



Irrigation - Protocol

Smart Contract Security Assessment

Prepared by: **Halborn**

Date of Engagement: June 23rd, 2023 - August 4th, 2023

Visit: Halborn.com

DOCUMENT REVISION HISTORY	5
CONTACTS	6
1 EXECUTIVE OVERVIEW	7
1.1 INTRODUCTION	8
1.2 ASSESSMENT SUMMARY	9
1.3 TEST APPROACH & METHODOLOGY	10
2 RISK METHODOLOGY	11
2.1 EXPLOITABILITY	12
2.2 IMPACT	13
2.3 SEVERITY COEFFICIENT	15
2.4 SCOPE	17
3 ASSESSMENT SUMMARY & FINDINGS OVERVIEW	19
4 FINDINGS & TECH DETAILS	20
4.1 (HAL-01) SPRINKLERUPGRADEABLE CONTRACT CAN BE DRAINED THROUGH THE EXCHANGETOWATER FUNCTION - CRITICAL(10)	22
Description	22
Proof of Concept	25
BVSS	26
Recommendation	26
Remediation Plan	26
4.2 (HAL-02) AUCTIONS AND BIDS COULD BE STUCK PERMANENTLY IN THE AUCTIONUPGRADEABLE CONTRACT IF A BID IS PLACED BY A BLACKLISTED USDC/USDT USER - CRITICAL(10)	27
Description	27
Proof of Concept	28

BVSS	28
Recommendation	28
Remediation Plan	29
4.3 (HAL-03) BIDS CAN BE DOSED BY PLACING A VERY HIGH BID ON A VERY SMALL BIDAMOUNT - MEDIUM(5.0)	30
Description	30
Proof of Concept	31
BVSS	31
Recommendation	31
Remediation Plan	31
4.4 (HAL-04) LATESTANSWER CALL MAY RETURN STALE RESULTS - MEDIUM(6.2)	32
Description	32
BVSS	32
Recommendation	33
Remediation Plan	33
4.5 (HAL-05) USER COULD CANCEL THE REST OF BIDS OF AN AUCTION BY DOING 200 DIFFERENT BIDS - MEDIUM(5.6)	34
Description	34
BVSS	36
Recommendation	37
Remediation Plan	37
4.6 (HAL-06) SWAPETHFORWATER CALL CAN BE SANDWICCHED - MEDIUM(5.0)	38
Description	38
BVSS	39
Recommendation	40

Remediation Plan	40
4.7 (HAL-07) AUTOIRRIGATECALL WILL ALWAYS REVERT WITH OVERFLOW IF ITS CALLED WITH THE FULL REWARDAMOUNT - LOW(2.5)	41
Description	41
BVSS	41
Recommendation	42
Remediation Plan	42
4.8 (HAL-08) LACK OF A DOUBLE-STEP TRANSFEROWNERSHIP PATTERN - INFORMATONAL(0.0)	43
Description	43
BVSS	43
Recommendation	43
Remediation Plan	45
4.9 (HAL-09) CALLING SETMIDDLEASSET WOULD CAUSE THE IRRIGATION BONUS TO ALWAYS BE ZERO - INFORMATIONAL(0.0)	46
Description	46
BVSS	47
Recommendation	47
Remediation Plan	47
5 RECOMMENDATIONS OVERVIEW	48
6 AUTOMATED TESTING	50
6.1 STATIC ANALYSIS REPORT	51
Description	51
Slither results	51
6.2 AUTOMATED SECURITY SCAN	67

Description	67
MythX results	67

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE
0.1	Document Creation	06/23/2023
0.2	Document Updates	06/23/2023
0.3	Document Updates	08/04/2023
0.4	Draft Review	08/04/2023
0.5	Draft Review	08/04/2023
1.0	Remediation Plan	09/25/2023
1.1	Remediation Plan Review	09/25/2023
1.2	Remediation Plan Review	09/25/2023

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Gokberk Gulgún	Halborn	Gokberk.Gulgún@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Irrigation Protocol engaged Halborn to conduct a security assessment on their smart contracts beginning on June 23rd, 2023 and ending on August 4th, 2023. The security assessment was scoped to the smart contracts provided in the following GitHub repositories:

- [IrrigationProtocol/irrigation-contracts-diamond](https://github.com/IrrigationProtocol/irrigation-contracts-diamond).

1.2 ASSESSMENT SUMMARY

The team at Halborn was provided six weeks for the engagement and assigned a full-time security engineer to verify the security of the smart contracts. The security engineer is a blockchain and smart-contract security expert with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this assessment is to:

- Ensure that smart contract functions operate as intended.
- Identify potential security issues with the smart contracts.

In summary, Halborn identified some security risks that were mostly addressed by the Irrigation Protocol team.

1.3 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 this assessment. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of the code and can quickly identify items that do not follow the security best practices. The following phases and associated tools were used during the assessment:

- Research into architecture and purpose.
- Smart contract manual code review 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.
- Manual testing by custom scripts.
- Scanning of solidity files for vulnerabilities, security hot-spots or bugs. ([MythX](#))
- Static Analysis of security for scoped contract, and imported functions. ([Slither](#))
- Testnet deployment. ([Foundry](#))

2. RISK METHODOLOGY

Every vulnerability and issue observed by Halborn is ranked based on **two sets of Metrics** and a **Severity Coefficient**. This system is inspired by the industry standard Common Vulnerability Scoring System.

The two **Metric sets** are: **Exploitability** and **Impact**. **Exploitability** captures the ease and technical means by which vulnerabilities can be exploited and **Impact** describes the consequences of a successful exploit.

The **Severity Coefficients** is designed to further refine the accuracy of the ranking with two factors: **Reversibility** and **Scope**. These capture the impact of the vulnerability on the environment as well as the number of users and smart contracts affected.

The final score is a value between 0-10 rounded up to 1 decimal place and 10 corresponding to the highest security risk. This provides an objective and accurate rating of the severity of security vulnerabilities in smart contracts.

The system is designed to assist in identifying and prioritizing vulnerabilities based on their level of risk to address the most critical issues in a timely manner.

2.1 EXPLOITABILITY

Attack Origin (AO):

Captures whether the attack requires compromising a specific account.

Attack Cost (AC):

Captures the cost of exploiting the vulnerability incurred by the attacker relative to sending a single transaction on the relevant blockchain. Includes but is not limited to financial and computational cost.

Attack Complexity (AX):

Describes the conditions beyond the attacker's control that must exist in order to exploit the vulnerability. Includes but is not limited to macro situation, available third-party liquidity and regulatory challenges.

Metrics:

Exploitability Metric (m_E)	Metric Value	Numerical Value
Attack Origin (AO)	Arbitrary (AO:A)	1
	Specific (AO:S)	0.2
Attack Cost (AC)	Low (AC:L)	1
	Medium (AC:M)	0.67
	High (AC:H)	0.33
Attack Complexity (AX)	Low (AX:L)	1
	Medium (AX:M)	0.67
	High (AX:H)	0.33

Exploitability E is calculated using the following formula:

$$E = \prod m_e$$

2.2 IMPACT

Confidentiality (C):

Measures the impact to the confidentiality of the information resources managed by the contract due to a successfully exploited vulnerability. Confidentiality refers to limiting access to authorized users only.

Integrity (I):

Measures the impact to integrity of a successfully exploited vulnerability. Integrity refers to the trustworthiness and veracity of data stored and/or processed on-chain. Integrity impact directly affecting Deposit or Yield records is excluded.

Availability (A):

Measures the impact to the availability of the impacted component resulting from a successfully exploited vulnerability. This metric refers to smart contract features and functionality, not state. Availability impact directly affecting Deposit or Yield is excluded.

Deposit (D):

Measures the impact to the deposits made to the contract by either users or owners.

Yield (Y):

Measures the impact to the yield generated by the contract for either users or owners.

Metrics:

Impact Metric (m_I)	Metric Value	Numerical Value
Confidentiality (C)	None (I:N)	0
	Low (I:L)	0.25
	Medium (I:M)	0.5
	High (I:H)	0.75
	Critical (I:C)	1
Integrity (I)	None (I:N)	0
	Low (I:L)	0.25
	Medium (I:M)	0.5
	High (I:H)	0.75
	Critical (I:C)	1
Availability (A)	None (A:N)	0
	Low (A:L)	0.25
	Medium (A:M)	0.5
	High (A:H)	0.75
	Critical	1
Deposit (D)	None (D:N)	0
	Low (D:L)	0.25
	Medium (D:M)	0.5
	High (D:H)	0.75
	Critical (D:C)	1
Yield (Y)	None (Y:N)	0
	Low (Y:L)	0.25
	Medium (Y:M)	0.5
	High (Y:H)	0.75
	Critical (Y:H)	1

Impact I is calculated using the following formula:

$$I = \max(m_I) + \frac{\sum m_I - \max(m_I)}{4}$$

2.3 SEVERITY COEFFICIENT

Reversibility (R):

Describes the share of the exploited vulnerability effects that can be reversed. For upgradeable contracts, assume the contract private key is available.

Scope (S):

Captures whether a vulnerability in one vulnerable contract impacts resources in other contracts.

Coefficient (C)	Coefficient Value	Numerical Value
Reversibility (r)	None (R:N)	1
	Partial (R:P)	0.5
	Full (R:F)	0.25
Scope (s)	Changed (S:C)	1.25
	Unchanged (S:U)	1

Severity Coefficient C is obtained by the following product:

$$C = rs$$

The Vulnerability Severity Score S is obtained by:

$$S = \min(10, EIC * 10)$$

The score is rounded up to 1 decimal places.

Severity	Score Value Range
Critical	9 - 10
High	7 - 8.9
Medium	4.5 - 6.9
Low	2 - 4.4
Informational	0 - 1.9

2.4 SCOPE

1. IN-SCOPE TREE & COMMIT :

The security assessment was scoped to the following smart contracts:

GitHub repository: [IrrigationProtocol/irrigation-contracts-diamond](#)

Commit ID: [159927ff1e8f8e212cd09894c29c1019e5d417f6](#)

Remediation Commit ID: [1b9420fb77d856bb57a35e947ae255b035e3037c](#)

Smart contracts in scope:

- SprinklerUpgradeable.sol
- WaterTowerUpgradeable.sol
- TrancheBondUpgradeable.sol
- AuctionUpgradeable.sol
- ERC1155WhitelistUpgradeable.sol
- WaterCommonUpgradeable.sol
- PriceOracleUpgradeable.sol
- PodsOracleUpgradeable.sol
- WaterUpgradeable.sol
- SprinklerStorage.sol
- WaterTowerStorage.sol
- TrancheBondStorage.sol
- AuctionStorage.sol
- ERC1155WhitelistStorage.sol
- WaterCommonStorage.sol
- PriceOracleStorage.sol
- IrrigationDiamond.sol
- Diamond.sol
- DiamondCutFacet.sol
- DiamondLoupeFacet.sol
- EIP2535Initializable.sol
- IrrigationAccessControl.sol
- OwnershipFacet.sol
- FullMath.sol
- PodTransferHelper.sol
- LibPrice.sol

EXECUTIVE OVERVIEW

- BeanPriceOracle.sol
- ChainlinkOracle.sol
- UniswapV3Twap.sol

3. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

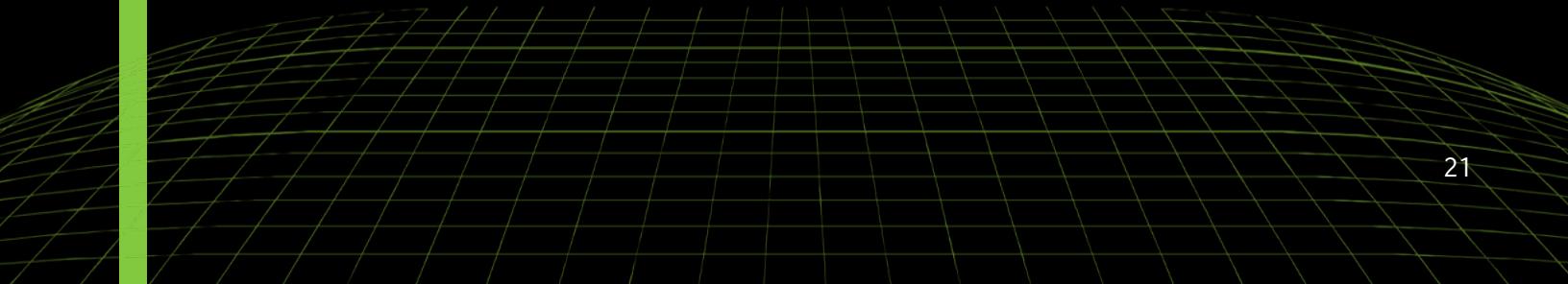
CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
2	0	4	1	2

EXECUTIVE OVERVIEW

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
(HAL-01) SPRINKLERUPGRADEABLE CONTRACT CAN BE DRAINED THROUGH THE EXCHANGETOWATER FUNCTION	Critical (10)	SOLVED - 08/21/2023
(HAL-02) AUCTIONS AND BIDS COULD BE STUCK PERMANENTLY IN THE AUCTIONUPGRADEABLE CONTRACT IF A BID IS PLACED BY A BLACKLISTED USDC/USDT USER	Critical (10)	SOLVED - 08/21/2023
(HAL-03) BIDS CAN BE DOSED BY PLACING A VERY HIGH BID ON A VERY SMALL BIDAMOUNT	Medium (5.0)	SOLVED - 08/21/2023
(HAL-04) LATESTANSWER CALL MAY RETURN STALE RESULTS	Medium (6.2)	SOLVED - 08/21/2023
(HAL-05) USER COULD CANCEL THE REST OF BIDS OF AN AUCTION BY DOING 200 DIFFERENT BIDS	Medium (5.6)	SOLVED - 08/21/2023
(HAL-06) SWAPETHFORWATER CALL CAN BE SANDWICCHED	Medium (5.0)	SOLVED - 08/21/2023
(HAL-07) AUTOIRRIGATECALL WILL ALWAYS REVERT WITH OVERFLOW IF ITS CALLED WITH THE FULL REWARDAMOUNT	Low (2.5)	RISK ACCEPTED
(HAL-08) LACK OF A DOUBLE-STEP TRANSFEROWNERSHIP PATTERN	Informational (0.0)	ACKNOWLEDGED
(HAL-09) CALLING SETMIDDLEASSET WOULD CAUSE THE IRRIGATION BONUS TO ALWAYS BE ZERO	Informational (0.0)	ACKNOWLEDGED



FINDINGS & TECH DETAILS



4.1 (HAL-01) SPRINKLERUPGRADEABLE CONTRACT CAN BE DRAINED THROUGH THE EXCHANGETOWATER FUNCTION - CRITICAL(10)

Commit IDs affected:

- 159927ff1e8f8e212cd09894c29c1019e5d417f6

Description:

In the `SprinklerUpgradeable` contract, a set of assets are whitelisted which can be used to be swapped for WATER. The assets whitelisted can be seen [here](#):

- Native Ether represented by the address:

`0xEeeeeEeeeEeEeeEeEeeEEeeeeEeeeeEEeE`

- USDC
- DAI
- USDT
- LUSD
- BEAN
- ROOT
- SPOT
- OHM
- PAXG
- CNHT

Native Ether, represented by the `0xEeeeeEeeeEeEeeEeEeeEEeeeeEeeeeEEeE` address, is used to check the multiplier assigned to the Native Ether exchanges:

Listing 1: SprinklerUpgradeable.sol (Line 100)

```
94 /**
95 * @notice Exchange ETH to water
96 * @return waterAmount received water amount
97 */
```

```

98 function exchangeETHToWater() external payable nonReentrant
99     returns (uint256 waterAmount) {
100     require(msg.value != 0, "Invalid amount");
101     waterAmount = getWaterAmount(Constants.ETHER, msg.value);
102     if (waterAmount > sprinkleableWater()) revert
103     InsufficientWater();
104     require(waterAmount != 0, "No water output"); // if price is 0
105     or tokenMultiplier is 0, amount can be 0
106     transferWater(waterAmount);
107     SprinklerStorage.layout().reserves[Constants.ETHER] += msg.
108     value;
109     emit WaterExchanged(msg.sender, Constants.ETHER, msg.value,
110     waterAmount, false);
111 }
```

Listing 2: SprinklerUpgradeable.sol (Line 155)

```

151 function getWaterAmount(
152     address _token,
153     uint256 _amount
154 ) public view returns (uint256 waterAmount) {
155     uint256 multiplier = tokenMultiplier(_token);
156     uint256 tokenPrice = IPriceOracleUpgradeable(address(this)).
157     getPrice(_token);
158     uint256 waterPrice = IPriceOracleUpgradeable(address(this)).
159     getWaterPrice();
160     waterAmount = (_amount * tokenPrice * multiplier) / waterPrice
161     ;
162 }
```

The function `exchangeTokenToWater()` is used to exchange whitelisted assets for WATER:

Listing 3: SprinklerUpgradeable.sol (Lines 80,88)

```

71 /**
72  * @notice Exchange whitelisted asset(BEAN, BEAN:3CRV, Spot, and
73  * so on) to water
74  * @param token source token address
75  * @param amount source token amount
76  * @return waterAmount received water amount
77 */
```

```

77 function exchangeTokenToWater(
78     address token,
79     uint256 amount
80 ) external onlyListedAsset(token) nonReentrant returns (uint256
↳ waterAmount) {
81     require(token != address(this), "Invalid token");
82     require(amount != 0, "Invalid amount");
83
84     waterAmount = getWaterAmount(token, amount);
85     if (waterAmount > sprinkleableWater()) revert
↳ InsufficientWater();
86     require(waterAmount != 0, "No water output"); // if price is
↳ 0, amount can be 0
87
88     TransferHelper.safeTransferFrom(token, msg.sender, address(
↳ this), amount);
89     transferWater(waterAmount);
90     SprinklerStorage.layout().reserves[token] += amount;
91     emit WaterExchanged(msg.sender, token, amount, waterAmount,
↳ false);
92 }
```

This function makes use of the `TransferHelper` library:

Listing 4: TransferHelper.sol (Line 21)

```

18 function safeTransferFrom(address token, address from, address to,
↳ uint value) internal {
19     // bytes4(keccak256(bytes('transferFrom(address,address,
↳ uint256)')));
20     (bool success, bytes memory data) = token.call(abi.
↳ encodeWithSelector(0x23b872dd, from, to, value));
21     require(success && (data.length == 0 || abi.decode(data, (bool
↳ )), 'STF'));
22 }
```

Although, this library, unlike OpenZeppelin's `SafeERC20`, does not check that the `token` address is actually a smart contract with some code deployed in that address (`&& address(token).code.length > 0`):

Listing 5: SafeERC20.sol (Line 14)

Based on this and considering that the `0xEeeeeEeeeEeEeeEeEeeEEeeeeEeeeeeeeEEeE` address is a representative address with no actual code deployed in it, a malicious user could call `SprinklerUpgradeable.exchangeTokenToWater("0xEeeeeEeeeEeEeeEeEeeEEeeeeEeeeeeeeEEeE", <very high amount>)` and drain all the WATER tokens from the contract.

