrancy attacks, it may be necessary to use a mutex. By using this lock, an attacker can no longer exploit the “withdraw” function with a recursive call. OpenZeppelin has it’s own mutex implementation called ReentrancyGuard which provides a modifier to any function called “nonReentrant” that guards the function with a mutex against the Reentrancy attacks.

In ReefLiquidityBond.sol contract, functions like withdraw() and getReward() are missing nonReentrant guard. Though, these methods are implemented following checks-effects-interactions pattern only. But in longer term it is better to use “nonReentrant” guard to avoid unfortunate event in future due to code changes.

Code Location

ReefLiquidityBond.sol (https://github.com/reef-defi/reef-protocol/blob/45579c600e9bd49b4d0a0c495f158535c98ee691/contracts/ReefLiquidityBond.sol#L145)

ReefLiquidityBond.sol (https://github.com/reef-defi/reef-protocol/blob/45579c600e9bd49b4d0a0c495f158535c98ee691/contracts/ReefLiquidityBond.sol#L157)

3.2 USE OF BLOCK.TIMESTAMP - INFORMATIONAL

Description:

Block timestamps have historically been used for a variety of applications, such as entropy for random numbers, locking funds for periods of time and various state-changing conditional statements that are time-dependent. Miner's have the ability to

adjust timestamps slightly which can prove to be quite dangerous if block timestamps are used incorrectly in smart contracts. block.timestamp or its alias now can be manipulated by miners if they have some incentive to do so.

Recommendation:
Avoid relying on **block.timestamp**

Code Location:
Many instances in ReefLiquidityBond.sol(https://github.com/reef-defi/reef-protocol/blob/45579c600e9bd49b4d0a0c495f158535c98ee691/contracts/ReefLiquidityBond.sol) contract on line number - 85, 86, 87, 102, 131, 132, 139, 140, 147, 158, 168, 169, 172

3.3 IGNORE RETURN VALUES - INFORMATIONAL

Description:
The return value of an external call is not stored in a local or state variable. In contracts ReefBasket.sol, ReefBalancer.sol, ReefMooniswapV1.sol and ReefVaultsBasket.sol there are few instances where external methods are being called and return value(bool or uint) are being ignored.

Recommendation:
Add return value check to avoid unexpected crash of the contract. Return value check will help in handling the exceptions better way.

Code Locatioin:
ReefBasket.sol: Ignoring *boolean* return value (https://github.com/reef-defi/reef-protocol/blob/45579c600e9bd49b4d0a0c495f158535c98ee691/contracts/ReefBasket.sol#L636)

```
633                 .mul(_percent)
634                 .div(100);
635
636             IERC20(availableBaskets[_basketIndex].balancerPools[b].poolAddress)
637                 .approve(address(ReefBalancer), disinvestAmount);
638
639             // TODO: figure out slippage
640             uint256 balancerTokens = ReefBalancer.disinvestFromBalancerPool(
```

ReefBasket.sol: Ignoring *uint* return value

(https://github.com/reef-defi/reef-protocol/blob/45579c600e9bd49b4d0a0c495f158535c98ee691/contracts/ReefBasket.sol#L719)

```
717             // Check if we restake into the ETH/protocolToken pool
718             if (shouldRestake) {
719                 ReefUniswap._investIntoUniswapPool(
720                     address(0),
721                     wethTokenAddress,
722                     protocolTokenAddress,
723                     msg.sender,
724                     profit.mul(yieldRatio).div(100)
725                 );
726             } else {
```

ReefBalancer.sol: Ignoring boolean return value

(https://github.com/reef-defi/reef-protocol/blob/45579c600e9bd49b4d0a0c495f158535c98ee691/contracts/ReefBalancer.sol#L250)

```
248         require(returnedTokens >= _minTokensRec, "High slippage");
249
250         IERC20(_ToTokenContractAddress).transfer(
251             _toWhomToIssue,
252             returnedTokens
253         );
```

(https://github.com/reef-defi/reef-protocol/blob/45579c600e9bd49b4d0a0c495f158535c98ee691/contracts/ReefBalancer.sol#L408)

```
406         uint256 tokens2Trade
407     ) internal returns (uint256 tokenBought) {
408         IERC20(_FromTokenContractAddress).approve(
409             address(uniswapRouter),
410             tokens2Trade
411         );
412
```

ReefMooniswapV1.sol: Ignoring *boolean* return values

(https://github.com/reef-defi/reef-protocol/blob/45579c600e9bd49b4d0a0c495f158535c98ee691/contracts/ReefMooniswapV1.sol#L45)

(https://github.com/reef-defi/reef-protocol/blob/45579c600e9bd49b4d0a0c495f158535c98ee691/contracts/ReefMooniswapV1.sol#L65)

```
44
45              token.approve(
46                  poolAddress,
47                  tokenBought
48              );
49
50              fairSupply = pool.deposit{value: halfAmount}(
51                  amounts,
52                  minAmounts
53              );
54          } else {
55              (uint256 token0Bought, uint256 token1Bought) = ReefUniswap.exchangeTokensV2(
56                  _FromTokenContractAddress,
57                  address(ercTokens[0]),
58                  address(ercTokens[1]),
59                  _amount
60              );
61
62              amounts[0] = token0Bought;
63              amounts[1] = token1Bought;
64
65              ercTokens[0].approve(
66                  poolAddress,
67                  token0Bought
68              );
```

ReefMooniswapV1.sol: Ignoring uint return values (https://github.com/reef-defi/reef-protocol/blob/45579c600e9bd49b4d0a0c495f158535c98ee691/contracts/ReefMooniswapV1.sol#L86)

(https://github.com/reef-defi/reef-protocol/blob/45579c600e9bd49b4d0a0c495f158535c98ee691/contracts/ReefMooniswapV1.sol#L91)

```
85          if (token0Balance > 0) {
86              ReefUniswap.swapFromV2(address(ercTokens[0]),
87                                     _FromTokenContractAddress, token0Balance);
88          }
89
90          if (token1Balance > 0) {
91              ReefUniswap.swapFromV2(address(ercTokens[1]),
92                                     _FromTokenContractAddress, token1Balance);
93          }
```

(https://github.com/reef-defi/reef-protocol/blob/45579c600e9bd49b4d0a0c495f158535c98ee691/contracts/ReefMooniswapV1.sol#L121)

```
118
119         for (uint i = 0; i < tokenReturns.length; i++) {
120             if (!ercTokens[i].isETH()) {
121                 ReefUniswap.swapFromV2(address(ercTokens[i]),
122                                        _ToTokenContractAddress,
123                                        tokenReturns[i]);
124             }
125         }
```

ReefVaultsBasket.sol: Ignoring uint return value (https://github.com/reef-defi/reef-protocol/blob/45579c600e9bd49b4d0a0c495f158535c98ee691/contracts/ReefVaultsBasket.sol#L406)

```
404             // Check if we restake into the ETH/protocolToken pool
405             if (shouldRestake) {
406                 ReefUniswap._investIntoUniswapPool(
407                     address(0),
408                     wethTokenAddress,
409                     protocolTokenAddress,
410                     msg.sender,
411                     profit.mul(yieldRatio).div(100)
412                 );
413             } else {
```

3.4 TAUTOLOGY EXPRESSIONS - INFORMATIONAL

Description:

