HALBORN

Beanstalk - Silo V3

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **April 20th, 2023 - May 15th, 2023**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	05/12/2023	Omar Alshaeb
0.2	Document Updates	05/13/2023	Omar Alshaeb
0.3	Draft Review	05/15/2023	Gokberk Gulgun
0.4	Draft Review	05/15/2023	Gabi Urrutia
0.5	Document Updates	06/04/2023	Omar Alshaeb
0.6	Draft Updates Review	06/05/2023	Gokberk Gulgun
0.7	Draft Updates Review	06/05/2023	Gabi Urrutia
0.8	Document Updates	06/29/2023	Omar Alshaeb
1.0	Remediation Plan	06/29/2023	Omar Alshaeb
1.1	Remediation Plan Review	06/30/2023	Gokberk Gulgun
1.2	Remediation Plan Review	06/30/2023	Gabi Urrutia

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 Gulgun	Halborn	Gokberk.Gulgun@halborn.com
Omar Alshaeb	Halborn	Omar.Alshaeb@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Beanstalk requires the ability to coordinate protocol upgrades. The Silo (the Beanstalk DAO) uses the Stalk System to create protocol-native financial incentives that coordinate Beanstalk upgrades and consistently improve security, stability, and liquidity. Stakeholders earn passive yield from participation in governance of Beanstalk upgrades and passive contributions to security, stability, and liquidity. Active contributions to peg maintenance within the Silo earn additional Stalk.

Beanstalk engaged Halborn to conduct a security audit on their Silo V3 smart contracts beginning on April 20th, 2023 and ending on May 15th, 2023. The security assessment was scoped to the smart contracts provided to the Halborn team. Furthermore, the audit was extended to include minor final changes.

1.2 AUDIT SUMMARY

The team at Halborn was provided four weeks for the engagement and assigned a full-time security engineer to audit the security of the smart contract. 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.

Moreover, the audit was extended by one week to include minor final changes to the Silo V3.

The purpose of this audit 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 addressed and accepted by the Beanstalk team.

1.3 SCOPE

1. IN-SCOPE:

The security assessment was scoped to the following smart contracts:

- protocol/contracts/beanstalk/silo/ApprovalFacet.sol
- protocol/contracts/beanstalk/silo/BDVFacet.sol
- protocol/contracts/beanstalk/silo/ConvertFacet.sol
- protocol/contracts/beanstalk/silo/MigrationFacet.sol
- protocol/contracts/beanstalk/silo/WhitelistFacet.sol
- protocol/contracts/beanstalk/silo/SiloFacet/Silo.sol
- protocol/contracts/beanstalk/silo/SiloFacet/SiloExit.sol
- protocol/contracts/beanstalk/silo/SiloFacet/SiloFacet.sol
- protocol/contracts/beanstalk/silo/SiloFacet/TokenSilo.sol
- protocol/contracts/beanstalk/AppStorage.sol
- protocol/contracts/libraries/Token/LibTransfer.sol
- protocol/contracts/libraries/Silo/*
- protocol/contracts/libraries/Convert/*
- protocol/contracts/beanstalk/metadata/MetadataFacet.sol
- protocol/contracts/beanstalk/metadata/MetadataImage.sol

Commit ID: e3e92b1a658a224af6c6e0e03710ecc2e5a4ce24

Also, the new commit ID including the minor final changes made by the team, where the audit was focused on the MigrationFacet (adding a Merkle root to fix the stalk/seed discrepancy), and the new variable added within AppStorage called depositedBDV:

Commit ID: 58c30d31ba4a934b01ef0d51dae48bc8dde140a3

Moreover, focusing on the introduction of MetadataFacet and MetadataImage contracts. These contracts contain logic for generating a dynamic SVG representation of the ERC1155 deposit and additional metadata that users may want.

Commit ID: 24bf3d33355f516648b02780b4b232181afde200

2. REMEDIATION PR/COMMITS:

- Fix Commit ID (HAL-03): da370ddcc86490b7b37c497b190a8bdaf62eb62c
- Fix Commit ID (HAL-04): 55813422bc515a5e36469108d4c4f835158fa8fd

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

- Research into architecture 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 hotspots 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) Specific (AO:S)	1 0.2
Attack Cost (AC)	Low (AC:L) Medium (AC:M) High (AC:H)	1 0.67 0.33
Attack Complexity (AX)	Low (AX:L) Medium (AX:M) High (AX:H)	1 0.67 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) Partial (R:P) Full (R:F)	1 0.5 0.25
Scope (s)	Changed (S:C) Unchanged (S:U)	1.25 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

3. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	0	2	2

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
APPROVALFACET.PERMITDEPOSIT() CAN INVALIDATE OTHER OWNER PERMITS	Low (2.7)	RISK ACCEPTED
INCONSISTENCY WITH FUNCTION RETURN VALUES	Low (2.5)	RISK ACCEPTED
MISSING USEFUL INFORMATION WITHIN SILOFACET.PLANT() FUNCTION RETURN VALUES	Informational (1.2)	SOLVED - 06/29/2023
USE ++I INSTEAD OF I++ IN LOOPS FOR GAS OPTIMIZATION	Informational (0.0)	SOLVED - 06/29/2023

FINDINGS & TECH DETAILS

4.1 (HAL-01) APPROVALFACET.PERMITDEPOSIT() CAN INVALIDATE OTHER OWNER PERMITS - LOW (2.7)

Description:

An owner of some deposits for a specific token can sign off-chain a permit to transfer some amount of tokens of those deposits to a specific spender or more than one spender. The scenario where it can be an issue is if the owner signs for more than one spender (with consecutive nonces, for each spender a different one), as he does not know the order of those spenders sending the transactions for permitDeposit, some spenders will not be able to claim their allowance as the _useNonce function will return the incorrect nonce for them.