Proof of Concept:

Exploit:

test exchangeEther

```
WaterUpgradeable(irrigationdiamond).balanceOf(user1) -> 0
```

USER1 CALLS -> SprinklerUpgradeable(irrigationdiamond).exchangeTokenToWater(address(ETHER), 1e18)
WaterUpgradeable(irrigationdiamond).balanceOf(user1) -> 381021632000000000000000

Debugged call:

BVSS:

A0:A/AC:L/AX:L/C:N/I:N/A:N/D:C/Y:N/R:N/S:U (10)

Recommendation:

It is recommended to use the `OpenZeppelin's SafeERC20` library to perform all the token transfers instead of the `TransferHelper` library.

Remediation Plan:

SOLVED: The Irrigation Protocol team solved the issue by implementing the recommended solution.

Commit ID : 2a75e858618864b3962c623035cca5b5aef4ff4d.

4.2 (HAL-02) AUCTIONS AND BIDS COULD BE STUCK PERMANENTLY IN THE AUCTIONUPGRADEABLE CONTRACT IF A BID IS PLACED BY A BLACKLISTED USDC/USDT USER - CRITICAL(10)

Commit IDs affected:

- 159927ff1e8f8e212cd09894c29c1019e5d417f6

Description:

The `AuctionUpgradeable` contract allows creating different auctions through the `createAuction()` function:

Listing 6: AuctionUpgradeable.sol

```
100 function createAuction(
101     uint96 startTime,
102     uint96 duration,
103     address sellToken,
104     uint256 trancheIndex,
105     uint128 sellAmount,
106     uint128 minBidAmount,
107     uint128 fixedPrice,
108     uint128 priceRangeStart,
109     uint128 priceRangeEnd,
110     AuctionType auctionType
111 ) external payable returns (uint256)
```

Once an auction is completed, the `closeAuction()` function will be called and will distribute the `sellToken` between all the bidders. As currently USDC contains a blacklist of users which cannot transfer or receive this token, the following issue could occur:

1. Alice creates an auction of 1000 USDC(`sellToken`).
2. Users place different bids for the USDC auction.

3. A Blacklisted USDC user also places a bid for the USDC auction. The bid is placed using a whitelisted `purchaseToken` like DAI.
 4. The auction is completed. `closeAuction()` is called, but it reverts every time as the USDC transfer to the blacklisted USDC user is forbidden.
 5. The auction USDC `sellTokens` are now stuck in the contract. All the bids are also stuck permanently in the contract.
- Similar issue can occur with USDT as it also contains a blacklist of users.

Proof of Concept:

```

USER1 CALLS ---> < USDC.approve(irrigationdiamond, 2000e6) >
USER1 CALLS ---> < AuctionUpgradeable(irrigationdiamond).createAuction(uint96(block.timestamp), uint96(86400 * 7), address(USDC), 0, 1000e6, 1, 1e18, 1, 2e18, AuctionType.TimedAuction) >
USER2 CALLS ---> < DAI.approve(irrigationdiamond, 1000000e18) >
USDC_BLACKLISTED_USER CALLS ---> < DAI.approve(irrigationdiamond, 1000000e18) >
USER2 CALLS ---> < AuctionUpgradeable(irrigationdiamond).placeBid(1, 10000, address(DAI), 10000) >
USER2 CALLS ---> < AuctionUpgradeable(irrigationdiamond).placeBid(1, 20000, address(DAI), 20000) >
USDC_BLACKLISTED_USER CALLS ---> < AuctionUpgradeable(irrigationdiamond).placeBid(1, 30000, address(DAI), 30000) >
7 DAYS LATER...
USER2 CALLS ---> < AuctionUpgradeable(irrigationdiamond).closeAuction(1) >
    [0] ←: prank(0x88C0e901bd1fd1a778dA342f0d2210fDC71CeF6B)
    [1] ←: ()
    [15266] IrrigationDiamond::closeAuction(1) [delegatecall]
        [15626] AuctionUpgradeable::closeAuction(1) [delegatecall]
            [3794] 0xA0b6691c6218b36c1d190a2e9Ebce3606e848::transfer(0xeefef8E968a0dB92781ac7B3B7c821909eF10c88, 30000)
            [2986] 0xa2327a938fFeFFC13baCFb16Ae10FcBc4cbDF::transfer(0xefeeF8E968a0dB92781ac7B3B7c821909eF10c88, 30000) [delegatecall]
                [2] ←: "Blacklistable: account is blacklisted"
                [3] ←: "Blacklistable: account is blacklisted"
                    [4] ←: "ST"
                    [5] ←: "ST"

```

BVSS:

A0:A/AC:L/AX:L/C:N/I:N/A:N/D:C/Y:N/R:N/S:U (10)

Recommendation:

Consider adding a try/catch code block for the `safeTransfer()` call in the `AuctionUpgradeable._settleBid()` function.

FINDINGS & TECH DETAILS

Remediation Plan:

SOLVED: The Irrigation Protocol team solved the issue by enforcing a whitelist on the `sellTokens`. The whitelist will not include blacklistable tokens like USDC/USDT.

Commit ID : `cae1d1718ef504f4fc27ba1aa464600a71ae635a`.

4.3 (HAL-03) BIDS CAN BE DOSED BY PLACING A VERY HIGH BID ON A VERY SMALL BIDAMOUNT - MEDIUM (5.0)

Commit IDs affected:

- [159927ff1e8f8e212cd09894c29c1019e5d417f6](#)

Description:

As mentioned before, the `AuctionUpgradeable` contract allows creating different auctions through the `createAuction()` function:

Listing 7: AuctionUpgradeable.sol (Line 106)

```

100 function createAuction(
101     uint96 startTime,
102     uint96 duration,
103     address sellToken,
104     uint256 trancheIndex,
105     uint128 sellAmount,
106     uint128 minBidAmount,
107     uint128 fixedPrice,
108     uint128 priceRangeStart,
109     uint128 priceRangeEnd,
110     AuctionType auctionType
111 ) external payable returns (uint256)

```

The `minBidAmount` parameter sets the minimum amount of the token that can be bid. In the case that the auctioneer sets the `minBidAmount` to a very low value, i.e. 1, the following attack vector would be possible:

1. Alice (auctioneer) creates an auction of 1000 USDC tokens and sets `minBidAmount` to 1. The `auctionType` is a `TimedAuction`.
2. Bob calls `placeBid(1, 1, DAI, 1000e18)` bidding 0.01 DAI for 0.000001 USDC. (`bidPrice = 1000 DAI`).
3. Joe calls `placeBid(1, 100e6, DAI, 1e18)` bidding 100 DAI for 100 USDC. (`bidPrice = 1 DAI`) but it reverts with a `low Bid` error.

4. If any user wants to bid now, they should place a bid higher than the abusive bid placed by Bob. ($\text{bidPrice} = 1000 \text{ DAI}$). Bob managed to cause a Denial of Service, losing only 0.01 DAI.

Proof of Concept:

BVSS:

A0:A/AC:L/AX:L/C:N/I:M/A:N/D:N/Y:N/R:N/S:U (5.0)

Recommendation:

It is recommended to set a minimum `minBidAmount` threshold of, for example 1% of the total `sellAmount`.

Remediation Plan:

SOLVED: The Irrigation Protocol team solved the issue by implementing the recommended solution.

Commit ID : 0a94a93c0f191c992e9f75a41d630ae3dbf80de8.

4.4 (HAL-04) LATESTANSWER CALL MAY RETURN STALE RESULTS - MEDIUM (6.2)

Commit IDs affected:

- 159927ff1e8f8e212cd09894c29c1019e5d417f6

Description:

In the `ChainlinkOracle` contract, the function `getChainlinkPrice()` is used to retrieve prices from different Chainlink aggregators:

Listing 8: ChainlinkOracle.sol (Lines 35,37)

```
29 /// @dev returns price with decimals 18
30 function getChainlinkPrice(AggregatorV2V3Interface feed) internal
↳ view returns (uint256) {
31     // Chainlink USD-denominated feeds store answers at 8 decimals
32     uint256 decimalDelta = uint256(18) - feed.decimals();
33     // Ensure that we don't multiply the result by 0
34     if (decimalDelta > 0) {
35         return uint256(feed.latestAnswer()) * 10 ** decimalDelta;
36     } else {
37         return uint256(feed.latestAnswer());
38     }
39 }
```

According to the [Chainlink's documentation](#) this function is deprecated and should not be used. Moreover, using `latestAnswer()` could lead to return invalid/stale prices.

BVSS:

A0:A/AC:L/AX:L/C:N/I:M/A:N/D:N/Y:N/R:N/S:C (6.2)

Recommendation:

It is recommended to update the `getChainlinkPrice()` function as shown below:

Listing 9: `getChainlinkPrice()` new (Lines 1,8,9,10,11,14,15,16,17)

```
1 uint256 private constant GRACE_PERIOD_TIME = 3600;
2 /// @dev returns price with decimals 18
3 function getChainlinkPrice(AggregatorV2V3Interface feed) internal
4 view returns (uint256) {
5     // Chainlink USD-denominated feeds store answers at 8 decimals
6     uint256 decimalDelta = uint256(18) - feed.decimals();
7     // Ensure that we don't multiply the result by 0
8     if (decimalDelta > 0) {
9         (uint80 roundId, int256 price, uint startedAt, uint
10        updatedAt, uint80 answeredInRound) = feed.latestRoundData();
11         require(answeredInRound >= roundID, "Stale price");
12         require(price > 0, "invalid price");
13         require(block.timestamp <= updatedAt + GRACE_PERIOD_TIME,
14 "Stale price");
15         return uint256(price) * 10 ** decimalDelta;
16     } else {
17         (uint80 roundId, int256 price, uint startedAt, uint
18        updatedAt, uint80 answeredInRound) = feed.latestRoundData();
19         require(answeredInRound >= roundID, "Stale price");
20         require(price > 0, "invalid price");
21         require(block.timestamp <= updatedAt + GRACE_PERIOD_TIME,
22 "Stale price");
23         return uint256(price);
24     }
25 }
```

Remediation Plan:

SOLVED: The [Irrigation Protocol](#) team solved the issue by implementing the recommended solution.

Commit ID : 57d3945089a52425b91cdc8005b62b73d5648c68.

4.5 (HAL-05) USER COULD CANCEL THE REST OF BIDS OF AN AUCTION BY DOING 200 DIFFERENT BIDS - MEDIUM (5.6)

Commit IDs affected:

- 159927ff1e8f8e212cd09894c29c1019e5d417f6

Description:

In the `AuctionUpgradeable` contract, the internal function `_settleAuction()` is called every time an auction is closed:

Listing 10: AuctionUpgradeable.sol (Line 84)

```
310 function _settleAuction(uint256 auctionId) internal {
311     AuctionData memory auction = AuctionStorage.layout().auctions[
312         auctionId];
313     uint256 trancheIndex = auction.trancheIndex;
314     // when there are no bids, all token amount will be transferred
315     // back to seller
316     if (auction.curBidId == 0) {
317         if (trancheIndex == 0) {
318             TransferHelper.safeTransfer(auction.sellToken, auction
319                 .seller, auction.reserve);
320         } else {
321             IERC1155Upgradeable(address(this)).safeTransferFrom(
322                 address(this),
323                 auction.seller,
324                 trancheIndex,
325                 auction.reserve,
326                 Constants.EMPTY
327             );
328         }
329     }
330     AuctionStorage.layout().auctions[auctionId].reserve = 0;
331     AuctionStorage.layout().auctions[auctionId].status =
332         AuctionStatus.Closed;
333     emit AuctionClosed(auction.reserve, auctionId);
334     return;
335 }
```

```
332     uint128 availableAmount = auction.reserve;
333     uint256 settledBidCount = 0;
334     uint256 curBidId = auction.curBidId;
335     do {
336         Bid memory bid = AuctionStorage.layout().bids[auctionId][
337             curBidId];
338         uint128 settledAmount = _settleBid(
339             auction.sellToken,
340             auction.trancheIndex,
341             auction.seller,
342             bid,
343             availableAmount
344         );
345         availableAmount -= settledAmount;
346         AuctionStorage.layout().bids[auctionId][curBidId].bCleared
347             = true;
348         --curBidId;
349         ++settledBidCount;
350     } while (
351         curBidId > 0 &&
352         settledBidCount <= MAX_CHECK_BID_COUNT &&
353         availableAmount >= auction.minBidAmount
354     );
355     if (availableAmount > 0) {
356         if (auction.assetType == AssetType.ERC20) {
357             TransferHelper.safeTransfer(auction.sellToken, auction
358                 .seller, availableAmount);
359         } else {
360             IERC1155Upgradeable(address(this)).safeTransferFrom(
361                 address(this),
362                 auction.seller,
363                 trancheIndex,
364                 availableAmount,
365                 Constants.EMPTY
366             );
367     }
368     }
369     AuctionStorage.layout().auctions[auctionId].reserve = 0;
370     AuctionStorage.layout().auctions[auctionId].status =
371         AuctionStatus.Closed;
372     emit AuctionClosed(availableAmount, auctionId);
373 }
```

This function iterates over all the bids, starting with the last one, transferring the purchased tokens to the bidder until reaching `MAX_CHECK_BID_COUNT` (which is hardcoded to 200). The rest of the bids will be automatically cancelled.

Based on this, the following attack vector could be possible:

1. Alice creates an auction:

Listing 11

```
1 AuctionUpgradeable(irrigationdiamond).createAuction(  
2     uint96(block.timestamp),  
3     uint96(86400 * 7),  
4     address(USDC),  
5     0,  
6     1000e6,  
7     1,  
8     1e18,  
9     1,  
10    2e18,  
11    AuctionType.TimedAuction  
12 );
```

2. Different users place different bids. In total, 150 bids were placed.
3. Alice is not happy with the current bid amounts, so she places 200 different bids on her own auction, 1 second before the auction is completed.
4. The auction is completed. Alice gets her 200 bids, although the rest of the bids of the other users placed before would be automatically canceled.

BVSS:

A0:A/AC:L/AX:L/C:N/I:M/A:N/D:L/Y:N/R:N/S:U (5.6)

Recommendation:

Consider redesigning the logic of the `_settleAuction()` function, so such large number of iterations are not required.

Remediation Plan:

SOLVED: The [Irrigation Protocol team](#) solved the issue by implementing the recommended solution.

Commit ID : [c269697ae4b7997abec4c836d46e3f73f292af03](#).

4.6 (HAL-06) SWAPETHFORWATER CALL CAN BE SANDWICCHED - MEDIUM (5.0)

Commit IDs affected:

- [159927ff1e8f8e212cd09894c29c1019e5d417f6](#)

Description:

In the `WaterTowerUpgradeable` contract, the function `_swapEthForWater()` is used to swap Ether for Beans using the Curve Router:

Listing 12: WaterTowerUpgradeable.sol (Lines 200-202)

```

180 function _swapEthForWater(uint256 amount) internal returns (
181     uint256 waterAmount) {
182     if (WaterTowerStorage.layout().middleAssetForIrrigate ==
183         Constants.BEAN) {
184         /// @dev swap ETH for BEAN using curve router
185         address[9] memory route = [
186             Constants.ETHER,
187             Constants.TRI_CRYPTO_POOL,
188             Constants.USDT,
189             Constants.CURVE_BEAN_METAPOOL,
190             Constants.BEAN,
191             Constants.ZERO,
192             Constants.ZERO,
193             Constants.ZERO,
194             Constants.ZERO];
195         uint256[3][4] memory swapParams = [
196             [uint(2), 0, 3],
197             [uint(3), 0, 2],
198             [uint(0), 0, 0],
199             [uint(0), 0, 0]];
200         uint256 beanAmount = ICurveSwapRouter(Constants.
201             CURVE_ROUTER).exchange_multiple{
202             value: amount
203             }(route, swapParams, amount, 0);
204     }

```

```

204         waterAmount = ISprinklerUpgradeable(address(this)).
205     ↳ getWaterAmount(
206             Constants.BEAN,
207             beanAmount
208         );
209     }

```

This function is calling the `exchange_multiple` function with no prevention against slippage:

Listing 13: CURVE_ROUTER contract (Line 7)

```

1 @external
2 @payable
3 def exchange_multiple(
4     _route: address[9],
5     _swap_params: uint256[3][4],
6     _amount: uint256,
7     _expected: uint256,
8     _pools: address[4]=[ZERO_ADDRESS, ZERO_ADDRESS, ZERO_ADDRESS,
↳ ZERO_ADDRESS],
9     _receiver: address=msg.sender
10 ) -> uint256:

```

Based on this, any `autoIrrigate()` or `irrigate()` call could be sandwiched. A malicious user could be monitoring the mempool for any `autoIrrigate()` or `irrigate()` call and:

1. Front-run, the `autoIrrigate()` or `irrigate()` call to buy beans from the Curve pool. This pushes the price of Bean in the pool up.
2. `autoIrrigate()` or `irrigate()` call is executed and Beans are bought at a higher price. This raises, once again, the Bean price.
3. The malicious user back-runs the `autoIrrigate()` or `irrigate()` call and sells the Beans at a higher price for a profit.