In contracts ReefBasket.sol and ReefVaultsBasket.sol, tautology expression has been detected. These expressions are of no use since it will always returns true while using in any condition

Recommendation:

Correct these expressions

Code Location:

ReefBasket.sol: (https://github.com/reef-defi/reef-protocol/blob/45579c600e9bd49b4d0a0c495f158535c98ee691/contracts/ReefBasket.sol#L753)

Since _newpercentage variable is declared as type of uint16, it will always >=0

```
748     function setProtocolTokenDisinvestPercentage(uint16 _newPercentage)
749         public
750         onlyOwner
751     {
752         require(
753             _newPercentage >= 0 && _newPercentage < 100,
754             "_newPercentage must be between 0 and 100."
755         );
756         protocolTokenDisinvestPercentage = _newPercentage;
757     }
758
```

ReefVaultsBasket.sol: (https://github.com/reef-defi/reef-protocol/blob/45579c600e9bd49b4d0a0c495f158535c98ee691/contracts/ReefVaultsBasket.sol#L699)

Since _newpercentage variable is declared as type of uint16, it

will always >=0

```
694     function setProtocolTokenDisinvestPercentage(uint16 _newPercentage)
695         public
696         onlyOwner
697     {
698         require(
699             _newPercentage >= 0 && _newPercentage < 100,
700             "_newPercentage must be between 0 and 100."
701         );
702         protocolTokenDisinvestPercentage = _newPercentage;
703     }
```

3.5 STATIC ANALYSIS - LOW

Description:
Slither and MythX has been run on all the scoped contracts (ReefBalancer.sol, ReefBasket.sol, ReefMooniswapV1.sol, ReefLiquidityBond.sol, ReefToken.sol, ReefUniswap.sol and ReefVaultsBasket.sol)

```
INFO:Detectors:
ReefBalancer._performZapOut(address,address,address,uint256,address,uint256) (contracts/ReefBalancer.sol#180-225) sends eth to arbitrary user
        Dangerous calls:
        - _toWhomToIssue.transfer(ethBought) (contracts/ReefBalancer.sol#213)
ReefBasket._multiInvest(uint256[],uint256[],uint256) (contracts/ReefBasket.sol#365-393) sends eth to arbitrary user
        Dangerous calls:
        - (success) = msg.sender.call{value: address(this).balance}() (contracts/ReefBasket.sol#392)
ReefBasket.withdraw() (contracts/ReefBasket.sol#798-802) sends eth to arbitrary user
        Dangerous calls:
        - _to.transfer(contractBalance) (contracts/ReefBasket.sol#801)
ReefMooniswapV1.investIntoMooniswapPool(address,address,uint256) (contracts/ReefMooniswapV1.sol#23-95) sends eth to arbitrary user
        Dangerous calls:
        - fairSupply = pool.deposit{value: halfAmount}(amounts,minAmounts) (contracts/ReefMooniswapV1.sol#50-53)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
Reentrancy in ReefLiquidityBond.exit() (contracts/ReefLiquidityBond.sol#152-155):
        External calls:
        - withdraw(balanceOf(msg.sender)) (contracts/ReefLiquidityBond.sol#153)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (contracts/openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
                - lpToken.safeTransfer(msg.sender,amount) (contracts/ReefLiquidityBond.sol#48)
                - (success,returndata) = target.call{value: value}(data) (contracts/openzeppelin/contracts/utils/Address.sol#119)
        - getReward() (contracts/ReefLiquidityBond.sol#154)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (contracts/openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
                - (success,returndata) = target.call{value: value}(data) (contracts/openzeppelin/contracts/utils/Address.sol#119)
                - reef.safeTransfer(msg.sender,reward) (contracts/ReefLiquidityBond.sol#162)
        External calls sending eth:
        - withdraw(balanceOf(msg.sender)) (contracts/ReefLiquidityBond.sol#153)
                - (success,returndata) = target.call{value: value}(data) (contracts/openzeppelin/contracts/utils/Address.sol#119)
        - getReward() (contracts/ReefLiquidityBond.sol#154)
                - (success,returndata) = target.call{value: value}(data) (contracts/openzeppelin/contracts/utils/Address.sol#119)
        State variables written after the call(s):
        - getReward() (contracts/ReefLiquidityBond.sol#154)
                - lastUpdateTime = lastTimeRewardApplicable() (contracts/ReefLiquidityBond.sol#92)
        - getReward() (contracts/ReefLiquidityBond.sol#154)
                - rewardPerTokenStored = rewardPerToken() (contracts/ReefLiquidityBond.sol#91)
        - getReward() (contracts/ReefLiquidityBond.sol#154)
                - rewards[msg.sender] = 0 (contracts/ReefLiquidityBond.sol#161)
                - rewards[account] = earned(account) (contracts/ReefLiquidityBond.sol#95)
        - getReward() (contracts/ReefLiquidityBond.sol#154)
                - userRewardPerTokenPaid[account] = rewardPerTokenStored (contracts/ReefLiquidityBond.sol#96)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
INFO:Detectors:
UniERC20._toHex(bytes) (contracts/libraries/UniERC20.sol#90-103) performs a multiplication on the result of a division:
        -str[j ++] = bytes1(uint8(a + 48 + (a / 10) * 39)) (contracts/libraries/UniERC20.sol#98)
UniERC20._toHex(bytes) (contracts/libraries/UniERC20.sol#90-103) performs a multiplication on the result of a division:
        -str[j ++] = bytes1(uint8(b + 48 + (b / 10) * 39)) (contracts/libraries/UniERC20.sol#99)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
Reentrancy in ReefBasket.disinvest(uint256,uint256,uint256,bool) (contracts/ReefBasket.sol#530-746):
        External calls:
        - (amountA,amountB) = ReefUniswap._disinvestFromUniswapPool(address(0),pool.uniswapToken0,pool.uniswapToken1,disinvestAmount) (contracts/ReefBasket.sol#566-572)
        State variables written after the call(s):
        - userBalance[msg.sender].basketBalances[_basketIndex].uniswapPools[p] = userBalance[msg.sender].basketBalances[_basketIndex].uniswapPools[p].sub(disinvestAmount) (contracts/ReefBasket.sol#574-578)
Reentrancy in ReefBasket.disinvest(uint256,uint256,uint256,bool) (contracts/ReefBasket.sol#530-746):
        External calls:
        - tokenBought = ReefUniswap.swapFromV2(token.tokenAddress,address(0),disinvestAmount_scope_1) (contracts/ReefBasket.sol#597-601)
        - TransferHelper.safeTransfer(address(0),msg.sender,tokenBought) (contracts/ReefBasket.sol#603-607)
        State variables written after the call(s):
        - userBalance[msg.sender].basketBalances[_basketIndex].tokens[t] = userBalance[msg.sender].basketBalances[_basketIndex].tokens[t].sub(disinvestAmount_scope_1) (contracts/ReefBasket.sol#610-614)
Reentrancy in ReefBasket.disinvest(uint256,uint256,uint256,bool) (contracts/ReefBasket.sol#530-746):
        External calls:
        - IERC20(availableBaskets[_basketIndex].balancerPools[b].poolAddress).approve(address(ReefBalancer),disinvestAmount_scope_2) (contracts/ReefBasket.sol#636-637)
        - balancerTokens = ReefBalancer.disinvestFromBalancerPool(address(address(this)),address(0),availableBaskets[_basketIndex].balancerPools[b].poolAddress,disinvestAmount_scope_2,1) (contracts/ReefBasket.sol#640
-646)
        State variables written after the call(s):
        - userBalance[msg.sender].basketBalances[_basketIndex].balancerPools[b] = userBalance[msg.sender].basketBalances[_basketIndex].balancerPools[b].sub(disinvestAmount_scope_2) (contracts/ReefBasket.sol#648-652)
Reentrancy in ReefBasket.disinvest(uint256,uint256,uint256,bool) (contracts/ReefBasket.sol#530-746):
        External calls:
        - ReefMooniswapV1.disinvestFromMooniswapPool(address(0),availableBaskets[_basketIndex].mooniswapPools[b_scope_3].poolAddress,disinvestAmount_scope_4) (contracts/ReefBasket.sol#675-679)
        State variables written after the call(s):
        - userBalance[msg.sender].basketBalances[_basketIndex].mooniswapPools[b_scope_3] = userBalance[msg.sender].basketBalances[_basketIndex].mooniswapPools[b_scope_3].sub(disinvestAmount_scope_4) (contracts/ReefBa
sket.sol#681-685)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
```