BVSS:

A0:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:M/R:N/S:U (5.0)

Recommendation:

It is recommended to add any type of slippage protection during the `_swapEthForWater()` call. A possible solution would be adding a `minAmountOut` parameter to the `autoIrrigate()` and `irrigate()` functions. This function should be calculated off-chain before any `autoIrrigate()` or `irrigate()` function call.

Remediation Plan:

SOLVED: The Irrigation Protocol team solved the issue by implementing the recommended solution.

Commit ID : f573cdc385d11a8ab3b8929e97678ace5221e4d7.

4.7 (HAL-07) AUTOIRRIGATECALL WILL ALWAYS REVERT WITH OVERFLOW IF ITS CALLED WITH THE FULL REWARDAMOUNT - LOW (2.5)

Commit IDs affected:

- [159927ff1e8f8e212cd09894c29c1019e5d417f6](#)

Description:

In the `WaterTowerUpgradeable` contract, the function `autoIrrigate()` is used to auto-compound the user rewards:

Listing 14: WaterTowerUpgradeable.sol (Line 84)

```
79 function autoIrrigate(address user, uint256 rewardAmount) external
↳   onlySuperAdminRole {
80     if (!WaterTowerStorage.layout().users[user].isAutoIrrigate)
↳ revert NotAutoIrrigate();
81     _irrigate(user, rewardAmount);
82     /// @dev 870391 is the gasLimit for this function
83     uint256 gasFee = 870391 * tx.gasprice;
84     WaterTowerStorage.layout().users[user].pending -= gasFee;
85     emit AutoIrrigate(user, rewardAmount, gasFee);
86 }
```

As this function is called by the protocol admins, the gas fee paid by the admins is subtracted from the user pending balance. Although, the pending balance of the user will always be zero after an `_irrigate()` call of the full reward amount. Hence, if `autoIrrigate()` is called with the full reward amount the call will always revert.

BVSS:

A0:A/AC:L/AX:L/C:N/I:L/A:N/D:N/Y:N/R:N/S:U (2.5)

Recommendation:

It is recommended to update the `autoIrrigate()` implementation, so it is possible to auto-compound the full reward amount. A possible implementation would be storing the `gasFees` to pay per user in a mapping and force the users to pay for them upon withdrawal. Note that the `gasFees` are initially in Ether, they should be converted to `WATER` before they can be subtracted from the `UserInfo.amount`.

Remediation Plan:

RISK ACCEPTED: The Irrigation Protocol team accepted this risk, as they state that the `autoIrrigate()` function will always be called by an admin using `full` reward amount - fee amount.

4.8 (HAL-08) LACK OF A DOUBLE-STEP TRANSFER OWNERSHIP PATTERN - INFORMATIONAL (0.0)

Commit IDs affected:

- [159927ff1e8f8e212cd09894c29c1019e5d417f6](#)

Description:

The `OwnershipFacet` contract allows transferring the ownership of the `IrrigationDiamond` in a single step:

Listing 15: OwnershipFacet.sol (Line 10)

```
8 function transferOwnership(address _newOwner) external override {
9     LibDiamond.enforceIsContractOwner();
10    LibDiamond.setContractOwner(_newOwner);
11 }
```

If the nominated EOA account is not a valid account, it is entirely possible that the owner may accidentally transfer ownership to an uncontrolled account, losing the access to all functions with the `onlySuperAdminRole` modifier.

BVSS:

A0:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:F/S:C (0.0)

Recommendation:

It is recommended to implement a two-step process where the owner nominates an account and the nominated account needs to call an `acceptOwnership()` function for the transfer of the ownership to fully succeed. This ensures the nominated EOA account is a valid and active account. A good code example could be OpenZeppelin's [Ownable2Step contract](#):

Listing 16: Ownable2Step.sol (Lines 52-56)

```
1 // SPDX-License-Identifier: MIT
2 // OpenZeppelin Contracts (last updated v4.8.0) (access/
↳ Ownable2Step.sol)
3
4 pragma solidity ^0.8.0;
5
6 import "./Ownable.sol";
7
8 /**
9  * @dev Contract module which provides access control mechanism,
↳ where
10 * there is an account (an owner) that can be granted exclusive
↳ access to
11 * specific functions.
12 *
13 * By default, the owner account will be the one that deploys the
↳ contract. This
14 * can later be changed with {transferOwnership} and {
↳ acceptOwnership}.
15 *
16 * This module is used through inheritance. It will make available
↳ all functions
17 * from parent (Ownable).
18 */
19 abstract contract Ownable2Step is Ownable {
20     address private _pendingOwner;
21
22     event OwnershipTransferStarted(address indexed previousOwner,
↳ address indexed newOwner);
23
24     /**
25      * @dev Returns the address of the pending owner.
26      */
27     function pendingOwner() public view virtual returns (address)
↳ {
28         return _pendingOwner;
29     }
30
31     /**
32      * @dev Starts the ownership transfer of the contract to a new
↳ account. Replaces the pending transfer if there is one.
33      * Can only be called by the current owner.
34      */
```

```
35     function transferOwnership(address newOwner) public virtual
36     override onlyOwner {
37         _pendingOwner = newOwner;
38         emit OwnershipTransferStarted(owner(), newOwner);
39     }
40     /**
41      * @dev Transfers ownership of the contract to a new account
42      * (`newOwner`) and deletes any pending owner.
43      * Internal function without access restriction.
44      */
45     function _transferOwnership(address newOwner) internal virtual
46     override {
47         delete _pendingOwner;
48         super._transferOwnership(newOwner);
49     }
50     /**
51      * @dev The new owner accepts the ownership transfer.
52      */
53     function acceptOwnership() external {
54         address sender = _msgSender();
55         require(pendingOwner() == sender, "Ownable2Step: caller is
56         not the new owner");
57         _transferOwnership(sender);
58     }
59 }
```

Remediation Plan:

ACKNOWLEDGED: The Irrigation Protocol team acknowledged this finding.

4.9 (HAL-09) CALLING SETMIDDLEASSET WOULD CAUSE THE IRRIGATION BONUS TO ALWAYS BE ZERO - INFORMATIONAL (0.0)

Commit IDs affected:

- [159927ff1e8f8e212cd09894c29c1019e5d417f6](#)

Description:

The contract `WaterTowerUpgradeable` contains the function `setMiddleAsset()`:

Listing 17: WaterTowerUpgradeable.sol (Line 270)

```
269 function setMiddleAsset(address middleAsset) external
↳ onlySuperAdminRole {
270     WaterTowerStorage.layout().middleAssetForIrrigate =
↳ middleAsset;
271 }
```

If this function is ever called to set as `middleAssetForIrrigate` an asset different from the BEAN token, the irrigation bonus would always be zero as currently the `_swapEthForWater()` would only work if `WaterTowerStorage.layout().middleAssetForIrrigate == Constants.BEAN`:

Listing 18: WaterTowerUpgradeable.sol (Line 196)

```
195 function _swapEthForWater(uint256 amount) internal returns (
↳ uint256 waterAmount) {
196     if (WaterTowerStorage.layout().middleAssetForIrrigate ==
↳ Constants.BEAN) {
197         /// @dev swap ETH for BEAN using curve router
198         address[9] memory route = [
199             Constants.ETHER,
200             Constants.TRI_CRYPTO_POOL,
201             Constants.USDT,
202             Constants.CURVE_BEAN_METAPOLY,
203             Constants.BEAN,
```

```
204         Constants.ZERO,
205         Constants.ZERO,
206         Constants.ZERO,
207         Constants.ZERO
208     ];
209     uint256[3][4] memory swapParams = [
210         [uint(2), 0, 3],
211         [uint(3), 0, 2],
212         [uint(0), 0, 0],
213         [uint(0), 0, 0]
214     ];
215     uint256 beanAmount = ICurveSwapRouter(Constants.
↳ CURVE_ROUTER).exchange_multiple{
216         value: amount
217     }(route, swapParams, amount, 0);
218
219     waterAmount = ISprinklerUpgradeable(address(this)).
↳ getWaterAmount(
220         Constants.BEAN,
221         beanAmount
222     );
223 }
224 }
```

BVSS:

A0:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:F/S:C (0.0)

Recommendation:

Consider removing the `setMiddleAsset()` function, as using it would nevertheless require a facet upgrade to work with a new `middleAssetForIrrigate`.

Remediation Plan:

ACKNOWLEDGED: The `Irrigation Protocol` team acknowledged this finding.

RECOMMENDATIONS OVERVIEW

1. Use the `OpenZeppelin's SafeERC20` library to perform all the token transfers instead of the `TransferHelper` library.
2. Consider adding a try/catch code block for the `safeTransfer()` call in the `AuctionUpgradeable._settleBid()` function.
3. Set a minimum `minBidAmount` threshold of for example 1% of the total `sellAmount` in the `AuctionUpgradeable` contract.
4. Update the `getChainlinkPrice()` function as suggested, so it prevents returning any stale price.
5. Consider redesigning the logic of the `AuctionUpgradeable._settleAuction()` function, so such large number of iterations are not required.
6. Add any type of slippage protection during the `_swapEthForWater()` call. A possible solution would be adding a `minAmountOut` parameter to the `autoIrrigate()` and `irrigate()` functions. This function should be calculated off-chain before any `autoIrrigate()` or `irrigate()` function call.
7. Update the `autoIrrigate()` implementation, so it is possible to auto-compound the full reward amount. A possible implementation would be storing the `gasFees` to pay per user in a mapping and force the users to pay for them upon withdrawal. Note that the `gasFees` are initially in Ether, they should be converted to `WATER` before they can be subtracted from the `UserInfo.amount`.
8. Implement a two-step process in the `OwnershipFacet` where the owner nominates an account and the nominated account needs to call an `acceptOwnership()` function for the transfer of the ownership to fully succeed.
9. Consider removing the `setMiddleAsset()` function from the `WaterTowerUpgradeable` as using it would nevertheless require a Facet upgrade to work with a new `middleAssetForIrrigate`.

AUTOMATED TESTING

6.1 STATIC ANALYSIS REPORT

Description:

Halborn used automated testing techniques to enhance the coverage of certain areas of the smart contracts in scope. Among the tools used was Slither, a Solidity static analysis framework. After Halborn verified the smart contracts in the repository and was able to compile them correctly into their ABIs and binary format, Slither was run against the contracts. This tool can statically verify mathematical relationships between Solidity variables to detect invalid or inconsistent usage of the contracts' APIs across the entire code-base.

Slither results:

SprinklerUpgradeable.sol

```
INFODetectors:
Function SprinklerUpgradeable.setTokenMultiplier(address,uint256) (contracts/core/SprinklerUpgradeable.sol#33-35) is a strange setter. Nothing is set in constructor or set in a function without using function parameters
Function LibDiamond.setContractOwner(address) (contracts/libraries/LibDiamond.sol#42-47) is a strange setter. Nothing is set in constructor or set in a function without using function parameters
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/strange_setter.md

INFODetectors:
Function LibDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#217-235) is an unprotected initializer.
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/unprotected_initializer.md

INFODetectors:
LibDiamond.diamondCut(IDiamondCut.FacetCut[],bytes) (contracts/libraries/LibDiamond.sol#85) is a local variable never initialized
LibDiamond.addReplaceRemoveFacetSelectors(bytes,address,IDLamondCut.FacetCutAction,bytes[]) (selectorIndex_scope_0) (contracts/libraries/LibDiamond.sol#139) is a local variable never initialized
LibDiamond.addReplaceRemoveFacetSelectors(bytes,address,IDLamondCut.FacetCutAction,bytes[]) (selectorIndex_scope_3) (contracts/libraries/LibDiamond.sol#156) is a local variable never initialized
LibDiamond.addReplaceRemoveFacetSelectors(bytes,address,IDLamondCut.FacetCutAction,bytes[]) (selectorIndex_scope_10) (contracts/libraries/LibDiamond.sol#110) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized_local_variable

INFODetectors:
Function LibDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#217-235) contains a low level call to a custom address
Function LibDiamond.addReplaceRemoveFacetSelectors(bytes,address,IDLamondCut.FacetCutAction,bytes[]) (selectorIndex_scope_0) (contracts/libraries/LibDiamond.sol#139) contains a low level call to a custom address
Function LibDiamond.addReplaceRemoveFacetSelectors(bytes,address,IDLamondCut.FacetCutAction,bytes[]) (selectorIndex_scope_3) (contracts/libraries/LibDiamond.sol#156) contains a low level call to a custom address
Function LibDiamond.addReplaceRemoveFacetSelectors(bytes,address,IDLamondCut.FacetCutAction,bytes[]) (selectorIndex_scope_10) (contracts/libraries/LibDiamond.sol#110) contains a low level call to a custom address
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/call_forward_to_protected.md

INFODetectors:
IBeanstalkUpgradeable.permitERC20(address,address,address,uint256,uint8,bytes32).owner (contracts/beanstalk/IBeanstalkUpgradeable.sol#30) shadows:
- IBeanstalkUpgradeable.owner() (contracts/beanstalk/IBeanstalkUpgradeable.sol#150) (function)
IBeanstalkUpgradeable.tokenPermitWithes(address).owner (contracts/beanstalk/IBeanstalkUpgradeable.sol#321) shadows:
- IBeanstalkUpgradeable.owner() (contracts/beanstalk/IBeanstalkUpgradeable.sol#150) (function)

IBeanstalkUpgradeable.permitERC20(address,address,address,uint256,uint8,bytes32).owner (contracts/beanstalk/IBeanstalkUpgradeable.sol#352) shadows:
- IBeanstalkUpgradeable.owner() (contracts/beanstalk/IBeanstalkUpgradeable.sol#150) (function)
IBeanstalkUpgradeable.allowancePools(address,address).owner (contracts/beanstalk/IBeanstalkUpgradeable.sol#477) shadows:
- IBeanstalkUpgradeable.owner() (contracts/beanstalk/IBeanstalkUpgradeable.sol#150) (function)
IBeanstalkUpgradeable.claimUnclaimedRewards(uint32,int8).season (contracts/beanstalk/IBeanstalkUpgradeable.sol#721) shadows:
- IBeanstalkUpgradeable.season() (contracts/beanstalk/IBeanstalkUpgradeable.sol#720) (function)
IBeanstalkUpgradeable.depositPermitWithes(address).owner (contracts/beanstalk/IBeanstalkUpgradeable.sol#741) shadows:
- IBeanstalkUpgradeable.owner() (contracts/beanstalk/IBeanstalkUpgradeable.sol#150) (function)
IBeanstalkUpgradeable.getClaimedRewards(uint32,int8).season (contracts/beanstalk/IBeanstalkUpgradeable.sol#754) shadows:
- IBeanstalkUpgradeable.season() (contracts/beanstalk/IBeanstalkUpgradeable.sol#750) (function)
IBeanstalkUpgradeable.getWithdrawableAddress(address,uint32).season (contracts/beanstalk/IBeanstalkUpgradeable.sol#764) shadows:
- IBeanstalkUpgradeable.season() (contracts/beanstalk/IBeanstalkUpgradeable.sol#760) (function)
IBeanstalkUpgradeable.withdrawRewards(uint32,int8,bytes32,bytes32,bytes32).owner (contracts/beanstalk/IBeanstalkUpgradeable.sol#778) shadows:
- IBeanstalkUpgradeable.owner() (contracts/beanstalk/IBeanstalkUpgradeable.sol#150) (function)
IBeanstalkUpgradeable.permitDeposits(address,address,uint32,int75,int76,int8,bytes32,bytes32).owner (contracts/beanstalk/IBeanstalkUpgradeable.sol#789) shadows:
- IBeanstalkUpgradeable.owner() (contracts/beanstalk/IBeanstalkUpgradeable.sol#150) (function)
IBeanstalkUpgradeable.transferDeposit(address,address,int32,int256).hvy (contracts/beanstalk/IBeanstalkUpgradeable.sol#805) shadows:
- IBeanstalkUpgradeable.hvy(address,int32,int256) (contracts/beanstalk/IBeanstalkUpgradeable.sol#802) (function)
IBeanstalkUpgradeable.withdrawDeposit(address,address,int32,int256).hvy (contracts/beanstalk/IBeanstalkUpgradeable.sol#829) shadows:
- IBeanstalkUpgradeable.hvy(address,int32,int256) (contracts/beanstalk/IBeanstalkUpgradeable.sol#826) (function)
IBeanstalkUpgradeable.pledgeRoot(uint32).season (contracts/beanstalk/IBeanstalkUpgradeable.sol#866) shadows:
- IBeanstalkUpgradeable.season() (contracts/beanstalk/IBeanstalkUpgradeable.sol#870) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

INFODetectors:
Reentrancy in SprinklerUpgradeable.depositWater(uint256) (contracts/core/SprinklerUpgradeable.sol#108-115):
- External calls:
  - TransferHelper.safeTransferFrom(address(this),msg.sender,address(this),amount) (contracts/core/SprinklerUpgradeable.sol#113)
    Event emitted after the call(s):
    - DepositWater(amount) (contracts/core/SprinklerUpgradeable.sol#114)

Reentrancy in SprinklerUpgradeable.exchangeETHforWater() (contracts/core/SprinklerUpgradeable.sol#98-106):
- External calls:
  - transferWater(waterAmount) (contracts/core/SprinklerUpgradeable.sol#103)
    Event emitted after the call(s):
    - (success,data) = token.call{gas:encodedIntSelector(0xa09090b0,to,value}) (contracts/libraries/TransferHelper.sol#114)
      Event emitted after the call(s):
      - WaterExchanged(msg.sender,Constants.ETH8,msg.value,waterAmount,false) (contracts/core/SprinklerUpgradeable.sol#105)
    Reentrancy in SprinklerUpgradeable.exchangeTokenToWater(address,uint256) (contracts/core/SprinklerUpgradeable.sol#877-92):
      Event emitted after the call(s):
      - TransferHelper.safeTransferFrom(token,msg.sender,address(this),amount) (contracts/core/SprinklerUpgradeable.sol#88)
        Event emitted after the call(s):
        - TransferHelper.safeTransfer(address(this),msg.sender,amount) (contracts/core/SprinklerUpgradeable.sol#89)
          - (success,data) = token.call{gas:encodedIntSelector(0xa09090b0,to,value}) (contracts/libraries/TransferHelper.sol#114)
            Event emitted after the call(s):
            - WaterExchanged(msg.sender,to,amount,waterAmount) (contracts/core/SprinklerUpgradeable.sol#91)
      Reentrancy in SprinklerUpgradeable.withdrawToken(address,address,uint256) (contracts/core/SprinklerUpgradeable.sol#131-142):
        Event emitted after the call(s):
        - TransferHelper.safeTransferETH(to,amount) (contracts/core/SprinklerUpgradeable.sol#115)
          - TransferHelper.safeTransfer(token,to,amount) (contracts/core/SprinklerUpgradeable.sol#117)
            Event emitted after the call(s):
            - WaterExchanged(msg.sender,to,amount) (contracts/core/SprinklerUpgradeable.sol#111)
      Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

INFODetectors:
Function SprinklerUpgradeable.addAssetToWhitelist(address,uint256) (contracts/core/SprinklerUpgradeable.sol#42-60) has a dubious typecast: IERC20MetadataUpgradeable<address>
Function SprinklerUpgradeable.depositWater(uint256) (contracts/core/SprinklerUpgradeable.sol#103-115) has a dubious typecast: address<bytes>
Function LibDiamond.setContractOwner(address) (contracts/libraries/LibDiamond.sol#42-47) has a dubious typecast: IPricerOracleUpgradeable<address>
Function LibDiamond.addReplaceRemoveFacetSelectors(bytes,address,IDLamondCut.FacetCutAction,bytes[]) (contracts/libraries/LibDiamond.sol#107-215) has a dubious typecast: address<bytes>
Function LibDiamond.addReplaceRemoveFacetSelectors(bytes,address,IDLamondCut.FacetCutAction,bytes[]) (contracts/libraries/LibDiamond.sol#107-215) has a dubious typecast: bytes32<bytes20>
Function LibDiamond.addReplaceRemoveFacetSelectors(bytes,address,IDLamondCut.FacetCutAction,bytes[]) (contracts/libraries/LibDiamond.sol#107-215) has a dubious typecast: bytes32<uint256>
Function LibDiamond.setContractOwner(address) (contracts/libraries/LibDiamond.sol#42-47) has a dubious typecast: bytes32<uint256>
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/dubious_typecast.md
```