FINDINGS & TECH DETAILS

```
INFO:Detectors:
ReefBasket.setProtocolTokenDisinvestPercentage(uint16) (contracts/ReefBasket.sol#748-757) contains a tautology or contradiction:
	- require(bool,string)(_newPercentage >= 0 && _newPercentage < 100,_newPercentage must be between 0 and 100.) (contracts/ReefBasket.sol#752-753)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#tautology-or-contradiction
INFO:Detectors:
ReefBalancer._getBestDeal(address,uint256).maxEth (contracts/ReefBalancer.sol#568) is a local variable never initialized
ReefUniswap._investIntoUniswapPool(address,address,address,address,uint256).token0Bought (contracts/ReefUniswap.sol#35) is a local variable never initialized
ReefBalancer._getBestDeal(address,uint256,address).maxBPT (contracts/ReefBalancer.sol#501) is a local variable never initialized
ReefUniswap._investIntoUniswapPool(address,address,address,address,uint256).residue (contracts/ReefUniswap.sol#73) is a local variable never initialized
ReefUniswap._investIntoUniswapPool(address,address,address,address,uint256).token1Bought (contracts/ReefUniswap.sol#36) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
ReefBalancer._directZapout(address,address,address,uint256,uint256) (contracts/ReefBalancer.sol#235-254) ignores return value by IERC20(_ToTokenContractAddress).transfer(_toWhomToIssue,returnedTokens) (contracts/ReefBalancer.sol#250-253)
ReefBalancer._token2Token(address,address,address,uint256) (contracts/ReefBalancer.sol#402-453) ignores return value by IERC20(_FromTokenContractAddress).approve(address(uniswapRouter),tokensZTrade) (contracts/ReefBalancer.sol#408-411)
ReefBasket.disinvest(uint256,uint256,uint256,bool) (contracts/ReefBasket.sol#530-746) ignores return value by IERC20(availableBaskets[_basketIndex].balancerPools[b].poolAddress).approve(address(ReefBalancer),disinvestAmount_scope_2) (contracts/ReefBasket.sol#636-637)
ReefBasket.disinvest(uint256,uint256,uint256,bool) (contracts/ReefBasket.sol#530-746) ignores return value by ReefUniswap._investIntoUniswapPool(address(0),wethTokenAddress,protocolTokenAddress,msg.sender,profit.mul(yieldRatio).div(100)) (contracts/ReefBasket.sol#719-725)
ReefMooniswapV1.investIntoMooniswapPool(address,address,uint256) (contracts/ReefMooniswapV1.sol#23-95) ignores return value by token.approve(poolAddress,tokenBought) (contracts/ReefMooniswapV1.sol#45-48)
ReefMooniswapV1.investIntoMooniswapPool(address,address,uint256) (contracts/ReefMooniswapV1.sol#23-95) ignores return value by ercTokens[0].approve(poolAddress,token0Bought) (contracts/ReefMooniswapV1.sol#65-68)
ReefMooniswapV1.investIntoMooniswapPool(address,address,uint256) (contracts/ReefMooniswapV1.sol#23-95) ignores return value by ReefUniswap.swapFromV2(address(ercTokens[0]),_FromTokenContractAddress,token0Balance) (contracts/ReefMooniswapV1.sol#86-87)
ReefMooniswapV1.investIntoMooniswapPool(address,address,uint256) (contracts/ReefMooniswapV1.sol#23-95) ignores return value by ReefUniswap.swapFromV2(address(ercTokens[1]),_FromTokenContractAddress,token1Balance) (contracts/ReefMooniswapV1.sol#91-92)
ReefMooniswapV1.disinvestFromMooniswapPool(address,address,uint256) (contracts/ReefMooniswapV1.sol#97-126) ignores return value by ReefUniswap.swapFromV2(address(ercTokens[i_scope_0]),_ToTokenContractAddress,tokenReturns[i_scope_0]) (contracts/ReefMooniswapV1.sol#121-123)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
ReefToken._writeCheckpoint(address,uint32,uint256,uint256) (contracts/ReefToken.sol#216-234) uses a dangerous strict equality:
	- nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber (contracts/ReefToken.sol#226)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
ReefBasket.balanceOfUniswapPools(address,uint256)._owner (contracts/ReefBasket.sol#96) shadows:
	- Ownable._owner (contracts/openzeppelin/contracts/access/Ownable.sol#19) (state variable)
ReefBasket.balanceOfTokens(address,uint256)._owner (contracts/ReefBasket.sol#115) shadows:
	- Ownable._owner (contracts/openzeppelin/contracts/access/Ownable.sol#19) (state variable)
ReefBasket.balanceOfBalancerPools(address,uint256)._owner (contracts/ReefBasket.sol#131) shadows:
	- Ownable._owner (contracts/openzeppelin/contracts/access/Ownable.sol#19) (state variable)
ReefBasket.balanceOfMooniswapPools(address,uint256)._owner (contracts/ReefBasket.sol#150) shadows:
	- Ownable._owner (contracts/openzeppelin/contracts/access/Ownable.sol#19) (state variable)
ReefBasket.investedAmountInBasket(address,uint256)._owner (contracts/ReefBasket.sol#169) shadows:
	- Ownable._owner (contracts/openzeppelin/contracts/access/Ownable.sol#19) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
ReefBalancer._getBestDeal(address,uint256,address) (contracts/ReefBalancer.sol#462-551) has external calls inside a loop: UniSwapV2FactoryAddress.getPair(tokens[index],wethTokenAddress) == address(0) (contracts/ReefBalancer.sol#509-512)
ReefBalancer._getBestDeal(address,uint256,address) (contracts/ReefBalancer.sol#462-551) has external calls inside a loop: expectedTokens = uniswapRouter.getAmountsOut(amount,path)[1] (contracts/ReefBalancer.sol#519-522)
ReefBalancer._getBestDeal(address,uint256) (contracts/ReefBalancer.sol#560-610) has external calls inside a loop: UniSwapV2FactoryAddress.getPair(tokens[index],wethTokenAddress) == address(0) (contracts/ReefBalancer.sol#581-584)
ReefBalancer._getBestDeal(address,uint256) (contracts/ReefBalancer.sol#560-610) has external calls inside a loop: ethReturned = uniswapRouter.getAmountsOut(tokensForBPT,path)[1] (contracts/ReefBalancer.sol#592-595)
ReefBasket.disinvest(uint256,uint256,uint256,bool) (contracts/ReefBasket.sol#530-746) has external calls inside a loop: IERC20(availableBaskets[_basketIndex].balancerPools[b].poolAddress).approve(address(ReefBalancer),disinvestAmount_scope_2) (contracts/ReefBasket.sol#636-637)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#calls-inside-a-loop
INFO:Detectors:
Reentrancy in ReefBasket._invest(uint256,uint256,uint256) (contracts/ReefBasket.sol#395-528):
	External calls:
	- LPBought = ReefUniswap._investIntoUniswapPool(address(0),pool.uniswapToken0,pool.uniswapToken1,address(this),investAmount) (contracts/ReefBasket.sol#417-423)
	State variables written after the call(s):
	- userBalance[msg.sender].basketBalances[_basketIndex].uniswapPools[i] = userBalance[msg.sender].basketBalances[_basketIndex].uniswapPools[i].add(LPBought) (contracts/ReefBasket.sol#427-431)
Reentrancy in ReefBasket._invest(uint256,uint256,uint256) (contracts/ReefBasket.sol#395-528):
	External calls:
	- tokenBought = ReefUniswap.swapFromV2(address(0),token.tokenAddress,investAmount_scope_1) (contracts/ReefBasket.sol#443-447)
	State variables written after the call(s):
	- userBalance[msg.sender].basketBalances[_basketIndex].tokens[i_scope_0] = userBalance[msg.sender].basketBalances[_basketIndex].tokens[i_scope_0].add(tokenBought) (contracts/ReefBasket.sol#449-453)
Reentrancy in ReefBasket._invest(uint256,uint256,uint256) (contracts/ReefBasket.sol#395-528):
	External calls:
	- balancerTokens = ReefBalancer.investIntoBalancerPool(address(this),address(0),balancerPool.poolAddress,investAmount_scope_3,_minPoolTokens) (contracts/ReefBasket.sol#466-472)
	State variables written after the call(s):
	- userBalance[msg.sender].basketBalances[_basketIndex].balancerPools[i_scope_2] = userBalance[msg.sender].basketBalances[_basketIndex].balancerPools[i_scope_2].add(balancerTokens) (contracts/ReefBasket.sol#474-478)
Reentrancy in ReefBasket._invest(uint256,uint256,uint256) (contracts/ReefBasket.sol#395-528):
	External calls:
	- fairSupply = ReefMooniswapV1.investIntoMooniswapPool(address(0),mooniswapPool.poolAddress,investAmount_scope_5) (contracts/ReefBasket.sol#491-495)
	State variables written after the call(s):
	- userBalance[msg.sender].basketBalances[_basketIndex].mooniswapPools[i_scope_4] = userBalance[msg.sender].basketBalances[_basketIndex].mooniswapPools[i_scope_4].add(fairSupply) (contracts/ReefBasket.sol#497-501)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in ReefliquidityBond.exit() (contracts/ReefliquidityBond.sol#152-155):
	External calls:
	- withdraw(balanceOf(msg.sender)) (contracts/ReefliquidityBond.sol#153)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (contracts/openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
		- lpToken.safeTransfer(msg.sender,amount) (contracts/ReefliquidityBond.sol#48)
		- (success,returndata) = target.call{value: value}(data) (contracts/openzeppelin/contracts/utils/Address.sol#119)
	- getReward() (contracts/ReefliquidityBond.sol#154)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (contracts/openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
		- (success,returndata) = target.call{value: value}(data) (contracts/openzeppelin/contracts/utils/Address.sol#119)
		- reef.safeTransfer(msg.sender,reward) (contracts/ReefliquidityBond.sol#162)
	External calls sending eth:
	- withdraw(balanceOf(msg.sender)) (contracts/ReefliquidityBond.sol#153)
		- (success,returndata) = target.call{value: value}(data) (contracts/openzeppelin/contracts/utils/Address.sol#119)
	- getReward() (contracts/ReefliquidityBond.sol#154)
		- (success,returndata) = target.call{value: value}(data) (contracts/openzeppelin/contracts/utils/Address.sol#119)
	Event emitted after the call(s):
	- RewardPaid(msg.sender,reward) (contracts/ReefliquidityBond.sol#163)
		- getReward() (contracts/ReefliquidityBond.sol#154)
Reentrancy in ReefliquidityBond.getReward() (contracts/ReefliquidityBond.sol#157-165):
	External calls:
	- reef.safeTransfer(msg.sender,reward) (contracts/ReefliquidityBond.sol#162)
	Event emitted after the call(s):
	- RewardPaid(msg.sender,reward) (contracts/ReefliquidityBond.sol#163)
Reentrancy in ReefliquidityBond.increaseReward(uint256) (contracts/ReefliquidityBond.sol#167-176):
	External calls:
	- reef.safeTransferFrom(msg.sender,address(this),amount) (contracts/ReefliquidityBond.sol#174)
	Event emitted after the call(s):
	- RewardAdded(amount) (contracts/ReefliquidityBond.sol#175)
Reentrancy in ReefliquidityBond.stake(uint256) (contracts/ReefliquidityBond.sol#137-143):
	External calls:
	- super.stake(amount) (contracts/ReefliquidityBond.sol#141)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (contracts/openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
		- lpToken.safeTransferFrom(msg.sender,address(this),amount) (contracts/ReefliquidityBond.sol#36)
		- (success,returndata) = target.call{value: value}(data) (contracts/openzeppelin/contracts/utils/Address.sol#119)
	External calls sending eth:
	- super.stake(amount) (contracts/ReefliquidityBond.sol#141)
		- (success,returndata) = target.call{value: value}(data) (contracts/openzeppelin/contracts/utils/Address.sol#119)
	Event emitted after the call(s):
	- Staked(msg.sender,amount) (contracts/ReefliquidityBond.sol#142)
Reentrancy in ReefliquidityBond.stakeFor(address,uint256) (contracts/ReefliquidityBond.sol#129-135):
	External calls:
	- super.stakeFor(beneficiary,amount) (contracts/ReefliquidityBond.sol#133)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (contracts/openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
		- lpToken.safeTransferFrom(msg.sender,address(this),amount) (contracts/ReefliquidityBond.sol#42)
		- (success,returndata) = target.call{value: value}(data) (contracts/openzeppelin/contracts/utils/Address.sol#119)
	External calls sending eth:
	- super.stakeFor(beneficiary,amount) (contracts/ReefliquidityBond.sol#133)
		- (success,returndata) = target.call{value: value}(data) (contracts/openzeppelin/contracts/utils/Address.sol#119)
	Event emitted after the call(s):
	- Staked(beneficiary,amount) (contracts/ReefliquidityBond.sol#134)
Reentrancy in ReefliquidityBond.withdraw(uint256) (contracts/ReefliquidityBond.sol#145-150):
	External calls:
	- super.withdraw(amount) (contracts/ReefliquidityBond.sol#148)
		- returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (contracts/openzeppelin/contracts/token/ERC20/SafeERC20.sol#69)
INFO:Detectors:
owner() should be declared external:
	- Ownable.owner() (contracts/openzeppelin/contracts/access/Ownable.sol#35-37)
renounceOwnership() should be declared external:
	- Ownable.renounceOwnership() (contracts/openzeppelin/contracts/access/Ownable.sol#54-57)
transferOwnership(address) should be declared external:
	- Ownable.transferOwnership(address) (contracts/openzeppelin/contracts/access/Ownable.sol#63-67)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
ReefToken.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (contracts/ReefToken.sol#82-123) uses timestamp for comparisons
	Dangerous comparisons:
	- require(bool,string)(now <= expiry,REEF::delegateBySig: signature expired) (contracts/ReefToken.sol#121)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
ReefToken.getChainId() (contracts/ReefToken.sol#241-245) uses assembly
	- INLINE ASM (contracts/ReefToken.sol#243)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

FINDINGS & TECH DETAILS