AUTOMATED TESTING

WaterTowerUpgradeable.sol

```

INFO:Detectors:
Function WaterTowerUpgradable.setAutoIrrigate(bool) (contracts/core/WaterTowerUpgradable.sol#88-91) is a strange setter. Nothing is set in constructor or set in a function without using function parameters
Function WaterTowerUpgradable.setMintableAddress(address) (contracts/core/WaterTowerUpgradable.sol#94-256) is a strange setter. Nothing is set in constructor or set in a function without using function parameters
Function WaterTowerUpgradable.setIrrigatedAddress(address) (contracts/core/WaterTowerUpgradable.sol#94-256) is a strange setter. Nothing is set in constructor or set in a function without using function parameters
Function LibDiamond.setContractOwner(address) (contracts/libraries/LibDiamond.sol#42-47) is a strange setter. Nothing is set in constructor or set in a function without using function parameters
Reference: https://github.com/pessimistic-lotus/lithium/blob/master/docs/strange.setter.md

INFO:Detectors:
Function WaterTowerUpgradable.setAutoIrrigate(address,uint256) (contracts/core/WaterTowerUpgradable.sol#79-86) tx.gaspire (which is set by user) variable is used. Make sure that it can not exploit the contract logic.
Reference: https://github.com/pessimistic-lotus/lithium/blob/master/docs/tx.gaspire_warning.md

INFO:Detectors:
Function LibDiamond.InitializedDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#217-235) is an unprotected initializer.
Reference: https://github.com/pessimistic-lotus/lithium/blob/master/docs/unprotected_initializer.md

INFO:Detectors:
LibDiamond.addReplaceRemoveFaceSelectors(uint256,bytes32,address,IDIAMonCut.FacetCutAction,bytes4[]).selectorIndex_scope_0 (contracts/libraries/LibDiamond.sol#139) is a local variable never initialized
LibDiamond.diamondCut(IDiamondCut.FacetCut[],address,bytes).facetIndex (contracts/libraries/LibDiamond.sol#95) is a local variable never initialized
LibDiamond.addReplaceRemoveFaceSelectors(uint256,bytes32,address,IDIAMonCut.FacetCutAction,bytes4[]).selectorIndex (contracts/libraries/LibDiamond.sol#118) is a local variable never initialized
LibDiamond.addReplaceRemoveFaceSelectors(uint256,bytes32,address,IDIAMonCut.FacetCutAction,bytes4[]).selectorIndex_scope_3 (contracts/libraries/LibDiamond.sol#156) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/HackerDocumentation#uninitialized-local-variables

INFO:Detectors:
Function LibDiamond.increaseTotalSupply(address,bytes32) (contracts/libraries/LibDiamond.sol#217-235) contains a low level call to a custom address
Function TransferHelper.safeTransferFrom(address,address,uint256) (contracts/libraries/TransferHelper.sol#10-16) contains a low level call to a custom address
Function TransferHelper.safeTransferFrom(address,address,uint256) (contracts/libraries/TransferHelper.sol#18-22) contains a low level call to a custom address
Function TransferHelper.safeTransferETH(address,uint256) (contracts/libraries/TransferHelper.sol#24-27) contains a low level call to a custom address
Reference: https://github.com/pessimistic-lotus/lithium/blob/master/docs/call_forward_to_protected.md

INFO:Detectors:
Reentrancy in WaterTowerUpgradable._irrigate(address,uint256) (contracts/core/WaterTowerUpgradable.sol#122-137):
  External calls:
    - swapDepositAmount = ICurveRouter(water(rewardAmount)) (contracts/core/WaterTowerUpgradable.sol#24)
    - swapDepositAmount = ICurveRouter(water(rewardAmount)).exchange_multiple(value: amount, route: swapParams, amount, 0) (contracts/core/WaterTowerUpgradable.sol#200-202)
  Event emitted after the (call(s)):
    - Deposited(user,amount) (contracts/core/WaterTowerUpgradable.sol#163)
      - deposit(Irrigator,totalDepositWaterAmount) (contracts/core/WaterTowerUpgradable.sol#18)
        - deposit(Irrigator,water(waterAmount.layout().totalDepositForIrrigator,rewardAmount,totalDepositWaterAmount,bonusAmount)) (contracts/core/WaterTowerUpgradable.sol#130-136)
  Reentrancy in WaterTowerUpgradable._irrigate(address,uint256) (contracts/core/WaterTowerUpgradable.sol#79-86):
  External calls:
    - _irrigate(rewardAmount) (contracts/core/WaterTowerUpgradable.sol#81)
    - swapDepositAmount = ICurveRouter(water(rewardAmount)).exchange_multiple(value: amount)(route,swapParams,amount,0) (contracts/core/WaterTowerUpgradable.sol#200-202)
  Event emitted after the (call(s)):
    - AutoIrrigate(user,rewardAmount,gasFee) (contracts/core/WaterTowerUpgradable.sol#85)
  Reentrancy in WaterTowerUpgradable.deposit(uint256,bool) (contracts/core/WaterTowerUpgradable.sol#82-96):
  External calls:
    - swapDepositAmount = ICurveRouter(water(rewardAmount)).exchange_multiple(value: amount)(route,swapParams,amount,0) (contracts/core/WaterTowerUpgradable.sol#200-202)

```

TrancheBondUpgradeable.sol

```

Event emitted after the call(s):
- Deposited(uint amount) (contracts/core/WaterTowerUpgradeable.sol#63)
  - deposit(msg.sender,amount) (contracts/core/WaterTowerUpgradeable.sol#55)
  - SetAutoIrrigate(uint sender,block.timestamp,uintAutoIrrigate) (contracts/core/WaterTowerUpgradeable.sol#90)
Reference: https://github.com/crytic/solidity-detector-Documentation#reentrancy-vulnerabilities-1
INFO-Detectors:
WaterTowerUpgradeable.setPool(uint256,uint256) (contracts/core/WaterTowerUpgradeable.sol#102-271) uses timestamp for comparisons
  Dangerous comparisons:
    - uint256 <= block.timestamp (contracts/core/WaterTowerUpgradeable.sol#26)
    - endline ==> Contracts/core/WaterTowerUpgradeable.sol#265
Reference: https://github.com/crytic/solidity-detector-Documentation#block-timestamp
INFO-Detectors:
Function WaterTowerUpgradeable.deposit(uint256,uint) (contracts/core/WaterTowerUpgradeable.sol#100-209) has a dubious typecast: IERC20Upgradeable<address>
Function WaterTowerUpgradeable.withdraw(uint256) (contracts/core/WaterTowerUpgradeable.sol#159-22) has a dubious typecast: IERC20Upgradeable<address>
Function WaterTowerUpgradeable.sweatForWater(uint256) (contracts/core/WaterTowerUpgradeable.sol#108-209) has a dubious typecast: ISprinklerUpgradeable<address>
Function WaterTowerUpgradeable.getsweatForWater(uint256) (contracts/core/WaterTowerUpgradeable.sol#108-209) has a dubious typecast: uint256<=>CurveveMetapool
Function WaterTowerUpgradeable.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut,FacetCutAction,bytes4[]) (contracts/libraries/LibDiamond.sol#107-215) has a dubious typecast: address<bytes20>
Function LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut,FacetCutAction,bytes4[]) (contracts/libraries/LibDiamond.sol#107-215) has a dubious typecast: bytes32<bytes20>
Function LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut,FacetCutAction,bytes4[]) (contracts/libraries/LibDiamond.sol#107-215) has a dubious typecast: uint16<uint256>
Reference: https://github.com/pessimistic-io/solidther/blob/master/docs/dubious_typecasts.md
INFO-Detectors:
Setter Function WaterTowerUpgradeable.setLdIdleset(address) (contracts/core/WaterTowerUpgradeable.sol#254-256) does not emit an event
Setter Function WaterTowerUpgradeable.setLdIdleset(address,uint256) (contracts/core/WaterTowerUpgradeable.sol#250-256) does not emit an event
Reference: https://github.com/pessimistic-io/solidther/blob/master/docs/revert_setter_md
INFO-Detectors:
WaterTowerStorage.layout() (contracts/core/WaterTowerStorage.sol#51-56) uses assembly
  - INLINE ASM (contracts/core/WaterTowerStorage.sol#51-56) is never used and should be removed
LibDiamond.diamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#31-38) uses assembly
  - INLINE ASM (contracts/libraries/LibDiamond.sol#35-37)
LibDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#217-235) uses assembly
  - INLINE ASM (contracts/libraries/LibDiamond.sol#227-230)
LibDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#237-243) uses assembly
  - INLINE ASM (contracts/libraries/LibDiamond.sol#239-241)
Reference: https://github.com/crytic/solidity-detector-Documentation#assembly_usage
INFO-Detectors:
UtilLib.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut,FacetCutAction,bytes4[]) (contracts/libraries/LibDiamond.sol#107-215) has a high cyclomatic complexity (12).
Reference: https://github.com/crytic/solidity-detector-Documentation#cyclomatic_complexity
INFO-Detectors:
LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut,FacetCutAction,bytes4[]) (contracts/libraries/LibDiamond.sol#107-215) is never used and should be removed
LibDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#217-235) is never used and should be removed
LibDiamond.initializeDiamondCut(IDamondCut,bytes) (contracts/libraries/LibDiamond.sol#227-243) is never used and should be removed
LibDiamond.enforceLastContractCode(address,string) (contracts/libraries/LibDiamond.sol#237-243) is never used and should be removed
LibDiamond.enforceContractOwner() (contracts/libraries/LibDiamond.sol#35-35) is never used and should be removed
LibDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#217-235) is never used and should be removed
LibDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#242-41) is never used and should be removed
TransferHelper.safeApprove(address,address,uint256) (contracts/libraries/TransferHelper.sol#6-10) is never used and should be removed
TransferHelper.safeTransfer(address,address,uint256) (contracts/libraries/TransferHelper.sol#12-16) is never used and should be removed
TransferHelper.safeTransferFrom(address,address,uint256) (contracts/libraries/TransferHelper.sol#24-27) is never used and should be removed
TransferHelper.safeTransferFrom(address,address,uint256) (contracts/libraries/TransferHelper.sol#24-28) is never used and should be removed
Reference: https://github.com/crytic/solidity-detector-Documentation#dead_code
INFO-Detectors:
Pragma version#0.8.17 (contracts/core/WaterTowerStorage.sol#3) allows old versions
Pragma version#0.8.17 (contracts/core/WaterTowerUpgradeable.sol#2) allows old versions
Pragma version#0.8.17 (contracts/core/WaterTowerStorage.sol#1) allows old versions
Pragma version#0.8.17 (contracts/interfaces/ICurveSwapRouter.sol#2) allows old versions
Pragma version#0.8.17 (contracts/interfaces/IDiamondCut.sol#2) allows old versions
Pragma version#0.8.17 (contracts/interfaces/IDiamondUpgradable.sol#2) allows old versions
Pragma version#0.8.17 (contracts/interfaces/ISignInUpgradable.sol#2) allows old versions
Pragma version#0.8.17 (contracts/interfaces/ISignOutUpgradable.sol#2) allows old versions
Pragma version#0.8.17 (contracts/libraries/Constants.sol#2) allows old versions
Pragma version#0.8.17 (contracts/libraries/LibDiamond.sol#2) allows old versions
Pragma version#0.8.17 (contracts/libraries/LibDiamond.sol#1) allows old versions
Pragma version#0.8.17 (contracts/libraries/UtilLib.sol#2) allows old versions
Pragma version#0.8.17 (contracts/libraries/UtilLib.sol#1) allows old versions
Pragma version#0.8.17 (contracts/utils/IrrigationAccessControl.sol#3) allows old versions
solc 0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/solidity-detector-Documentation#incorrect_versions_of_solidity
INFO-Detectors:
Low level call in WaterTowerUpgradeable.claimLdMint() (contracts/core/WaterTowerUpgradeable.sol#65-69)
  - (success) == msg.sender.call.value (claimLdMint) (contracts/core/WaterTowerUpgradeable.sol#67)
Low level call in LibDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#217-235):
  - (success) == token.call.evm.encodeWithSelector(0x999ea0b3,to,value) (contracts/libraries/LibDiamond.sol#217-235)
Low level call in LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut,FaceCutAction,bytes4[]):
  - (success,data) = token.call.evm.encodeWithSelector(0x999ea0b3,to,value) (contracts/libraries/LibDiamond.sol#217-235)
Low level call in TransferHelper.safeTransfer(address,address,uint256) (contracts/libraries/TransferHelper.sol#12-16):
  - (success,data) = token.call.evm.encodeWithSelector(0x999db000,to,value) (contracts/libraries/TransferHelper.sol#12-16)
Low level call in TransferHelper.safeTransferFrom(address,address,uint256) (contracts/libraries/TransferHelper.sol#24-27):
  - (success,data) = token.call.evm.encodeWithSelector(0x230872dd,to,value) (contracts/libraries/TransferHelper.sol#24-27)
Low level call in TransferHelper.safeTransferFrom(address,address,uint256) (contracts/libraries/TransferHelper.sol#24-27):
  - (success) == to.call.value.value((nbytes)(0)) (contracts/libraries/TransferHelper.sol#24-27)
Reference: https://github.com/crytic/solidity-detector-Documentation#low_level_calls
INFO-Detectors:
Function ICurveVetaPool.get_prune_cumulative_last() (contracts/curve/ICurveVetaPool.sol#18) is not in mixedCase
Function ICurveVetaPool.get_dy(int128,int128,uint256,int256[2]) (contracts/curve/ICurveVetaPool.sol#17-12) is not in mixedCase
Function ICurveVetaPool.get_dy_underlying(int128,int128,uint256,int256[2]) (contracts/curve/ICurveVetaPool.sol#17-12) is not in mixedCase
Function ICurveVetaPool.get_dy_underlying(int128,int128,uint256,int256[2]) (contracts/curve/ICurveVetaPool.sol#19-24) is not in mixedCase
Function ICurveVetaPool.get_dy_underlying(int128,int128,uint256,int256[2]) (contracts/curve/ICurveVetaPool.sol#19-24) is not in mixedCase
Parameter ICurveSwapRouter.exchange_multiple(address[],uint256[3][4],uint256[3][4],uint256[3][4],_swap_params) (contracts/curve/ICurveSwapRouter.sol#6-10) is not in mixedCase
Function ICurveSwapRouter.get_exchange_multiple(address[],uint256[3][4],uint256[3][4],uint256[3][4],_swap_params) (contracts/curve/ICurveSwapRouter.sol#12-16) is not in mixedCase
Parameter ICurveSwapRouter.get_exchange_multiple_amount(address[],uint256[3][4],uint256[3][4],uint256[3][4],_swap_params) (contracts/curve/ICurveSwapRouter.sol#12-16) is not in mixedCase
Parameter IDiamond.diamondCut(IDamondCut,FaceCutAction,bytes4[],address,bytes) (contracts/libraries/LibDiamond.sol#69) is not in mixedCase
Parameter LibDiamond.diamondCut(IDamondCut,FaceCutAction,bytes4[],address,bytes) (contracts/libraries/LibDiamond.sol#70) is not in mixedCase
Parameter LibDiamond.diamondCut(IDamondCut,FaceCutAction,bytes4[],address,bytes)_calldata (contracts/libraries/LibDiamond.sol#71) is not in mixedCase
Parameter LibDiamond.diamondCut(IDamondCut,FaceCutAction,bytes4[],address,bytes)_calldata (contracts/libraries/LibDiamond.sol#71) is not in mixedCase
Parameter LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut,FaceCutAction,bytes4[]) (contracts/libraries/LibDiamond.sol#107-215) is not in mixedCase
Parameter LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut,FaceCutAction,bytes4[]) _action (contracts/libraries/LibDiamond.sol#111) is not in mixedCase
Parameter LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut,FaceCutAction,bytes4[]) _action (contracts/libraries/LibDiamond.sol#112) is not in mixedCase
Parameter LibDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#127) is not in mixedCase
Parameter LibDiamond.enforceLastContractCode(address,string) _contract (contracts/libraries/LibDiamond.sol#137) is not in mixedCase
Modifier EP2535Initializable.EP2535Initializable() (contracts/utils/EP2535Initializable.sol#19-19) is not in mixedCase
Modifier EP2535Initializable.EP2535Initializable(uint24) (contracts/utils/EP2535Initializable.sol#21-29) is not in mixedCase
Function IrrigationAccessControl._IrrigationAccessControl_init() (contracts/utils/IrrigationAccessControl.sol#1-16) is not in mixedCase
Function IrrigationAccessControl._IrrigationAccessControl_init_unchained() (contracts/utils/IrrigationAccessControl.sol#18-22) is not in mixedCase
Reference: https://github.com/crytic/solidity-detector-Documentation#solidity_naming_conventions
INFO-Detectors:
Variable LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut,FaceCutAction,bytes4[]) _oldFacet_scope_2 (contracts/libraries/LibDiamond.sol#141) is too similar to LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut,FaceCutAction,bytes4[]) _oldFacet_scope_1 (contracts/libraries/LibDiamond.sol#171)
Variable LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut,FaceCutAction,bytes4[]) _selectorIndex_scope_0 (contracts/libraries/LibDiamond.sol#156) is too similar to LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut,FaceCutAction,bytes4[]) _selector_scope_1 (contracts/libraries/LibDiamond.sol#140)
Variable LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut,FaceCutAction,bytes4[]) _selector_scope_0 (contracts/libraries/LibDiamond.sol#170)
Reference: https://github.com/pessimistic-io/solidther/blob/master/docs/duplicate_names_too_similar
INFO-Detectors:
Function WaterTowerUpgradeable.setAutoRiglate(address,uint256) (contracts/core/WaterTowerUpgradeable.sol#79-86) contains magic number: 878391
Function WaterTowerUpgradeable.getBonusForRiglate(uint256) (contracts/core/WaterTowerUpgradeable.sol#211-216) contains magic number: 3
Function WaterTowerUpgradeable.setPool(uint256,uint256) (contracts/core/WaterTowerUpgradeable.sol#262-271) contains magic number: 259200
Function LibDiamond.diamondCut(IDamondCut,FaceCutAction,bytes4[],address,bytes) (contracts/libraries/LibDiamond.sol#106-108) contains magic numbers: 7, 7, 7
Function LibDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#217-235) contains magic numbers: 7, 224, 3, 3, 7, 7, 5, 3, 7, 8
Function LibDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#217-235) contains magic number: 32
Reference: https://github.com/pessimistic-io/solidther/blob/master/docs/magic_number_md
INFO-Detectors:
  In the function LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut,FaceCutAction,bytes4[]) (contracts/libraries/LibDiamond.sol#107-215) variable LibDiamond.CLEAR_SELECTOR_MASK (contracts/libraries/LibDiamond.sol#60) is read multiple times
Reference: https://github.com/pessimistic-io/solidther/blob/master/docs/multiple_storage_read.md