```
INFO:Detectors:
ReefUniswap._investIntoUniswapPool(address,address,address,address,uint256) (contracts/ReefUniswap.sol#28-115) uses timestamp for comparisons
        Dangerous comparisons:
        - SafeMath.sub(token0Bought,amountA) > 0 (contracts/ReefUniswap.sol#74)
        - SafeMath.sub(token1Bought,amountB) > 0 (contracts/ReefUniswap.sol#90)
        - residue > 0 (contracts/ReefUniswap.sol#106)
ReefUniswap.exchangeTokensV2(address,address,address,uint256) (contracts/ReefUniswap.sol#189-236) uses timestamp for comparisons
        Dangerous comparisons:
        - amountToSwap <= 0 (contracts/ReefUniswap.sol#210)
        - amountToSwap_scope_0 <= 0 (contracts/ReefUniswap.sol#228)
ReefUniswap.swapFromV2(address,address,uint256) (contracts/ReefUniswap.sol#305-368) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(amount > 0,Invalid amount) (contracts/ReefUniswap.sol#317)
        - require(bool,string)(returnedAmount > 0,Error in swap) (contracts/ReefUniswap.sol#365)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (contracts/openzeppelin/contracts/utils/Address.sol#26-35) uses assembly
        - INLINE ASM (contracts/openzeppelin/contracts/utils/Address.sol#33)
Address._verifyCallResult(bool,bytes,string) (contracts/openzeppelin/contracts/utils/Address.sol#171-188) uses assembly
        - INLINE ASM (contracts/openzeppelin/contracts/utils/Address.sol#180-183)
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<0.8.0', '>=0.6.2<0.8.0', '^0.6.12']
        - ^0.6.12 (contracts/ReefBalancer.sol#30)
        - ^0.6.12 (contracts/ReefBasket.sol#13)
        - ^0.6.12 (contracts/ReefMooniswapV1.sol#16)
        - ^0.6.12 (contracts/ReefUniswap.sol#10)
        - ^0.6.12 (contracts/libraries/UniERC20.sol#5)
        - >=0.6.0<0.8.0 (contracts/openzeppelin/contracts/GSN/Context.sol#3)
        - >=0.6.0<0.8.0 (contracts/openzeppelin/contracts/access/Ownable.sol#3)
        - >=0.6.0<0.8.0 (contracts/openzeppelin/contracts/math/SafeMath.sol#3)
        - >=0.6.0<0.8.0 (contracts/openzeppelin/contracts/token/ERC20/IERC20.sol#3)
        - >=0.6.0<0.8.0 (contracts/openzeppelin/contracts/token/ERC20/SafeERC20.sol#3)
        - >=0.6.2<0.8.0 (contracts/openzeppelin/contracts/utils/Address.sol#3)
        - >=0.6.0<0.8.0 (contracts/openzeppelin/contracts/utils/ReentrancyGuard.sol#3)
        - ^0.6.12 (contracts/utils/Babylonian.sol#4)
        - ^0.6.12 (interfaces/IBFactory.sol#3)
        - ^0.6.12 (interfaces/IBPool.sol#1)
        - ^0.6.12 (interfaces/IMooniswap.sol#3)
        - ^0.6.12 (interfaces/IUniswapV2Factory.sol#1)
        - ^0.6.12 (interfaces/IUniswapV2Pair.sol#1)
        - ^0.6.12 (interfaces/IUniswapV2Router.sol#3)
        - ^0.6.12 (interfaces/TransferHelper.sol#1)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
```