```

AUTOMATED TESTING

PriceOracleUpgradeable.sol

```

INFO:Dectors:
LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[])(contracts/libraries/LibDiamond.sol#107-215) has a high cyclomatic complexity (12).
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#cyclomatic-complexity

INFO:Dectors:
IrrigationAccessControl._IrrigationAccessControl_initU((contracts/utils/IrrigationAccessControl.sol#13-16)) is never used and should be removed
IrrigationAccessControl._IrrigationAccessControl_initU_unchained((contracts/utils/IrrigationAccessControl.sol#13-16)) is never used and should be removed
LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[])(contracts/libraries/LibDiamond.sol#107-215) is never used and should be removed
LibDiamond.contractOwner() (contracts/libraries/LibDiamond.sol#49-51) is never used and should be removed
LibDiamond.diamondCut(IDiamondCut.FacetCut[],address,bytes) (contracts/libraries/LibDiamond.sol#68-105) is never used and should be removed
LibDiamond.setContractOwner(address) (contracts/libraries/LibDiamond.sol#237-239) is never used and should be removed
LibDiamond.enforceContractOwner() (contracts/libraries/LibDiamond.sol#53-55) is never used and should be removed
LibDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#217-235) is never used and should be removed
LibDiamond.setContractOwner(address) (contracts/libraries/LibDiamond.sol#42-47) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

INFO:Dectors:
Pragma version<=0.8.17 (contracts/beantalk/IBeantalkUpgradable.sol#2) allows old versions
Pragma version>=0.8.17 (contracts/core/WaterCommonStorage.sol#3) allows old versions
Pragma version<=0.8.17 (contracts/core/WaterCommonUpgradeable.sol#2) allows old versions
Pragma version>=0.8.17 (contracts/core/WaterCommonUpgradeable.sol#2) allows old versions
Pragma version<=0.8.17 (contracts/libraries/LibDiamond.sol#2) allows old versions
Pragma version>=0.8.17 (contracts/libraries/LibDiamond.sol#2) allows old versions
Pragma version<=0.8.17 (contracts/utils/IIP2535Initializable.sol#2) allows old versions
Pragma version>=0.8.17 (contracts/utils/IrrigationAccessControl.sol#3) allows old versions
solc -O1 --remap-imports for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

INFO:Dectors:
Low level call in LibDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#217-235);
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

INFO:Dectors:
Function _BeantalkUpgradable.getPenalizedUnderlying(address,uint256,uint256) (contracts/beantalk/IBeantalkUpgradable.sol#74-78) is not in mixedCase
Function WaterCommonUpgradeable.getWaterCommonInitiate(address,address) (contracts/core/WaterCommonUpgradeable.sol#17-21) is not in mixedCase
Parameter WaterCommonUpgradeable.getWaterCommonInitiate(address,address) (contracts/core/WaterCommonUpgradeable.sol#15) is not in mixedCase
Parameter WaterCommonUpgradeable._WaterCommon_initiate(address) (contracts/core/WaterCommonUpgradeable.sol#19-29) is not in mixedCase
Function WaterCommonUpgradeable._WaterCommon_initiate(address) (contracts/core/WaterCommonUpgradeable.sol#19) is not in mixedCase
Parameter LibDiamond.diamondCutContractOwner(address) (contracts/libraries/LibDiamond.sol#42) is not in mixedCase
Parameter LibDiamond.diamondCut(IFaceCut[],address,bytes) (contracts/libraries/LibDiamond.sol#69) is not in mixedCase
Parameter LibDiamond.diamondCut(IFaceCut[],address,bytes,init) (contracts/libraries/LibDiamond.sol#70) is not in mixedCase
Parameter LibDiamond.diamondCut(IFaceCut[],address,bytes,init,bytes) (contracts/libraries/LibDiamond.sol#71) is not in mixedCase
Parameter LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]), selectorCount (contracts/libraries/LibDiamond.sol#100) is not in mixedCase
Parameter LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]), selectorSlot (contracts/libraries/LibDiamond.sol#109) is not in mixedCase
Parameter LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]), selectorIndex (contracts/libraries/LibDiamond.sol#110) is not in mixedCase
Parameter LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]), selectorIndex_scope_0 (contracts/libraries/LibDiamond.sol#111) is not in mixedCase
Parameter LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]), selectorIndex_scope_1 (contracts/libraries/LibDiamond.sol#112) is not in mixedCase
Parameter LibDiamond.initializeDiamondCut(address,bytes), init (contracts/libraries/LibDiamond.sol#217) is not in mixedCase
Parameter LibDiamond.initializeDiamondCut(address,bytes), callData (contracts/libraries/LibDiamond.sol#217) is not in mixedCase
Parameter LibDiamond.enforceContractOwnerContract(address,string), contract (contracts/libraries/LibDiamond.sol#237-239) is not in mixedCase
Parameter LibDiamond.setContractOwnerContract(address,bytes) (contracts/libraries/LibDiamond.sol#237-239) is not in mixedCase
Modifier EP2535Initializable.IEP2535Initializable() (contracts/utils/EP2535Initializable.sol#19-19) is not in mixedCase
Modifier EP2535Initializable.IEP2535Initializable(uint256) (contracts/utils/EP2535Initializable.sol#21-29) is not in mixedCase
Function IrrigationAccessControl._IrrigationAccessControl_init() (contracts/utils/IrrigationAccessControl.sol#13-16) is not in mixedCase
Function LibDiamond._diamondCut(IFaceCut[],address,bytes) (contracts/libraries/LibDiamond.sol#18-22) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#solidity-naming-conventions

INFO:Dectors:
Variable LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]).selector_scope_2 (contracts/libraries/LibDiamond.sol#141) is too similar to LibDiamond.addReplaceRemoveFacetSelectors(uint256
,bytes32,address,IDIamondCut.FacetCutAction,bytes4[])
Variable LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]).selector_index_scope_0 (contracts/libraries/LibDiamond.sol#139) is too similar to LibDiamond.addReplaceRemoveFacetSelectors(ui
,bytes32,address,IDIamondCut.FacetCutAction,bytes4[])
Variable LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]).selector_scope_1 (contracts/libraries/LibDiamond.sol#140) is too similar to LibDiamond.addReplaceRemoveFacetSelectors(uint256
,bytes32,address,IDIamondCut.FacetCutAction,bytes4[])
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

INFO:Dectors:
Function LibDiamond.diamondCut(IDiamondCut.FacetCut[],address,bytes) (contracts/libraries/LibDiamond.sol#68-105) contains magic numbers: 7, 3, 3
Function LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[])(contracts/libraries/LibDiamond.sol#107-215) contains magic numbers: 7, 224, 3, 3, 7, 7, 5, 3, 7, 8
Function LibDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#217-235) contains magic number: 32
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/strange_setter.md

INFO:Dectors:
Function LibDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#217-235) is an unprotected initializer.
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/unprotected_initializer.md

INFO:Dectors:
PriceOracleUpgradeable.gettingEmpyPrice(address) (contracts/oracles/PriceOracleUpgradeable.sol#59-69) performs a multiplication on the result of a division:
    uint256(100 * price / Constants.DIV, uint256(price * Constants.DIV + Constants.MUL));
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#division-with-multiplication

INFO:Dectors:
LibDiamond.diamondCut(IFaceCut[],address,bytes) (contracts/libraries/LibDiamond.sol#18-22) is a local variable never initialized
LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]).selectorIndex_scope_0 (contracts/libraries/LibDiamond.sol#239) is a local variable never initialized
LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]).selectorIndex_scope_1 (contracts/libraries/LibDiamond.sol#156) is a local variable never initialized
LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]).selectorIndex (contracts/libraries/LibDiamond.sol#118) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialised-local-variables

INFO:Dectors:
Function LibDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#217-235) contains a low level call to a custom address
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/forward_to_protected_md

INFO:Dectors:
Function LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[])(contracts/libraries/LibDiamond.sol#107-215) has a dubious typecast: address=>bytes20
Function LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[])(contracts/libraries/LibDiamond.sol#107-215) has a dubious typecast: bytes32=>bytes20
Function LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[])(contracts/libraries/LibDiamond.sol#107-215) has a dubious typecast: uint16=>uint256
Function LibDiamond.slitherConstructorConstantVariables() (contracts/libraries/LibDiamond.sol#181-244) has a dubious typecast: bytes32=>int256
Function PriceOracleUpgradeable.setOracleAddress(address,bytes) (contracts/oracles/PriceOracleUpgradeable.sol#79-99) has a dubious typecast: AggregatorV2VInterface=>uint256
Function UniswapV3Pool.getPool(bytes32,address,int256) (contracts/oracles/UniswapV3Pool.sol#3-23) has a dubious typecast: int256=>int256
Function UniswapV3Pool.getPool(bytes32,address,int256) (contracts/oracles/UniswapV3Pool.sol#32-47) has a dubious typecast: uint256=>int256
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/dubious_typecast.md

INFO:Dectors:
LibDiamond.setContractOwner(address) (contracts/libraries/LibDiamond.sol#33-38) uses assembly
    - INLINE ASM (contract/libraries/LibDiamond.sol#35-37)
LibDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#217-235) uses assembly
    - INLINE ASM (contract/libraries/LibDiamond.sol#227-230)
LibDiamond.enforceContractOwnerContract(address,string) (contracts/libraries/LibDiamond.sol#237-243) uses assembly
    - INLINE ASM (contract/libraries/LibDiamond.sol#237-241)
PriceOracleStorage.layout() (contracts/oracles/PriceOracleStorage.sol#34-39) uses assembly
    - INLINE ASM (contract/oracles/PriceOracleStorage.sol#36-38)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-only-use

INFO:Dectors:
LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[])(contracts/libraries/LibDiamond.sol#107-215) has a high cyclomatic complexity (12).
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#cyclomatic-complexity

INFO:Dectors:
IrrigationAccessControl._IrrigationAccessControl_init() (contracts/utils/IrrigationAccessControl.sol#13-16) is never used and should be removed
IrrigationAccessControl._IrrigationAccessControl_init_unchained() (contracts/utils/IrrigationAccessControl.sol#13-16) is never used and should be removed
LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[])(contracts/libraries/LibDiamond.sol#107-215) is never used and should be removed
LibDiamond.contractOwner() (contracts/libraries/LibDiamond.sol#49-51) is never used and should be removed
LibDiamond.diamondCut(IDiamondCut.FacetCut[],address,bytes) (contracts/libraries/LibDiamond.sol#68-105) is never used and should be removed
LibDiamond.setContractOwnerContract(address,bytes) (contracts/libraries/LibDiamond.sol#237-239) is never used and should be removed
LibDiamond.enforceContractOwner() (contracts/libraries/LibDiamond.sol#53-55) is never used and should be removed
LibDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#217-235) is never used and should be removed
LibDiamond.setContractOwnerContract(address) (contracts/libraries/LibDiamond.sol#42-47) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

INFO:Dectors:
Pragma version<=0.8.17 (contracts/interfaces/ICustomTrade.sol#2) allows old versions
Pragma version>=0.8.17 (contracts/interfaces/IDiamondCut.sol#2) allows old versions
Pragma version<=0.8.17 (contracts/libraries/Constants.sol#2) allows old versions
Pragma version>=0.8.17 (contracts/libraries/Constants.sol#2) allows old versions
Pragma version<=0.8.17 (contracts/libraries/AggregatorV2VInterface.sol#2) allows old versions
Pragma version>=0.8.17 (contracts/oracles/PriceOracleStorage.sol#3) allows old versions
Pragma version<=0.8.17 (contracts/oracles/PriceOracleUpgradeable.sol#2) allows old versions
Pragma version>=0.8.17 (contracts/oracles/PriceOracle.sol#2) allows old versions
Pragma version<=0.8.17 (contracts/utils/EP2535Initializable.sol#2) allows old versions
Pragma version>=0.8.17 (contracts/utils/IrrigationAccessControl.sol#3) allows old versions
solc -O1 --remap-imports for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

INFO:Dectors:
Low level call in LibDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#217-235):
    - success,error = _init_delegatecall_(calldata) (contracts/libraries/LibDiamond.sol#222)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

```


IrrigationDiamond.sol

Diamond.sol

```

INFO[Detectors]:
LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]).selectorIndex (contracts/libraries/LibDiamond.sol#118) is a local variable never initialized
LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]).selectorIndex_scope_0 (contracts/libraries/LibDiamond.sol#139) is a local variable never initialized
LibDiamond.diamondCut(IDiamondCut.FacetCut[],address,_facetIndex (contracts/libraries/LibDiamond.sol#85) is a local variable never initialized
LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]).selectorIndex_scope_3 (contracts/libraries/LibDiamond.sol#156) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO[Detectors]:
Function LibDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#217-235) contains a low level call to a custom address
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/call_forward_to_protected.md
INFO[Detectors]:
Function LibDiamond.initializeDiamondCut(address,bytes20)
Function LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]) (contracts/libraries/LibDiamond.sol#107-215) has a dubious typecast: address->bytes20
Function LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]) (contracts/libraries/LibDiamond.sol#107-215) has a dubious typecast: bytes32->bytes20
Function LibDiamond.diamondCut(IDiamondCut.FacetCut[],address,_facetIndex (contracts/libraries/LibDiamond.sol#85) (contracts/libraries/LibDiamond.sol#107-215) has a dubious typecast: uint32<=>uint256
Function LibDiamond.slitherContractConstructorConstantVariables() (contracts/libraries/LibDiamond.sol#113-244) has a dubious typecast: bytes32<=>uint256
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/dubious_typecast.md
INFO[Detectors]:
LibDiamond.enforceContractCode(address,string) (contracts/libraries/LibDiamond.sol#237-243) uses assembly
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly
INFO[Detectors]:
LibDiamond.enforceContractCode(address) (contracts/libraries/LibDiamond.sol#33-38) uses assembly
- INLINE ASM (contracts/libraries/LibDiamond.sol#35-37)
LibDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#217-235) uses assembly
- INLINE ASM (contracts/libraries/LibDiamond.sol#227-230)
LibDiamond.enforceContractCode(address,string) (contracts/libraries/LibDiamond.sol#237-243) uses assembly
- INLINE ASM (contracts/libraries/LibDiamond.sol#237-243)
Diamond.fallback() (contracts/utils/Diamond.sol#32-59) uses assembly
- INLINE ASM (contracts/utils/Diamond.sol#38)
INFO[Detectors]:
LibDiamond.enforceContractOwner() (contracts/libraries/LibDiamond.sol#33-38) uses assembly
- INLINE ASM (contracts/libraries/LibDiamond.sol#43-45)
Reference: https://github.com/crytic/slitherin/blob/master/docs/contract_owner_usage.md
INFO[Detectors]:
LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]) (contracts/libraries/LibDiamond.sol#107-215) has a high cyclomatic complexity (12).
Reference: https://github.com/crytic/slitherin/blob/master/docs/cyclomatic_complexity.md
INFO[Detectors]:
LibDiamond.enforceContractOwner() (contracts/libraries/LibDiamond.sol#49-51) is never used and should be removed
LibDiamond.enforceContractOwner() (contracts/libraries/LibDiamond.sol#53-55) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead_code
INFO[Detectors]:
Pragma version 0.8.17 (contracts/levigationDiamond.sol#2) allows old versions
Pragma version 0.8.17 (contracts/interfaces/DiamondCut.sol#2) allows old versions
Pragma version 0.8.17 (contracts/interfaces/DiamondCut.sol#2) allows old versions
Pragma version 0.8.17 (contracts/libraries/LibDiamond.sol#2) allows old versions
pragma 0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect_versions_of_solidity
INFO[Detectors]:
Low level call in LibDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#217-235):
  - (success,error) = _init.delegatecall(_calldata) (contracts/libraries/LibDiamond.sol#222)
Reference: https://github.com/crytic/slitherin/blob/master/docs/low_level_calls
INFO[Detectors]:
Function IrrigationDiamond.finalizeInitialization() (contracts/IrrigationDiamond.sol#5-58) is not in mixedCase
Function IrrigationDiamond.finalizeInitialization(uint8) (contracts/IrrigationDiamond.sol#40-42) is not in mixedCase
Parameter LibDiamond.setContractOwner(address), _newOwner (contracts/libraries/LibDiamond.sol#42) is not in mixedCase
Parameter LibDiamond.diamondCut(IDiamondCut.FacetCut[],address,bytes)_diamondCut (contracts/libraries/LibDiamond.sol#109) is not in mixedCase
Parameter LibDiamond.diamondCut(IDiamondCut.FacetCut[],address,bytes)_facetIndex (contracts/libraries/LibDiamond.sol#109) is not in mixedCase
Parameter LibDiamond.diamondCut(IDiamondCut.FacetCut[],address,bytes)_facetIndex_scope_0 (contracts/libraries/LibDiamond.sol#109) is not in mixedCase
Parameter LibDiamond.diamondCut(IDiamondCut.FacetCut[],address,bytes)_facetIndex_scope_1 (contracts/libraries/LibDiamond.sol#109) is not in mixedCase
Parameter LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[])_selectorCount (contracts/libraries/LibDiamond.sol#108) is not in mixedCase
Parameter LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[])_selectorSlot (contracts/libraries/LibDiamond.sol#109) is not in mixedCase
Parameter LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[])_newFacetAddress (contracts/libraries/LibDiamond.sol#110) is not in mixedCase
Parameter LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[])_action (contracts/libraries/LibDiamond.sol#111) is not in mixedCase
Parameter LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[])_calldata (contracts/libraries/LibDiamond.sol#112) is not in mixedCase
Parameter LibDiamond.initializeDiamondCut(address,bytes)_init (contracts/libraries/LibDiamond.sol#121) is not in mixedCase
Parameter LibDiamond.initializeDiamondCut(address,bytes)_calldata (contracts/libraries/LibDiamond.sol#121) is not in mixedCase
Parameter LibDiamond.enforceContractCode(address,string)_contract (contracts/libraries/LibDiamond.sol#123) is not in mixedCase
Parameter LibDiamond.enforceContractCode(address,string)_selectorIndex (contracts/libraries/LibDiamond.sol#123) is not in mixedCase
Parameter LibDiamond.enforceContractCode(address,string)_selectorIndex_scope_0 (contracts/libraries/LibDiamond.sol#123) is not in mixedCase
Parameter LibDiamond.enforceContractCode(address,string)_selectorIndex_scope_1 (contracts/libraries/LibDiamond.sol#123) is not in mixedCase
Modifier IP2535Initializable(IP2535Initializer) (contracts/utils/IP2535Initializable.sol#11-19) is not in mixedCase
Modifier IP2535Initializer(IP2535Initializer) (contracts/utils/IP2535Initializable.sol#11-19) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance_to_solidity_naming_conventions
INFO[Detectors]:
Variable LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[])_facet_scope_2 (contracts/libraries/LibDiamond.sol#141) is too similar to LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[])_facet_scope_0
Variable LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[])_facet_scope_3 (contracts/libraries/LibDiamond.sol#171) is too similar to LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[])_facet_scope_0
Variable LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[])_selectorIndex_scope_3 (contracts/libraries/LibDiamond.sol#180) is too similar to LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[])_selector_index_scope_0
Variable LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[])_selector_scope_1 (contracts/libraries/LibDiamond.sol#140) is too similar to LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[])_selector_scope_0
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable_names_too_similar
INFO[Detectors]:
Function LibDiamond.diamondCut(IDiamondCut.FacetCut[],address,bytes) (contracts/libraries/LibDiamond.sol#86-105) contains magic numbers: 7, 3, 7, 3
Function LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]) (contracts/libraries/LibDiamond.sol#107-215) contains magic numbers: 7, 224, 3, 3, 7, 7, 5, 3, 7, 8
Function LibDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#217-235) contains magic number: 32
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number.md
INFO[Detectors]:
In a function LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]) (contracts/libraries/LibDiamond.sol#107-215) variable LibDiamond.CLEAR_SELECTOR_MASK (contracts/libraries/LibDiamond.sol#68) is read multiple times
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/multiple_storage_read.md

```

```

INFO[Detectors]:
Function LibDiamond.setContractOwner(address) (contracts/libraries/LibDiamond.sol#42-47) is a strange setter. Nothing is set in constructor or set in a function without using function parameters
Function LibDiamond.constructor(address,address) (contracts/utils/Diamond.sol#15-28) is a strange setter. Nothing is set in constructor or set in a function without using function parameters
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/strange_setter.md
INFO[Detectors]:
Function LibDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#217-235) is an unprotected initializer.
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/unprotected_initializer.md
INFO[Detectors]:
LibDiamond.diamondCut(IDiamondCut.FacetCut[],address,bytes)_facetIndex (contracts/libraries/LibDiamond.sol#85) is a local variable never initialized
LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[])_selectorIndex_scope_3 (contracts/libraries/LibDiamond.sol#156) is a local variable never initialized
LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[])_selectorIndex (contracts/libraries/LibDiamond.sol#118) is a local variable never initialized
LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[])_selectorIndex_scope_0 (contracts/libraries/LibDiamond.sol#139) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized_local_variables
INFO[Detectors]:
Function LibDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#217-235) contains a low level call to a custom address
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/call_forward_to_protected.md
INFO[Detectors]:
Function LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[])_facet_scope_2 (contracts/libraries/LibDiamond.sol#141) has a dubious typecast: address->bytes20
Function LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[])_facet_scope_0 (contracts/libraries/LibDiamond.sol#107-215) has a dubious typecast: bytes32->bytes20
Function LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[])_facet_scope_1 (contracts/libraries/LibDiamond.sol#107-215) has a dubious typecast: uint32<=>uint256
Function LibDiamond.fallback() (contracts/utils/Diamond.sol#32-59) has a dubious typecast: bytes32<=>uint256
Reference: https://github.com/crytic/slitherin/blob/master/docs/dubious_typecast.md
INFO[Detectors]:
LibDiamond.diamondStorage() (contracts/libraries/LibDiamond.sol#33-38) uses assembly
- INLINE ASM (contracts/libraries/LibDiamond.sol#35-37)
LibDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#217-235) uses assembly
- INLINE ASM (contracts/libraries/LibDiamond.sol#227-230)
LibDiamond.enforceContractCode(address,string) (contracts/libraries/LibDiamond.sol#237-243) uses assembly
- INLINE ASM (contracts/libraries/LibDiamond.sol#237-241)
Diamond.fallback() (contracts/utils/Diamond.sol#32-59) uses assembly
- INLINE ASM (contracts/utils/Diamond.sol#38)
- INLINE ASM (contracts/utils/Diamond.sol#43-45)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly
INFO[Detectors]:
LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]) (contracts/libraries/LibDiamond.sol#107-215) has a high cyclomatic complexity (12).
Reference: https://github.com/crytic/slitherin/blob/master/docs/cyclomatic_complexity.md
INFO[Detectors]:
LibDiamond.enforceContractOwner() (contracts/libraries/LibDiamond.sol#49-51) is never used and should be removed
LibDiamond.enforceContractOwner() (contracts/libraries/LibDiamond.sol#53-55) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead_code
INFO[Detectors]:
Pragma version 0.8.17 (contracts/interfaces/DiamondCut.sol#2) allows old versions
Pragma version 0.8.17 (contracts/libraries/LibDiamond.sol#2) allows old versions
Pragma version 0.8.17 (contracts/libraries/LibDiamond.sol#2) allows old versions
pragma 0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect_versions_of_solidity
INFO[Detectors]:
Low level call in LibDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#217-235):
  - (success,error) = _init.delegatecall(_calldata) (contracts/libraries/LibDiamond.sol#182)
Reference: https://github.com/crytic/slitherin/blob/master/docs/low_level_calls
INFO[Detectors]:
Parameter LibDiamond.setContractOwner(address) (contracts/libraries/LibDiamond.sol#42) is not in mixedCase
Parameter LibDiamond.diamondCut(IDiamondCut.FacetCut[],address,bytes)_diamondCut (contracts/libraries/LibDiamond.sol#69) is not in mixedCase
Parameter LibDiamond.diamondCut(IDiamondCut.FacetCut[],address,bytes)_facetIndex (contracts/libraries/LibDiamond.sol#70) is not in mixedCase
Parameter LibDiamond.diamondCut(IDiamondCut.FacetCut[],address,bytes)_calldata (contracts/libraries/LibDiamond.sol#71) is not in mixedCase

```

```

Parameter LIBDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]),_selectorCount (contracts/libraries/LibDiamond.sol#100) is not in mixedCase
Parameter LIBDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]),_selectorSlot (contracts/libraries/LibDiamond.sol#109) is not in mixedCase
Parameter LIBDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]),_newFacetAddress (contracts/libraries/LibDiamond.sol#110) is not in mixedCase
Parameter LIBDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]),_action (contracts/libraries/LibDiamond.sol#111) is not in mixedCase
Parameter LIBDiamond.initializeDiamondCut(address,bytes),_libraryName (contracts/libraries/LibDiamond.sol#121) is not in mixedCase
Parameter LIBDiamond.initializeDiamondCut(address,bytes),_calldata (contracts/libraries/LibDiamond.sol#127) is not in mixedCase
Parameter LIBDiamond.enforceContractCode(address,string),_contract (contracts/libraries/LibDiamond.sol#237) is not in mixedCase
Parameter LIBDiamond.enforceContractCode(address,string),_errorMessage (contracts/libraries/LibDiamond.sol#237) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFODetectors:
Variables IDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]),oldfacet_scope_2 (contracts/libraries/LibDiamond.sol#141) is too similar to LIBDiamond.addReplaceRemoveFacetSelectors(uint256
,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]),oldfacet_scope_1 (contracts/libraries/LibDiamond.sol#141)
Variables IDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]),_selectorIndex_scope_0 (contracts/libraries/LibDiamond.sol#139) is too similar to LIBDiamond.addReplaceRemoveFacetSelectors(ui
nt256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[])
Variables IDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]),_selector_scope_1 (contracts/libraries/LibDiamond.sol#140) is too similar to LIBDiamond.addReplaceRemoveFacetSelectors(ui
nt256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[])
Variables IDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]),_selector_scope_0 (contracts/libraries/LibDiamond.sol#170)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFODetectors:
Function IDiamond.diamondCut(IDiamondCut.FacetCut[],address,bytes) (contracts/libraries/LibDiamond.sol#68-105) contains magic numbers: 7, 3, 7, 3
Function IDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[])(contracts/libraries/LibDiamond.sol#107-215) contains magic numbers: 7, 224, 3, 3, 7, 7, 5, 3, 7, 8
Function IDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#217-235) contains magic number: 32
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/multiple_storage_read.md
INFODetectors:
In a function LIBDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[])(contracts/libraries/LibDiamond.sol#107-215) variable LIBDiamond.CLEAR_SELECTOR_MASK (contracts/libraries/LibDiamond.sol
#60) is read multiple times
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/multiple_storage_read.md

```

DiamondCutFacet.sol

```

INFODetectors:
Function LIBDiamond.setContractOwner(address) (contracts/libraries/LibDiamond.sol#42-47) is a strange setter. Nothing is set in constructor or set in a function without using function parameters
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/strange_setter.md
INFODetectors:
Function LIBDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#217-235) is an unprotected initializer.
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/unprotected_initializer.md
INFODetectors:
LIBDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]),_selectorIndex_scope_0 (contracts/libraries/LibDiamond.sol#119) is a local variable never initialized
LIBDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]),_action (contracts/libraries/LibDiamond.sol#119)
LIBDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]),_newFacetAddress (contracts/libraries/LibDiamond.sol#119)
LIBDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]),_libraryName (contracts/libraries/LibDiamond.sol#156)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFODetectors:
Function IDiamond.diamondCut(IDiamondCut.FacetCut[],address,bytes) (contracts/libraries/LibDiamond.sol#68-105) contains magic numbers: 7, 3, 7, 3
Function IDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[])(contracts/libraries/LibDiamond.sol#107-215) has a dubious typecast: address=bytes20
Function IDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#217-235) has a dubious typecast: bytes32=bytes20
Function IDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[])(contracts/libraries/LibDiamond.sol#107-215) has a dubious typecast: uint256=uint256
Function IDiamond.slitherContractForConstantVariable() (contracts/libraries/LibDiamond.sol#123-244) has a dubious typecast: bytes32<=uint256
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/dubious_typecasts.md
INFODetectors:
LIBDiamond.diamondStorage() (contracts/libraries/LibDiamond.sol#31-38) uses assembly
- IN-LINE ASM (contracts/libraries/LibDiamond.sol#35-37)
LIBDiamond.diamondStorage() (contracts/libraries/LibDiamond.sol#217-235) contains assembly
- IN-LINE ASM (contracts/libraries/LibDiamond.sol#227-230)
LIBDiamond.enforceContractCode(address,string) (contracts/libraries/LibDiamond.sol#237-243) uses assembly
- IN-LINE ASM (contracts/libraries/LibDiamond.sol#239-241)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-use
INFODetectors:
LIBDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[])(contracts/libraries/LibDiamond.sol#107-215) has a high cyclomatic complexity (12).
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#cyclomatic-complexity
INFODetectors:
LIBDiamond.setContractOwner() (contracts/libraries/LibDiamond.sol#40-51) is never used and should be removed
LIBDiamond.diamondCut(IDiamondCut.FacetCut[],address,bytes) (contracts/libraries/LibDiamond.sol#68-105) is never used and should be removed
LIBDiamond.setContractOwner(address) (contracts/libraries/LibDiamond.sol#42-47) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead_code
INFODetectors:
Pragma version="0.8.17" (contracts/interfaces/LibDiamond.sol#0) allows old versions
Pragma version="0.8.17" (contracts/libraries/LibDiamond.sol#2) allows old versions
Pragma version="0.8.17" (contracts/utils/IDiamondCut.sol#2) allows old versions
Solidity 0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFODetectors:
Low level call in LIBDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#217-235):
    (success,error) = init.delegatecall(calldata) (contracts/libraries/LibDiamond.sol#222)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFODetectors:
Parameter LIBDiamond.setContractOwner(address),_newOwner (contracts/libraries/LibDiamond.sol#42) is not in mixedCase
Parameter LIBDiamond.diamondCut(IDiamondCut.FacetCut[],address,bytes),_diamondCut (contracts/libraries/LibDiamond.sol#69) is not in mixedCase
Parameter LIBDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]),_action (contracts/libraries/LibDiamond.sol#119) is not in mixedCase
Parameter LIBDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]),_libraryName (contracts/libraries/LibDiamond.sol#156) is not in mixedCase
Parameter LIBDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]),_selectorIndex_scope_0 (contracts/libraries/LibDiamond.sol#141) is not in mixedCase
Parameter LIBDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]),_selectorSlot (contracts/libraries/LibDiamond.sol#140) is not in mixedCase
Parameter LIBDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]),_newFacetAddress (contracts/libraries/LibDiamond.sol#141) is not in mixedCase
Parameter LIBDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]),_selector_index_scope_0 (contracts/libraries/LibDiamond.sol#140) is not in mixedCase
Parameter LIBDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]),_selector_scope_0 (contracts/libraries/LibDiamond.sol#170)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFODetectors:
Function IDiamond.diamondCut(IDiamondCut.FacetCut[],address,bytes) (contracts/libraries/LibDiamond.sol#68-105) contains magic numbers: 7, 3, 7, 3
Function IDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[])(contracts/libraries/LibDiamond.sol#107-215) contains magic numbers: 7, 224, 3, 3, 7, 7, 5, 3, 7, 8
Function IDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#217-235) contains magic number: 32
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number.md
INFODetectors:
In a function LIBDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[])(contracts/libraries/LibDiamond.sol#107-215) variable LIBDiamond.CLEAR_SELECTOR_MASK (contracts/libraries/LibDiamond.sol
#60) is read multiple times
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/multiple_storage_read.md

```

DiamondLoupeFacet.sol

```

INFODetectors:
Function LIBDiamond.setContractOwner(address) (contracts/libraries/LibDiamond.sol#42-47) is a strange setter. Nothing is set in constructor or set in a function without using function parameters
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/strange_setter.md
INFODetectors:
Function LIBDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#217-235) is an unprotected initializer.
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/unprotected_initializer.md
INFODetectors:
LIBDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]),_selectorIndex_scope_3 (contracts/libraries/LibDiamond.sol#156) is a local variable never initialized
LIBDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]),_selectorIndex (contracts/libraries/LibDiamond.sol#118) is a local variable never initialized
LIBDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]),_libraryName (contracts/libraries/LibDiamond.sol#156) is a local variable never initialized
LIBDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]),_selectorIndex_scope_0 (contracts/libraries/LibDiamond.sol#139) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFODetectors:
Function LIBDiamond.diamondCut(IDiamondCut.FacetCut[],address,bytes) (contracts/libraries/LibDiamond.sol#68-105) contains magic numbers: 7, 3, 7, 3
Function LIBDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[])(contracts/libraries/LibDiamond.sol#107-215) contains magic numbers: 7, 224, 3, 3, 7, 7, 5, 3, 7, 8
Function LIBDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#217-235) contains magic number: 32
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number.md
INFODetectors:
In a function LIBDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[])(contracts/libraries/LibDiamond.sol#107-215) variable LIBDiamond.CLEAR_SELECTOR_MASK (contracts/libraries/LibDiamond.sol
#60) is read multiple times
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/multiple_storage_read.md