```
INFO:Detectors:
Pragma version^0.6.12 (contracts/ReefBalancer.sol#30) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version^0.6.12 (contracts/ReefBasket.sol#13) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version^0.6.12 (contracts/ReefMooniswapV1.sol#16) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version^0.6.12 (contracts/ReefUniswap.sol#10) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version^0.6.12 (contracts/libraries/UniERC20.sol#5) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version>=0.6.0<0.8.0 (contracts/openzeppelin/contracts/GSN/Context.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (contracts/openzeppelin/contracts/access/Ownable.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (contracts/openzeppelin/contracts/math/SafeMath.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (contracts/openzeppelin/contracts/token/ERC20/IERC20.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (contracts/openzeppelin/contracts/token/ERC20/SafeERC20.sol#3) is too complex
Pragma version>=0.6.2<0.8.0 (contracts/openzeppelin/contracts/utils/Address.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (contracts/openzeppelin/contracts/utils/ReentrancyGuard.sol#3) is too complex
Pragma version^0.6.12 (contracts/utils/Babylonian.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version^0.6.12 (interfaces/IBFactory.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version^0.6.12 (interfaces/IBPool.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version^0.6.12 (interfaces/IMooniswap.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version^0.6.12 (interfaces/IUniswapV2Factory.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version^0.6.12 (interfaces/IUniswapV2Pair.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version^0.6.12 (interfaces/IUniswapV2Router.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version^0.6.12 (interfaces/TransferHelper.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
solc-0.6.12 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in ReefBasket._multiInvest(uint256[],uint256[],uint256) (contracts/ReefBasket.sol#365-393):
        - (success) = msg.sender.call{value: address(this).balance}() (contracts/ReefBasket.sol#392)
Low level call in ReefBasket.disinvest(uint256,uint256,uint256,bool) (contracts/ReefBasket.sol#530-746):
        - (success) = msg.sender.call{value: address(this).balance}() (contracts/ReefBasket.sol#745)
Low level call in UniERC20.uniSymbol(IERC20) (contracts/libraries/UniERC20.sol#47-84):
        - (success,data) = address(token).staticcall{gas: 20000}(abi.encodeWithSignature(symbol())) (contracts/libraries/UniERC20.sol#52-54)
        - (success,data) = address(token).staticcall{gas: 20000}(abi.encodeWithSignature(SYMBOL())) (contracts/libraries/UniERC20.sol#56-58)
Low level call in Address.sendValue(address,uint256) (contracts/openzeppelin/contracts/utils/Address.sol#53-59):
        - (success) = recipient.call{value: amount}() (contracts/openzeppelin/contracts/utils/Address.sol#57)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (contracts/openzeppelin/contracts/utils/Address.sol#114-121):
        - (success,returndata) = target.call{value: value}(data) (contracts/openzeppelin/contracts/utils/Address.sol#119)
Low level call in Address.functionStaticCall(address,bytes,string) (contracts/openzeppelin/contracts/utils/Address.sol#139-145):
        - (success,returndata) = target.staticcall(data) (contracts/openzeppelin/contracts/utils/Address.sol#143)
Low level call in Address.functionDelegateCall(address,bytes,string) (contracts/openzeppelin/contracts/utils/Address.sol#163-169):
        - (success,returndata) = target.delegatecall(data) (contracts/openzeppelin/contracts/utils/Address.sol#167)
Low level call in TransferHelper.safeApprove(address,address,uint256) (interfaces/TransferHelper.sol#4-17):
        - (success,data) = token.call(abi.encodeWithSelector(0x095ea7b3,to,value)) (interfaces/TransferHelper.sol#10-12)
Low level call in TransferHelper.safeTransfer(address,address,uint256) (interfaces/TransferHelper.sol#19-32):
        - (success,data) = token.call(abi.encodeWithSelector(0xa9059cbb,to,value)) (interfaces/TransferHelper.sol#25-27)
Low level call in TransferHelper.safeTransferFrom(address,address,address,uint256) (interfaces/TransferHelper.sol#34-48):
        - (success,data) = token.call(abi.encodeWithSelector(0x23b872dd,from,to,value)) (interfaces/TransferHelper.sol#41-43)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

```
INFO:Detectors:
Parameter ReefBalancer.investIntoBalancerPool(address,address,address,uint256,uint256)._toWhomToIssue (contracts/ReefBalancer.sol#64) is not in mixedCase
Parameter ReefBalancer.investIntoBalancerPool(address,address,address,uint256,uint256)._FromTokenContractAddress (contracts/ReefBalancer.sol#65) is not in mixedCase
Parameter ReefBalancer.investIntoBalancerPool(address,address,address,uint256,uint256)._ToBalancerPoolAddress (contracts/ReefBalancer.sol#66) is not in mixedCase
Parameter ReefBalancer.investIntoBalancerPool(address,address,address,uint256,uint256)._amount (contracts/ReefBalancer.sol#67) is not in mixedCase
Parameter ReefBalancer.investIntoBalancerPool(address,address,address,uint256,uint256)._minPoolTokens (contracts/ReefBalancer.sol#68) is not in mixedCase
Parameter ReefBalancer.disinvestFromBalancerPool(address,address,address,uint256,uint256)._toWhomToIssue (contracts/ReefBalancer.sol#133) is not in mixedCase
Parameter ReefBalancer.disinvestFromBalancerPool(address,address,address,uint256,uint256)._ToTokenContractAddress (contracts/ReefBalancer.sol#134) is not in mixedCase
Parameter ReefBalancer.disinvestFromBalancerPool(address,address,address,uint256,uint256)._FromBalancerPoolAddress (contracts/ReefBalancer.sol#135) is not in mixedCase
Parameter ReefBalancer.disinvestFromBalancerPool(address,address,address,uint256,uint256)._IncomingBPT (contracts/ReefBalancer.sol#136) is not in mixedCase
Parameter ReefBalancer.disinvestFromBalancerPool(address,address,address,uint256,uint256)._minTokensRec (contracts/ReefBalancer.sol#137) is not in mixedCase
Parameter ReefBalancer.getToken2BPT(address,uint256,address)._ToBalancerPoolAddress (contracts/ReefBalancer.sol#297) is not in mixedCase
Parameter ReefBalancer.getToken2BPT(address,uint256,address)._IncomingERC (contracts/ReefBalancer.sol#298) is not in mixedCase
Parameter ReefBalancer.getToken2BPT(address,uint256,address)._FromToken (contracts/ReefBalancer.sol#299) is not in mixedCase
Constant ReefBalancer.UniSwapV2FactoryAddress (contracts/ReefBalancer.sol#37-40) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ReefBalancer.uniswapRouter (contracts/ReefBalancer.sol#41-43) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ReefBalancer.BalancerFactory (contracts/ReefBalancer.sol#44-46) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ReefBalancer.wethTokenAddress (contracts/ReefBalancer.sol#48-49) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ReefBalancer.deadline (contracts/ReefBalancer.sol#51-52) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter ReefBasket.balanceOfUniswapPools(address,uint256)._owner (contracts/ReefBasket.sol#96) is not in mixedCase