```

EIP2535Initializable.sol

```

INFODetectors:
Function LibDiamond.slitherConstructorConstantVariables() (contracts/libraries/LibDiamond.sol#13-244) has a dubious typecast: bytes32<=uint256
Function DiamondLoupeFacet.Facets() (contracts/utils/DiamondLoupeFacet.sol#26-76) has a dubious typecast: address<=bytes20
Function DiamondLoupeFacet.FaceFunctionSelectors(address) (contracts/utils/DiamondLoupeFacet.sol#81-107) has a dubious typecast: address<=bytes20
Function DiamondLoupeFacet.FaceAddresses() (contracts/utils/DiamondLoupeFacet.sol#111-145) has a dubious typecast: address<=bytes20
Function DiamondLoupeFacet.FacetStorage() (contracts/utils/DiamondLoupeFacet.sol#151-154) has a dubious typecast: address<=bytes20
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/dubious_typecast.md
INFODetectors:
LibDiamond.diamondStorage() (contracts/libraries/LibDiamond.sol#35-38) uses assembly
    - INLINE ASM (contracts/libraries/LibDiamond.sol#35-37)
LibDiamond.initializeOwner(address) (contracts/libraries/LibDiamond.sol#217-235) uses assembly
    - INLINE ASM (contracts/libraries/LibDiamond.sol#227-230)
LibDiamond.enforceIsContractCode(address, string) (contracts/libraries/LibDiamond.sol#237-243) uses assembly
    - INLINE ASM (contracts/libraries/LibDiamond.sol#239-241)
DiamondLoupeFacet.addReplaceRemoveFacetSelectors(uint256, bytes32, address, IDiamondCut.FacetCutAction, bytes4[]) (contracts/libraries/LibDiamond.sol#86-70)
    - INLINE ASM (contracts/utils/DiamondLoupeFacet.sol#73-75)
DiamondLoupeFacet.FaceFunctionSelectors(address) (contracts/utils/DiamondLoupeFacet.sol#81-107) uses assembly
    - INLINE ASM (contracts/utils/DiamondLoupeFacet.sol#84-106)
DiamondLoupeFacet.FaceAddresses() (contracts/utils/DiamondLoupeFacet.sol#111-145) uses assembly
    - INLINE ASM (contracts/utils/DiamondLoupeFacet.sol#142-144)
Reference: https://github.com/crytic/slither/wikid/Better-DocmentationAssemblyUsage
INFODetectors:
Util.replaceRemoveFacetSelectors(uint256, bytes32, address, IDiamondCut.FacetCutAction, bytes4[]) (contracts/libraries/LibDiamond.sol#107-215) has a high cyclomatic complexity (12).
Reference: https://github.com/crytic/slither/wikid/Better-DocmentationAsymCyclomaticComplexity
INFODetectors:
LibDiamond.addReplaceRemoveFacetSelectors(uint256, bytes32, address, IDiamondCut.FacetCutAction, bytes4[]) (contracts/libraries/LibDiamond.sol#107-215) is never used and should be removed
LibDiamond.diamondCut(FacetCut[] calldata, address, bytes32) (contracts/libraries/LibDiamond.sol#53-55) is never used and should be removed
LibDiamond.enforceIsContractCode(address, string) (contracts/libraries/LibDiamond.sol#37-243) is never used and should be removed
LibDiamond.enforceIsContractOwner() (contracts/libraries/LibDiamond.sol#53-55) is never used and should be removed
LibDiamond.setContractOwner(address) (contracts/libraries/LibDiamond.sol#107-215) is never used and should be removed
LibDiamond.setContractOwner(address) (contracts/libraries/LibDiamond.sol#142-147) is never used and should be removed
Reference: https://github.com/crytic/slither/wikid/Better-DocmentationDeadCode
INFODetectors:
Pragma version<=0.8.17 (contracts/interfaces/IDiamondCut.sol#2) allows old versions
Pragma version>0.8.17 (contracts/interfaces/IDiamondCut.sol#2) allows old versions
Pragma version<=0.8.17 (contracts/libraries/LibDiamond.sol#2) allows old versions
Pragma version<=0.8.17 (contracts/utils/DiamondLoupeFacet.sol#2) allows old versions
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wikid/Better-DocmentationIncorrectVersionsOfSolidity
INFODetectors:
Low level call in LibDiamond.initializeOwner(address, bytes) (contracts/libraries/LibDiamond.sol#217-235)
    - (success, error) = init.delegatecall(calldata) (contracts/libraries/LibDiamond.sol#222)
Reference: https://github.com/crytic/slither/wikid/Better-DocmentationLowLevelCalls
INFODetectors:
Parameter LibDiamond.setContractOwner(address), newOwner (contracts/libraries/LibDiamond.sol#42) is not in mixedCase
Parameter LibDiamond.diamondCut(IDiamondCut.FacetCut[], address, bytes32, IDiamondCut.FacetCutAction, bytes4[]) (contracts/libraries/LibDiamond.sol#9) is not in mixedCase
Parameter LibDiamond.addReplaceRemoveFacetSelectors(uint256, bytes32, address, IDiamondCut.FacetCutAction, bytes4[]) (contracts/libraries/LibDiamond.sol#107-215) is not in mixedCase
Parameter LibDiamond.addReplaceRemoveFacetSelectors(uint256, bytes32, address, IDiamondCut.FacetCutAction, bytes4[], selectorCount) (contracts/libraries/LibDiamond.sol#71) is not in mixedCase
Parameter LibDiamond.addReplaceRemoveFacetSelectors(uint256, bytes32, address, IDiamondCut.FacetCutAction, bytes4[], selectorIndex_scope_0) (contracts/libraries/LibDiamond.sol#108) is not in mixedCase
Parameter LibDiamond.addReplaceRemoveFacetSelectors(uint256, bytes32, address, IDiamondCut.FacetCutAction, bytes4[], selectorIndex_scope_1) (contracts/libraries/LibDiamond.sol#109) is not in mixedCase
Parameter LibDiamond.addReplaceRemoveFacetSelectors(uint256, bytes32, address, IDiamondCut.FacetCutAction, bytes4[], selectorIndex_scope_2) (contracts/libraries/LibDiamond.sol#110) is not in mixedCase
Parameter LibDiamond.addReplaceRemoveFacetSelectors(uint256, bytes32, address, IDiamondCut.FacetCutAction, bytes4[], selectorIndex_scope_3) (contracts/libraries/LibDiamond.sol#111) is not in mixedCase
Parameter LibDiamond.addReplaceRemoveFacetSelectors(uint256, bytes32, address, IDiamondCut.FacetCutAction, bytes4[], selectorIndex_scope_4) (contracts/libraries/LibDiamond.sol#112) is not in mixedCase
Parameter LibDiamond.addReplaceRemoveFacetSelectors(uint256, bytes32, address, IDiamondCut.FacetCutAction, bytes4[], selectorIndex_scope_5) (contracts/libraries/LibDiamond.sol#113) is not in mixedCase
Parameter LibDiamond.addReplaceRemoveFacetSelectors(uint256, bytes32, address, IDiamondCut.FacetCutAction, bytes4[], selectorIndex_scope_6) (contracts/libraries/LibDiamond.sol#114) is not in mixedCase
Parameter LibDiamond.addReplaceRemoveFacetSelectors(uint256, bytes32, address, IDiamondCut.FacetCutAction, bytes4[], selectorIndex_scope_7) (contracts/libraries/LibDiamond.sol#115) is not in mixedCase
Parameter LibDiamond.addReplaceRemoveFacetSelectors(uint256, bytes32, address, IDiamondCut.FacetCutAction, bytes4[], selectorIndex_scope_8) (contracts/libraries/LibDiamond.sol#116) is not in mixedCase
Parameter LibDiamond.addReplaceRemoveFacetSelectors(uint256, bytes32, address, IDiamondCut.FacetCutAction, bytes4[], selectorIndex_scope_9) (contracts/libraries/LibDiamond.sol#117) is not in mixedCase
Parameter LibDiamond.addReplaceRemoveFacetSelectors(uint256, bytes32, address, IDiamondCut.FacetCutAction, bytes4[], selectorIndex_scope_10) (contracts/libraries/LibDiamond.sol#118) is not in mixedCase
Parameter LibDiamond.addReplaceRemoveFacetSelectors(uint256, bytes32, address, IDiamondCut.FacetCutAction, bytes4[], selectorIndex_scope_11) (contracts/libraries/LibDiamond.sol#119) is not in mixedCase
Parameter LibDiamond.addReplaceRemoveFacetSelectors(uint256, bytes32, address, IDiamondCut.FacetCutAction, bytes4[], selectorIndex_scope_12) (contracts/libraries/LibDiamond.sol#120) is not in mixedCase
Parameter LibDiamond.addReplaceRemoveFacetSelectors(uint256, bytes32, address, IDiamondCut.FacetCutAction, bytes4[], selectorIndex_scope_13) (contracts/libraries/LibDiamond.sol#121) is not in mixedCase
Parameter LibDiamond.addReplaceRemoveFacetSelectors(uint256, bytes32, address, IDiamondCut.FacetCutAction, bytes4[], selectorIndex_scope_14) (contracts/libraries/LibDiamond.sol#122) is not in mixedCase
Parameter LibDiamond.addReplaceRemoveFacetSelectors(uint256, bytes32, address, IDiamondCut.FacetCutAction, bytes4[], selectorIndex_scope_15) (contracts/libraries/LibDiamond.sol#123) is not in mixedCase
Parameter LibDiamond.addReplaceRemoveFacetSelectors(uint256, bytes32, address, IDiamondCut.FacetCutAction, bytes4[], selectorIndex_scope_16) (contracts/libraries/LibDiamond.sol#124) is not in mixedCase
Reference: https://github.com/crytic/slither/wikid/Better-DocmentationVariableNamesTooSimilar
INFODetectors:
Function LibDiamond.diamondCut(IDiamondCut.FacetCut[], address, bytes32) (contracts/libraries/LibDiamond.sol#107-205) contains magic numbers: 7, 3, 7, 3
Function LibDiamond.addReplaceRemoveFacetSelectors(uint256, bytes32, address, IDiamondCut.FacetCutAction, bytes4[]) (contracts/libraries/LibDiamond.sol#107-215) contains magic numbers: 7, 224, 3, 3, 7, 7, 5, 3, 7, 8
Function LibDiamond.initializeOwner(address, bytes) (contracts/libraries/LibDiamond.sol#107-215) contains magic number: 32
Function LibDiamondLoupeFacet.Facets() (contracts/utils/DiamondLoupeFacet.sol#26-76) contains magic numbers: 8, 5, 255
Function LibDiamondLoupeFacet.FaceFunctionSelectors(address) (contracts/utils/DiamondLoupeFacet.sol#81-107) contains magic numbers: 8, 5
Function LibDiamondLoupeFacet.FaceAddresses() (contracts/utils/DiamondLoupeFacet.sol#111-145) contains magic numbers: 8, 5
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number.md
INFODetectors:
In a function LibDiamond.addReplaceRemoveFacetSelectors(uint256, bytes32, address, IDiamondCut.FacetCutAction, bytes4[]) (contracts/libraries/LibDiamond.sol#107-215) variable LibDiamond.CLEAR_SELECTOR_MASK (contracts/libraries/LibDiamond.sol#50) is read multiple times
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/multiple_storage_read.md

```

IrrigationAccessControl.sol

INFO:Detectors:
Function LibDiamond.setContractOwner(address) (contracts/libraries/LibDiamond.sol#42-47) is a strange setter. Nothing is set in constructor or set in a function without using function parameters
Reference: https://github.com/crytic/slither/wikil/Detector-Documentation#uninitialized-local-variables

INFO:Detectors:
Function LibDiamond.addReplaceRemoveFaceSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]).selectorIndex_scope_0 (contracts/libraries/LibDiamond.sol#139) is a local variable never initialized
LibDiamond.addReplaceRemoveFaceSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]).selectorIndex_scope_3 (contracts/libraries/LibDiamond.sol#156) is a local variable never initialized
LibDiamond.addReplaceRemoveFaceSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]).selectorIndex_scope_5 (contracts/libraries/LibDiamond.sol#165) is a local variable never initialized
LibDiamond.addReplaceRemoveFaceSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]).selectorIndex_scope_7 (contracts/libraries/LibDiamond.sol#181) is a local variable never initialized
Reference: https://github.com/crytic/slither/wikil/Detector-Documentation#uninitialized-local-variables

INFO:Detectors:
Function LibDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#217-235) contains a low level call to a custom address
Reference: https://github.com/pessimistic-lotus/bloin/master/docs/_call_forward_to_protected.md

INFO:Detectors:
Function LibDiamond.addReplaceRemoveFaceSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]) (contracts/libraries/LibDiamond.sol#107-215) has a dubious typecast: address<=>bytes20
Function LibDiamond.addReplaceRemoveFaceSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]) (contracts/libraries/LibDiamond.sol#107-215) has a dubious typecast: bytes32=>bytes20
Function LibDiamond.addReplaceRemoveFaceSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]) (contracts/libraries/LibDiamond.sol#107-215) has a dubious typecast: uint16=>uint256
Reference: https://github.com/pessimistic-lotus/bloin/master/docs/_dubious_typecasts_and

INFO:Detectors:
LibDiamond.diamondStorage() (contracts/libraries/LibDiamond.sol#3-30) uses assembly
- INLINE ASM (contracts/libraries/LibDiamond.sol#217-235) uses assembly
LibDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#217-235) uses assembly
- INLINE ASM (contracts/libraries/LibDiamond.sol#227-230)
LibDiamond.enforceIsContractCode(address,string) (contracts/libraries/LibDiamond.sol#237-243) uses assembly
- INLINE ASM (contracts/libraries/LibDiamond.sol#239-241)
Reference: https://github.com/crytic/slither/wikil/Detector-Documentation#assembly-usage

INFO:Detectors:
LibDiamond.addReplaceRemoveFaceSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]) (contracts/libraries/LibDiamond.sol#107-215) has a high cyclomatic complexity (12).
Reference: https://github.com/crytic/slither/wikil/Detector-Documentation#cyclomatic-complexity

INFO:Detectors:
IrrigationAccessControl . IrrigationAccessControl_init() (contracts/utils/IrrigationAccessControl.sol#13-16) is never used and should be removed
IrrigationAccessControl . IrrigationAccessControl_init() (contracts/utils/IrrigationAccessControl.sol#18-22) is never used and should be removed
LibDiamond.addReplaceRemoveFaceSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]) (contracts/libraries/LibDiamond.sol#107-215) is never used and should be removed
LibDiamond.contractOwner() (contracts/libraries/LibDiamond.sol#49-51) is never used and should be removed
LibDiamond.diamondStorage() (contracts/libraries/LibDiamond.sol#3-30) is never used and should be removed
LibDiamond.enforceIsContractCode(address,string) (contracts/libraries/LibDiamond.sol#237-243) is never used and should be removed
LibDiamond.enforceContractOwner() (contracts/libraries/LibDiamond.sol#53-55) is never used and should be removed
LibDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#217-235) is never used and should be removed
LibDiamond.setContractOwner(address) (contracts/libraries/LibDiamond.sol#42-47) is never used and should be removed
Reference: https://github.com/crytic/slither/wikil/Detector-Documentation#dead-code

INFO:Detectors:
Pragme version='0.8.17' (contracts/interfaces/IDiamondCut.sol#2) allow old version
Pragme version='0.8.17' (contracts/libraries/LibDiamond.sol#2) allow old versions
Pragme version='0.8.17' (contracts/utils/IrrigationAccessControl.sol#3) allow old versions
Pragme version='0.8.17' (contracts/utils/IrrigationAccessControl.sol#4) allow old versions
Reference: https://github.com/crytic/slither/wikil/Detector-Documentation#incorrect-versions-of-solidity

INFO:Detectors:
Low level call to LibDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#217-235);
LibDiamond.setContractOwner(address) . init.delegatecall(.callData) (contracts/libraries/LibDiamond.sol#222)
Reference: https://github.com/crytic/slither/wikil/Detector-Documentation#low-level-calls

INFO:Detectors:
Parameter LibDiamond.setContractOwner(address)._newAddress (contracts/libraries/LibDiamond.sol#42) is not in mixedCase
Parameter LibDiamond.initializeDiamondCut(address,bytes).calldata (contracts/libraries/LibDiamond.sol#217-235) is not in mixedCase
Parameter LibDiamond.addReplaceRemoveFaceSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]).selectorCount (contracts/libraries/LibDiamond.sol#108) is not in mixedCase
Parameter LibDiamond.addReplaceRemoveFaceSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]).selectorScope_0 (contracts/libraries/LibDiamond.sol#109) is not in mixedCase
Parameter LibDiamond.addReplaceRemoveFaceSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]).newFaceAddress (contracts/libraries/LibDiamond.sol#110) is not in mixedCase
Parameter LibDiamond.addReplaceRemoveFaceSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]).calldata (contracts/libraries/LibDiamond.sol#111) is not in mixedCase
Parameter LibDiamond.initializeDiamondCut(address,bytes).init (contracts/libraries/LibDiamond.sol#217) is not in mixedCase
Parameter LibDiamond.initializeDiamondCut(address,bytes).callData (contracts/libraries/LibDiamond.sol#217) is not in mixedCase
Parameter LibDiamond.enforceIsContractCode(address,string).contract (contracts/libraries/LibDiamond.sol#237) is not in mixedCase
Parameter LibDiamond.enforceContractOwner().calldata (contracts/libraries/LibDiamond.sol#53) is not in mixedCase
Function IrrigationAccessControl . IrrigationAccessControl_init() (contracts/utils/IrrigationAccessControl.sol#13-16) is not in mixedCase
Function IrrigationAccessControl . IrrigationAccessControl_init() (contracts/utils/IrrigationAccessControl.sol#18-22) is not in mixedCase
Reference: https://github.com/crytic/slither/wikil/Detector-Documentation#convention-to-solidity-naming-conventions