Parameter ReefBasket.balanceOfUniswapPools(address,uint256)._basketIndex (contracts/ReefBasket.sol#96) is not in mixedCase
Parameter ReefBasket.balanceOfTokens(address,uint256)._owner (contracts/ReefBasket.sol#115) is not in mixedCase
Parameter ReefBasket.balanceOfTokens(address,uint256)._basketIndex (contracts/ReefBasket.sol#115) is not in mixedCase
Parameter ReefBasket.balanceOfBalancerPools(address,uint256)._owner (contracts/ReefBasket.sol#131) is not in mixedCase
Parameter ReefBasket.balanceOfBalancerPools(address,uint256)._basketIndex (contracts/ReefBasket.sol#131) is not in mixedCase
Parameter ReefBasket.balanceOfMooniswapPools(address,uint256)._owner (contracts/ReefBasket.sol#150) is not in mixedCase
Parameter ReefBasket.balanceOfMooniswapPools(address,uint256)._basketIndex (contracts/ReefBasket.sol#150) is not in mixedCase
Parameter ReefBasket.investedAmountInBasket(address,uint256)._owner (contracts/ReefBasket.sol#169) is not in mixedCase
Parameter ReefBasket.investedAmountInBasket(address,uint256)._basketIndex (contracts/ReefBasket.sol#169) is not in mixedCase
Parameter ReefBasket.getAvailableBasketUniswapPools(uint256)._basketIndex (contracts/ReefBasket.sol#177) is not in mixedCase
Parameter ReefBasket.getAvailableBasketTokens(uint256)._basketIndex (contracts/ReefBasket.sol#199) is not in mixedCase
Parameter ReefBasket.getAvailableBasketBalancerPools(uint256)._basketIndex (contracts/ReefBasket.sol#217) is not in mixedCase
Parameter ReefBasket.getAvailableBasketMooniswapPools(uint256)._basketIndex (contracts/ReefBasket.sol#239) is not in mixedCase
Parameter ReefBasket.createBasket(string,address[2][],uint8[],address[],uint8[],address[],uint8[],address[],uint8[])._name (contracts/ReefBasket.sol#262) is not in mixedCase
Parameter ReefBasket.createBasket(string,address[2][],uint8[],address[],uint8[],address[],uint8[],address[],uint8[])._uniswapPools (contracts/ReefBasket.sol#263) is not in mixedCase
Parameter ReefBasket.createBasket(string,address[2][],uint8[],address[],uint8[],address[],uint8[],address[],uint8[])._uniswapPoolsWeights (contracts/ReefBasket.sol#264) is not in mixedCase
Parameter ReefBasket.createBasket(string,address[2][],uint8[],address[],uint8[],address[],uint8[],address[],uint8[])._tokens (contracts/ReefBasket.sol#265) is not in mixedCase
Parameter ReefBasket.createBasket(string,address[2][],uint8[],address[],uint8[],address[],uint8[],address[],uint8[])._tokensWeights (contracts/ReefBasket.sol#266) is not in mixedCase
Parameter ReefBasket.createBasket(string,address[2][],uint8[],address[],uint8[],address[],uint8[],address[],uint8[])._balancerPools (contracts/ReefBasket.sol#267) is not in mixedCase
Parameter ReefBasket.createBasket(string,address[2][],uint8[],address[],uint8[],address[],uint8[],address[],uint8[])._balancerPoolsWeights (contracts/ReefBasket.sol#268) is not in mixedCase
Parameter ReefBasket.createBasket(string,address[2][],uint8[],address[],uint8[],address[],uint8[],address[],uint8[])._mooniswapPools (contracts/ReefBasket.sol#269) is not in mixedCase
Parameter ReefBasket.createBasket(string,address[2][],uint8[],address[],uint8[],address[],uint8[],address[],uint8[])._mooniswapPoolsWeights (contracts/ReefBasket.sol#270) is not in mixedCase
Parameter ReefBasket.invest(uint256[],uint256[],uint256)._basketIndexes (contracts/ReefBasket.sol#358) is not in mixedCase
Parameter ReefBasket.invest(uint256[],uint256[],uint256)._weights (contracts/ReefBasket.sol#359) is not in mixedCase
Parameter ReefBasket.invest(uint256[],uint256[],uint256)._minPoolTokens (contracts/ReefBasket.sol#360) is not in mixedCase
Parameter ReefBasket.disinvest(uint256,uint256,uint256,bool)._basketIndex (contracts/ReefBasket.sol#531) is not in mixedCase
Parameter ReefBasket.disinvest(uint256,uint256,uint256,bool)._percent (contracts/ReefBasket.sol#532) is not in mixedCase
Parameter ReefBasket.disinvest(uint256,uint256,uint256,bool)._protocolYieldRatio (contracts/ReefBasket.sol#533) is not in mixedCase
Parameter ReefBasket.setProtocolTokenDisinvestPercentage(uint16)._newPercentage (contracts/ReefBasket.sol#748) is not in mixedCase
Parameter ReefBasket.setProtocolTokenAddress(address)._newProtocolTokenAddress (contracts/ReefBasket.sol#759) is not in mixedCase

Parameter ReefBasket.setMinimalInvestment(uint256)._minimalInvestment (contracts/ReefBasket.sol#766) is not in mixedCase
Parameter ReefBasket.setMaxInvestedFunds(uint256)._maxInvestedFunds (contracts/ReefBasket.sol#770) is not in mixedCase
Parameter ReefBasket.inCaseTokensGetStuck(IERC20)._TokenAddress (contracts/ReefBasket.sol#787) is not in mixedCase
Parameter ReefMooniswapV1.investIntoMooniswapPool(address,address,uint256)._FromTokenContractAddress (contracts/ReefMooniswapV1.sol#24) is not in mixedCase
Parameter ReefMooniswapV1.investIntoMooniswapPool(address,address,uint256)._amount (contracts/ReefMooniswapV1.sol#26) is not in mixedCase
Parameter ReefMooniswapV1.disinvestFromMooniswapPool(address,address,uint256)._ToTokenContractAddress (contracts/ReefMooniswapV1.sol#98) is not in mixedCase
Parameter ReefMooniswapV1.disinvestFromMooniswapPool(address,address,uint256)._amount (contracts/ReefMooniswapV1.sol#100) is not in mixedCase
Parameter ReefUniswap.exchangeTokensV2(address,address,address,uint256)._FromTokenContractAddress (contracts/ReefUniswap.sol#190) is not in mixedCase
Parameter ReefUniswap.exchangeTokensV2(address,address,address,uint256)._ToUnipoolToken0 (contracts/ReefUniswap.sol#191) is not in mixedCase
Parameter ReefUniswap.exchangeTokensV2(address,address,address,uint256)._ToUnipoolToken1 (contracts/ReefUniswap.sol#192) is not in mixedCase
Parameter ReefUniswap.exchangeTokensV2(address,address,address,uint256)._amount (contracts/ReefUniswap.sol#193) is not in mixedCase
Parameter ReefUniswap.canSwapFromV2(address,address)._fromToken (contracts/ReefUniswap.sol#238) is not in mixedCase
Parameter ReefUniswap.canSwapFromV2(address,address)._toToken (contracts/ReefUniswap.sol#238) is not in mixedCase
Parameter ReefUniswap.swapFromV2(address,address,uint256)._fromToken (contracts/ReefUniswap.sol#306) is not in mixedCase
Parameter ReefUniswap.swapFromV2(address,address,uint256)._toToken (contracts/ReefUniswap.sol#307) is not in mixedCase
Constant ReefUniswap.uniswapV2RouterAddress (contracts/ReefUniswap.sol#16) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ReefUniswap.wethTokenAddress (contracts/ReefUniswap.sol#17) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ReefUniswap.uniswapV2Router (contracts/ReefUniswap.sol#19-21) is not in UPPER_CASE_WITH_UNDERSCORES
Constant ReefUniswap.UniSwapV2FactoryAddress (contracts/ReefUniswap.sol#23-25) is not in UPPER_CASE_WITH_UNDERSCORES
Function IUniswapV2Router.WETH() (interfaces/IUniswapV2Router.sol#8) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformity-to-solidity-naming-conventions
INFO:Detectors:
ReefBalancer.slitherConstructorConstantVariables() (contracts/ReefBalancer.sol#32-611) uses literals with too many digits:
        - deadline = 0xf0000000000000000000000000000000000000000000000000000000000000 (contracts/ReefBalancer.sol#51-52)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
ReefBasket.wethTokenAddress (contracts/ReefBasket.sol#86) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
```