INFO:Detectors:
LibDiamond.addReplaceRemoveFaceSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]).idFacet_scope_2 (contracts/libraries/LibDiamond.sol#141) is too similar to LibDiamond.addReplaceRemoveFaceSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]).idFacet_scope_5 (contracts/libraries/LibDiamond.sol#171)
Variable LibDiamond.addReplaceRemoveFaceSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]).selectorIndex_scope_0 (contracts/libraries/LibDiamond.sol#139) is too similar to LibDiamond.addReplaceRemoveFaceSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]).selectorIndex_scope_3 (contracts/libraries/LibDiamond.sol#156)
Variable LibDiamond.addReplaceRemoveFaceSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]).selectorIndex_scope_1 (contracts/libraries/LibDiamond.sol#140) is too similar to LibDiamond.addReplaceRemoveFaceSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]).selectorIndex_scope_7 (contracts/libraries/LibDiamond.sol#181)
Reference: https://github.com/crytic/slither/wikil/Detector-Documentation#variable-names-too-similar

INFO:Detectors:
Function LibDiamond.diamondCut(uint256,bytes32,address,bytes32,IDIamondCut.FacetCutAction,bytes4[]).calldata (contracts/libraries/LibDiamond.sol#195) contains magic numbers: 7, 3, 7, 3
Function LibDiamond.addReplaceRemoveFaceSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]).calldata (contracts/libraries/LibDiamond.sol#107-215) contains magic numbers: 7, 224, 3, 3, 7, 7, 5, 32
Reference: https://github.com/pessimistic-lotus/bloin/master/docs/magic_number_md

INFO:Detectors:
Function LibDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#217-235) variable LibDiamond.CLEAR_SELECTOR_MASK (0x00) is multiple times
Reference: https://github.com/pessimistic-lotus/bloin/master/docs/multiple_storage_read_md

OwnershipFacet.sol

```
INFO[Detectors]: 
Function LibDiamond.setContractOwner(address) (contracts/libraries/LibDiamond.sol#02-07) is a strange setter. Nothing is set in constructor or set in a function without using function parameters
Reference: https://github.com/pessimistic-lotus/blox/master/docs/strange_setter.md

INFO[Detectors]: 
LibDiamond._addPlaceRemoveFactSetSelectors(juln256_byt32[], address, LibDiamondCut.FacetCutAction_bytetd[]), selectorIndex (contracts/libraries/LibDiamond.sol#118) is a local variable never initialized
LibDiamond._addPlaceRemoveFactSetSelectors(juln256_byt32[], address, LibDiamondCut.FacetCutAction_bytetd[]), selectorIndex_scope_0 (contracts/libraries/LibDiamond.sol#139) is a local variable never initialized
LibDiamond._addPlaceRemoveFactSetSelectors(juln256_byt32[], address, LibDiamondCut.FacetCutAction_bytetd[]), selectorIndex_scope_1 (contracts/libraries/LibDiamond.sol#156) is a local variable never initialized
LibDiamond._addPlaceRemoveFactSetSelectors(juln256_byt32[], address, LibDiamondCut.FacetCutAction_bytetd[]), selectorIndex_scope_2 (contracts/libraries/LibDiamond.sol#175) is a local variable never initialized
Reference: https://github.com/pessimistic-lotus/blox/master/docs/unprotected_initializer.md

INFO[Detectors]: 
LibDiamond._addPlaceRemoveFactSetSelectors(juln256_byt32[], address, LibDiamondCut.FacetCutAction_bytetd[]), selectorIndex (contracts/libraries/LibDiamond.sol#118) is a local variable never initialized
LibDiamond._addPlaceRemoveFactSetSelectors(juln256_byt32[], address, LibDiamondCut.FacetCutAction_bytetd[]), selectorIndex_scope_0 (contracts/libraries/LibDiamond.sol#139) is a local variable never initialized
LibDiamond._addPlaceRemoveFactSetSelectors(juln256_byt32[], address, LibDiamondCut.FacetCutAction_bytetd[]), selectorIndex_scope_1 (contracts/libraries/LibDiamond.sol#156) is a local variable never initialized
LibDiamond._addPlaceRemoveFactSetSelectors(juln256_byt32[], address, LibDiamondCut.FacetCutAction_bytetd[]), selectorIndex_scope_2 (contracts/libraries/LibDiamond.sol#175) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/DetectorDocumentation#uninitialized_local_variables

INFO[Detectors]: 
Function LibDiamond.initializeLibDiamond(address, bytes) (contracts/libraries/LibDiamond.sol#0217-235) contains a low level call to a custom address
Reference: https://github.com/pessimistic-lotus/blox/master/docs/call_forward_to_protected.md

INFO[Detectors]: 
LibDiamond._addPlaceRemoveFactSetSelectors(juln256_byt32[], address, LibDiamondCut.FacetCutAction_bytetd[]) (contracts/libraries/LibDiamond.sol#107-215) has a dubious typecast: address<=>bytes256
Function LibDiamond._addPlaceRemoveFactSetSelectors(juln256_byt32[], address, LibDiamondCut.FacetCutAction_bytetd[]) (contracts/libraries/LibDiamond.sol#107-215) has a dubious typecast: bytes32<=>bytes256
Function LibDiamond._addPlaceRemoveFactSetSelectors(juln256_byt32[], address, LibDiamondCut.FacetCutAction_bytetd[]) (contracts/libraries/LibDiamond.sol#107-215) has a dubious typecast: bytes32<=>bytes256
Function LibDiamond._slitherConstructorConstantVariableS() (contracts/libraries/LibDiamond.sol#13-244) has a dubious typecast: uint32<=>uint256
```

```

INFODetectors:
LibDiamond.diamondStorage() (contracts/libraries/LibDiamond.sol#33-38) uses assembly
  - INLINE ASM (contracts/libraries/LibDiamond.sol#35-37)
LibDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#217-235) uses assembly
  - INLINE ASM (contracts/libraries/LibDiamond.sol#222-223)
  - INLINE ASM (contracts/libraries/LibDiamond.sol#237-243)
  - INLINE ASM (contracts/libraries/LibDiamond.sol#239-241)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFODetectors:
Different versions of Solidity are used:
  - Version used: ["^0.8.0", "<0.8.17"]
    - 0.8.0 (contracts/interfaces/IERC1730Upgradable.sol#2)
    - 0.8.17 (contracts/interfaces/IDiamondCut.sol#2)
    - 0.8.17 (contracts/interfaces/IDiamondCutFacet.sol#2)
    - 0.8.17 (contracts/interfaces/IDiamondCutFacetAction.sol#2)
    - 0.8.17 (contracts/interfaces/IDiamondCutFacetActionBytecode.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFODetectors:
LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]) (contracts/libraries/LibDiamond.sol#107-215) has a high cyclomatic complexity (12).
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#cyclomatic-complexity
INFODetectors:
LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]) (contracts/libraries/LibDiamond.sol#107-215) is never used and should be removed
LibDiamond.initializeDiamondCut(address,bytes,bytes) (contracts/libraries/LibDiamond.sol#68-105) is never used and should be removed
LibDiamond.initializeDiamondCut(address,bytes,bytes) (contracts/libraries/LibDiamond.sol#237-243) is never used and should be removed
LibDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#217-235) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFODetectors:
Pragma version<=0.8.17 (contracts/interfaces/IERC730Upgradable.sol#2) allows old versions
Pragma version>=0.8.0 (contracts/libraries/LibDiamond.sol#2) allows old versions
Pragma version<=0.8.17 (contracts/libraries/LibDiamond.sol#2) allows old versions
Pragma version<=0.8.17 (contracts/utils/OwnableFacet.sol#2) allows old versions
Solidity 0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFODetectors:
Low level call in LibDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#217-235):
  (success,error) = init.delegatecall(_widder) (contracts/libraries/LibDiamond.sol#222)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFODetectors:
Parameter LibDiamond.setContractOwner(address,_newOwner) (contracts/libraries/LibDiamond.sol#42) is not in mixedCase
Parameter LibDiamond.diamondCut(IDiamondCut.FacetCut[],address,bytes)_diamondCut (contracts/libraries/LibDiamond.sol#93) is not in mixedCase
Parameter LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]) (contracts/libraries/LibDiamond.sol#107-215) is not in mixedCase
Parameter LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]) (contracts/libraries/LibDiamond.sol#217) is not in mixedCase
Parameter LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]) _selectorCount (contracts/libraries/LibDiamond.sol#108) is not in mixedCase
Parameter LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]) _selectorSlot (contracts/libraries/LibDiamond.sol#109) is not in mixedCase
Parameter LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]) _selectorTotal (contracts/libraries/LibDiamond.sol#110) is not in mixedCase
Parameter LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]) _selectors (contracts/libraries/LibDiamond.sol#111) is not in mixedCase
Parameter LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]) _selectors (contracts/libraries/LibDiamond.sol#112) is not in mixedCase
Parameter LibDiamond.initializeDiamondCut(address,bytes)_init (contracts/libraries/LibDiamond.sol#217) is not in mixedCase
Parameter LibDiamond.initializeDiamondCut(address,bytes)_initData (contracts/libraries/LibDiamond.sol#171) is not in mixedCase
Parameter LibDiamond.initializeDiamondCut(address,bytes)_initData (contracts/libraries/LibDiamond.sol#171) is not in mixedCase
Parameter LibDiamond.enforceRoleContractCode(address,string)_errorMessage (contracts/libraries/LibDiamond.sol#237) is not in mixedCase
Parameter OwnershipFacet.transferOwnership(address,_newOwner) (contracts/utils/OwnershipFacet.sol#8) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#solidity-naming-conventions
INFODetectors:
Variable LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]) _oldfacet_scope_2 (contracts/libraries/LibDiamond.sol#141) is too similar to LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]) _oldfacet_scope_1 [LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]) _oldfacet_scope_1]
Variable LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]) _selectorIndex_scope_0 (contracts/libraries/LibDiamond.sol#139) is too similar to LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]) _selectorIndex_scope_3 [LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]) _selectorIndex_scope_3]
Variable LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]) _selector_scope_4 (contracts/libraries/LibDiamond.sol#170) is too similar to LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[])
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFODetectors:
Function LibDiamond.diamondCut(IDiamondCut.FacetCut[],address,bytes) (contracts/libraries/LibDiamond.sol#68-105) contains magic numbers: 7, 3, 7, 3
Function LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]) (contracts/libraries/LibDiamond.sol#107-215) contains magic numbers: 7, 224, 3, 3, 7, 7, 5, 3, 7, 8
Function LibDiamond.initializeDiamondCut(address,bytes) (contracts/libraries/LibDiamond.sol#217-235) contains magic number: 32
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number.md
INFODetectors:
In function LibDiamond.addReplaceRemoveFacetSelectors(uint256,bytes32,address,IDIamondCut.FacetCutAction,bytes4[]) (contracts/libraries/LibDiamond.sol#107-215) variable LibDiamond.CLEAR_SELECTOR_MASK (contracts/libraries/LibDiamond.sol#60) is read multiple times
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/multiple_storage_read.md

```

PodTransferHelper.sol

```

INFODetectors:
PodTransferHelper.getPlotSplittedByFM(uint256,uint256,uint256,uint256) (contracts/libraries/Plotter.sol#39-36) is never used and should be removed
PodTransferHelper.getPlotWithOffset(uint256,uint256,uint256,uint256) (contracts/libraries/Plotter.sol#39-72) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFODetectors:
Pragma version<=0.8.17 (contracts/libraries/PodTransferHelper.sol#2) allows old versions
Solidity 0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFODetectors:
Function PodTransferHelper.getPlotWithOffset(uint256,uint256,uint256,uint256,uint256,uint256) (contracts/libraries/Plotter.sol#39-72) contains magic number: 3
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number.md

```

LibPrice.sol

```

INFODetectors:
LibPrice.getPriceOnPools(uint256,uint256,uint256,uint256) (contracts/libraries/Oracle/LibPrice.sol#19-36) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFODetectors:
Pragma version<=0.8.17 (contracts/libraries/Oracle/LibPrice.sol#2) allows old versions
Solidity 0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFODetectors:
In a function LibPrice.getPriceOnPools(uint256,uint256,uint256,uint256) (contracts/libraries/Oracle/LibPrice.sol#19-36) variable LibPrice.BDV_FACTOR (contracts/libraries/Oracle/LibPrice.sol#7) is read multiple times
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/multiple_storage_read.md

```

BeanPriceOracle.sol

```

INFODetectors:
Different versions of Solidity are used:
  - Version used: ["^0.8.0", "<0.8.17"]
    - 0.8.17 (contracts/interfaces/ICustomOracle.sol#2)
    - 0.8.17 (contracts/oracles/BeanPriceOracle.sol#2)
    - 0.8.17 (contracts/interfaces/IBeanstalkPrice.sol#2)
    - 0.8.17 (contracts/libraries/Constants.sol#4)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFODetectors:
Pragma version<=0.8.17 (contracts/interfaces/IBeanstalkPrice.sol#2) allows old versions
Pragma version<=0.8.17 (contracts/interfaces/ICustomOracle.sol#2) allows old versions
Pragma version<=0.8.17 (contracts/libraries/Constants.sol#4) allows old versions
Pragma version<=0.8.17 (contracts/oracles/BeanPriceOracle.sol#2) allows old versions
Solidity 0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFODetectors:
Function BeanPriceOracle.latestPrice() (contracts/oracles/BeanPriceOracle.sol#14-16) contains magic number: 1e12
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number.md

```

ChainlinkOracle.sol

```

INFODetectors:
ChainlinkOracle.setAdmin(address)_newAdmin (contracts/oracles/ChainlinkOracle.sol#68) lacks a zero-check on :
  - address _newAdmin (contracts/oracles/ChainlinkOracle.sol#62)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFODetectors:
ChainlinkOracle.getChainlinkPrice(AggregatorV2Interface) (contracts/oracles/ChainlinkOracle.sol#30-39) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFODetectors:
Pragma version<=0.8.17 (contracts/libraries/Constants.sol#2) allows old versions
Pragma version<=0.8.17 (contracts/oracles/AggregatorV2Interface.sol#3) allows old versions
Pragma version<=0.8.17 (contracts/libraries/ChainlinkOracle.sol#3) allows old versions
Solidity 0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFODetectors:
Function ChainlinkOracle.getChainlinkPrice(AggregatorV2Interface) (contracts/oracles/ChainlinkOracle.sol#30-39) contains magic numbers: 10, 10
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number.md

```

UniswapV3Twap.sol

- The unprotected initialize issues flagged by Slither were checked individually and are false positives.
 - No major issues found by Slither.

6.2 AUTOMATED SECURITY SCAN

Description:

Halborn used automated security scanners to assist with detection of well-known security issues and to identify low-hanging fruits on the targets for this engagement. Among the tools used was MythX, a security analysis service for Ethereum smart contracts. MythX performed a scan on the smart contracts and sent the compiled results to the analyzers to locate any vulnerabilities.

MythX results:

SprinklerUpgradeable.sol

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.
17	(SWC-123) Requirement Violation	Low	Requirement violation.
139	(SWC-107) Reentrancy	Low	Read of persistent state following external call.
156	(SWC-123) Requirement Violation	Low	Requirement violation.

WaterTowerUpgradeable.sol

Line	SWC Title	Severity	Short Description
27	(SWC-123) Requirement Violation	Low	Requirement violation.
67	(SWC-107) Reentrancy	Low	A call to a user-supplied address is executed.

TrancheBondUpgradeable.sol

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.
27	(SWC-123) Requirement Violation	Low	Requirement violation.
289	(SWC-123) Requirement Violation	Low	Requirement violation.

AuctionUpgradeable.sol

Line	SWC Title	Severity	Short Description
29	(SWC-123) Requirement Violation	Low	Requirement violation.
123	(SWC-123) Requirement Violation	Low	Requirement violation.
143	(SWC-107) Reentrancy	Low	Write to persistent state following external call.
143	(SWC-107) Reentrancy	Low	Read of persistent state following external call.
172	(SWC-107) Reentrancy	Low	Write to persistent state following external call.
172	(SWC-107) Reentrancy	Low	Read of persistent state following external call.
186	(SWC-107) Reentrancy	Low	Read of persistent state following external call.
188	(SWC-107) Reentrancy	Low	Read of persistent state following external call.

ERC1155WhitelistUpgradeable.sol

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

WaterCommonUpgradeable.sol

No issues found by MythX.

PriceOracleUpgradeable.sol

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.
14	(SWC-123) Requirement Violation	Low	Requirement violation.
23	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
24	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
87	(SWC-123) Requirement Violation	Low	Requirement violation.

PodsOracleUpgradeable.sol

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

WaterUpgradeable.sol

No issues found by MythX.

IrrigationDiamond.sol

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

Diamond.sol

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.
21	(SWC-110) Assert Violation	Unknown	Out of bounds array access
22	(SWC-110) Assert Violation	Unknown	Out of bounds array access
33	(SWC-109) Uninitialized Storage Pointer	Medium	Dangerous use of uninitialized storage variables.

DiamondCutFacet.sol

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.
15	(SWC-123) Requirement Violation	Low	Requirement violation.

DiamondLoupeFacet.sol

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

EIP2535Initializable.sol

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

IrrigationAccessControl.sol

Line	SWC Title	Severity	Short Description
3	(SWC-103) Floating Pragma	Low	A floating pragma is set.

OwnershipFacet.sol

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

PodTransferHelper.sol

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

LibPrice.sol

No issues found by MythX.

BeanPriceOracle.sol

No issues found by MythX.

ChainlinkOracle.sol

Line	SWC Title	Severity	Short Description
3	(SWC-103) Floating Pragma	Low	A floating pragma is set.

UniswapV3Twap.sol

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.

- MythX flagged some integer overflows and underflows which all were false positives, as the contracts are using Solidity ^0.8.17 version. After the Solidity version 0.8.0 Arithmetic operations revert to underflow and overflow by default.
- MythX also flagged some assert violations, which were all considered to be false positives.
- No major issues were found by MythX.

THANK YOU FOR CHOOSING
 HALBORN