FINDINGS & TECH DETAILS

```
INFO:Detectors:
investIntoBalancerPool(address,address,address,uint256,uint256) should be declared external:
        - ReefBalancer.investIntoBalancerPool(address,address,address,uint256,uint256) (contracts/ReefBalancer.sol#63-122)
disinvestFromBalancerPool(address,address,address,uint256,uint256) should be declared external:
        - ReefBalancer.disinvestFromBalancerPool(address,address,address,uint256,uint256) (contracts/ReefBalancer.sol#132-168)
balanceOfUniswapPools(address,uint256) should be declared external:
        - ReefBasket.balanceOfUniswapPools(address,uint256) (contracts/ReefBasket.sol#96-113)
balanceOfTokens(address,uint256) should be declared external:
        - ReefBasket.balanceOfTokens(address,uint256) (contracts/ReefBasket.sol#115-129)
balanceOfBalancerPools(address,uint256) should be declared external:
        - ReefBasket.balanceOfBalancerPools(address,uint256) (contracts/ReefBasket.sol#131-148)
balanceOfMooniswapPools(address,uint256) should be declared external:
        - ReefBasket.balanceOfMooniswapPools(address,uint256) (contracts/ReefBasket.sol#150-167)
investedAmountInBasket(address,uint256) should be declared external:
        - ReefBasket.investedAmountInBasket(address,uint256) (contracts/ReefBasket.sol#169-175)
getAvailableBasketUniswapPools(uint256) should be declared external:
        - ReefBasket.getAvailableBasketUniswapPools(uint256) (contracts/ReefBasket.sol#177-197)
getAvailableBasketTokens(uint8) should be declared external:
        - ReefBasket.getAvailableBasketTokens(uint8) (contracts/ReefBasket.sol#199-215)
getAvailableBasketBalancerPools(uint256) should be declared external:
        - ReefBasket.getAvailableBasketBalancerPools(uint256) (contracts/ReefBasket.sol#217-237)
getAvailableBasketMooniswapPools(uint256) should be declared external:
        - ReefBasket.getAvailableBasketMooniswapPools(uint256) (contracts/ReefBasket.sol#239-259)
createBasket(string,address[2][],uint8[],address[],uint8[],address[],uint8[],address[],uint8[]) should be declared external:
        - ReefBasket.createBasket(string,address[2][],uint8[],address[],uint8[],address[],uint8[],address[],uint8[]) (contracts/ReefBasket.sol#261-348)
invest(uint256[],uint256[],uint256) should be declared external:
        - ReefBasket.invest(uint256[],uint256[],uint256) (contracts/ReefBasket.sol#357-363)
disinvest(uint256,uint256,uint256,bool) should be declared external:
        - ReefBasket.disinvest(uint256,uint256,uint256,bool) (contracts/ReefBasket.sol#530-746)
setProtocolTokenDisinvestPercentage(uint16) should be declared external:
        - ReefBasket.setProtocolTokenDisinvestPercentage(uint16) (contracts/ReefBasket.sol#748-757)
setProtocolTokenAddress(address) should be declared external:
        - ReefBasket.setProtocolTokenAddress(address) (contracts/ReefBasket.sol#759-764)
setMinimalInvestment(uint256) should be declared external:
        - ReefBasket.setMinimalInvestment(uint256) (contracts/ReefBasket.sol#766-768)
setMaxInvestedFunds(uint256) should be declared external:
        - ReefBasket.setMaxInvestedFunds(uint256) (contracts/ReefBasket.sol#770-776)
inCaseTokengetsStuck(IERC20) should be declared external:
        - ReefBasket.inCaseTokengetsStuck(IERC20) (contracts/ReefBasket.sol#787-790)
toggleContractActive() should be declared external:
        - ReefBasket.toggleContractActive() (contracts/ReefBasket.sol#793-795)
withdraw() should be declared external:
        - ReefBasket.withdraw() (contracts/ReefBasket.sol#798-802)
investIntoMooniswapPool(address,address,uint256) should be declared external:
        - ReefMooniswapV1.investIntoMooniswapPool(address,address,uint256) (contracts/ReefMooniswapV1.sol#23-95)
disinvestFromMooniswapPool(address,address,uint256) should be declared external:
        - ReefMooniswapV1.disinvestFromMooniswapPool(address,address,uint256) (contracts/ReefMooniswapV1.sol#97-126)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (contracts/openzeppelin/contracts/access/Ownable.sol#54-57)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (contracts/openzeppelin/contracts/access/Ownable.sol#63-67)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

MythX:

Report for ReefLiquidityBond.sol
https://dashboard.mythx.io/#/console/analyses/7c4d3874-0a38-4123-afd5-808ef1bb511a

Line	SWC Title	Severity	Short Description
129	(SWC-000) Unknown	Medium	Function could be marked as external.
137	(SWC-000) Unknown	Medium	Function could be marked as external.

Report for contracts/ReefBasket.sol
https://dashboard.mythx.io/#/console/analyses/bbcc2c25-9360-4b2d-84e4-cd1d389529c4

Line	SWC Title	Severity	Short Description
13	(SWC-103) Floating Pragma	Low	A floating pragma is set.
15	(SWC-123) Requirement Violation	Low	Requirement violation.
392	(SWC-131) Presence of unused variables	Low	Unused local variable "success".
788	(SWC-123) Requirement Violation	Low	Requirement violation.
801	(SWC-134) Message call with hardcoded gas amount	Low	Call with hardcoded gas amount.

Report for contracts/ReefUniswap.sol
https://dashboard.mythx.io/#/console/analyses/bbcc2c25-9360-4b2d-84e4-cd1d389529c4

Line	SWC Title	Severity	Short Description
146	(SWC-131) Presence of unused variables	Low	Unused local variable "_wethToken".

Report for contracts/libraries/UniERC20.sol
https://dashboard.mythx.io/#/console/analyses/bbcc2c25-9360-4b2d-84e4-cd1d389529c4

Line	SWC Title	Severity	Short Description
26	(SWC-134) Message call with hardcoded gas amount	Low	Call with hardcoded gas amount.
39	(SWC-134) Message call with hardcoded gas amount	Low	Call with hardcoded gas amount.
52	(SWC-134) Message call with hardcoded gas amount	Low	Call with hardcoded gas amount.
56	(SWC-134) Message call with hardcoded gas amount	Low	Call with hardcoded gas amount.

Report for ReefToken.sol
https://dashboard.mythx.io/#/console/analyses/0163859c-9d62-4702-a39f-455cb4634323

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.
12	(SWC-000) Unknown	Medium	Function could be marked as external.
95	(SWC-128) DoS With Block Gas Limit	Low	Potentially unbounded data structure passed to builtin.
121	(SWC-116) Timestamp Dependence	Low	A control flow decision is made based on The block.timestamp environment variable.
151	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	A control flow decision is made based on The block.number environment variable.
151	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randomness.
170	(SWC-128) DoS With Block Gas Limit	Low	Loop over unbounded data structure.
224	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randomness.

3.6 ERC CONFORMAL CHECKER - INFORMATIONAL

Description:

Another Slither tool can test ERC token functions. Thus slither-check-erc was performed over ReefToken:

```
# Check ReefToken

## Check functions
[✓] totalSupply() is present
        [✓] totalSupply() -> () (correct return value)
        [✓] totalSupply() is view
[✓] balanceOf(address) is present
        [✓] balanceOf(address) -> () (correct return value)
        [✓] balanceOf(address) is view
[✓] transfer(address,uint256) is present
        [✓] transfer(address,uint256) -> () (correct return value)
        [✓] Transfer(address,address,uint256) is emitted
[✓] transferFrom(address,address,uint256) is present
        [✓] transferFrom(address,address,uint256) -> () (correct return value)
        [✓] Transfer(address,address,uint256) is emitted
[✓] approve(address,uint256) is present
        [✓] approve(address,uint256) -> () (correct return value)
        [✓] Approval(address,address,uint256) is emitted
[✓] allowance(address,address) is present
        [✓] allowance(address,address) -> () (correct return value)
        [✓] allowance(address,address) is view
[✓] name() is present
        [✓] name() -> () (correct return value)
        [✓] name() is view
[✓] symbol() is present
        [✓] symbol() -> () (correct return value)
        [✓] symbol() is view
[✓] decimals() is present
        [✓] decimals() -> () (correct return value)
        [✓] decimals() is view

## Check events
[✓] Transfer(address,address,uint256) is present
        [✓] parameter 0 is indexed
        [✓] parameter 1 is indexed
[✓] Approval(address,address,uint256) is present
        [✓] parameter 0 is indexed
        [✓] parameter 1 is indexed


        [✓] ReefToken has increaseAllowance(address,uint256)
```

Result: All tests are successfully passed.

THANK YOU FOR CHOOSING

// HALBORN