Code Location:

Listing 1: ApprovalFacet.sol (Line 158)

```
148 function permitDeposit(
149     address owner,
150     address spender,
151     address token,
152     uint256 value,
153     uint256 deadline,
154     uint8 v,
155     bytes32 r,
156     bytes32 s
157 ) external payable nonReentrant {
158     LibSiloPermit.permit(owner, spender, token, value, deadline, v
 ↳ , r, s);
159     LibSiloPermit._approveDeposit(owner, spender, token, value);
160 }
```

Listing 2: LibSiloPermit.sol (Line 75)

```
57 function permit(
58     address owner,
59     address spender,
60     address token,
61     uint256 value,
62     uint256 deadline,
63     uint8 v,
64     bytes32 r,
65     bytes32 s
66 ) internal {
67     require(block.timestamp <= deadline, "Silo: permit expired
   ↳ deadline");
68     bytes32 structHash = keccak256(
69         abi.encode(
70             DEPOSIT_PERMIT_TYPEHASH,
71             owner,
72             spender,
73             token,
74             value,
75             _useNonce(owner),
76             deadline
77         )
78     );
79     bytes32 hash = _hashTypedDataV4(structHash);
80     address signer = ECDSA.recover(hash, v, r, s);
81     require(signer == owner, "Silo: permit invalid signature");
82 }
```

Listing 3: LibSiloPermit.sol (Line 136)

```
133 function _useNonce(address owner) internal returns (uint256
    ↳ current) {
134     AppStorage storage s = LibAppStorage.diamondStorage();
135     current = s.a[owner].depositPermitNonces;
136     ++s.a[owner].depositPermitNonces;
137 }
```

Proof of Concept:

1. User1 signs an approval of 100 tokens amount of deposit to User2 with nonce 0.
2. User1 signs an approval of 100 tokens amount of deposit to User3 with nonce 1.
3. User1 signs an approval of 100 tokens amount of deposit to User4 with nonce 2.
4. User1 signs an approval of 100 tokens amount of deposit to User5 with nonce 3.
5. User2 does never send the transaction.
6. Internally, that means User3, User4, and User5 will never be able to claim their allowance as the transaction will revert for them and over time the expiration time will be reached.

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:C/D:L/Y:N/R:F/S:U (2.7)

Recommendation:

Instead of updating the owner nonce each time the permitDeposit function is called by a spender, the use of a mapping for each owner to know whether a specific nonce has been previously used or not is recommended to avoid this kind of scenario.

Remediation Plan:

RISK ACCEPTED: The Beanstalk team accepted the risk of the issue.

4.2 (HAL-02) INCONSISTENCY WITH FUNCTION RETURN VALUES - LOW (2.5)

Description:

The deposit function says that returns the amount, bdv, and stem after its execution but actually returns the amount, the stalk minted to the user depositing tokens to the silo, and the stem for the deposit. This can lead to further confusion within the overall protocol.

Code Location:

Listing 4: SiloFacet.sol (Lines 59,67)

```
50 function deposit(
51     address token,
52     uint256 _amount,
53     LibTransfer.From mode
54 )
55     external
56     payable
57     nonReentrant
58     mowSender(token)
59     returns (uint256 amount, uint256 bdv, int96 stem)
60 {
61     amount = LibTransfer.receiveToken(
62         IERC20(token),
63         _amount,
64         msg.sender,
65         mode
66     );
67     (bdv, stem) = _deposit(msg.sender, token, amount);
68 }
```

Listing 5: TokenSilo.sol (Line 161)

```
157 function _deposit(
158     address account,
159     address token,
```

```
160     uint256 amount
161 ) internal returns (uint256 stalk, int96 stem){
162     stalk = LibTokenSilo.deposit(
163         account,
164         token,
165         stem = LibTokenSilo.stemTipForToken(token),
166         amount
167     );
168     LibSilo.mintStalk(account, stalk);
169 }
```

BVSS:

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

Recommendation:

Change the name of bdv variable to stalk within the deposit function:

Listing 6: SiloFacet.sol (Lines 59,67)

```
50 function deposit(
51     address token,
52     uint256 _amount,
53     LibTransfer.From mode
54 )
55     external
56     payable
57     nonReentrant
58     mowSender(token)
59     returns (uint256 amount, uint256 stalk, int96 stem)
60 {
61     amount = LibTransfer.receiveToken(
62         IERC20(token),
63         _amount,
64         msg.sender,
65         mode
66     );
67     (stalk, stem) = _deposit(msg.sender, token, amount);
68 }
```

Remediation Plan:

RISK ACCEPTED: The Beanstalk team accepted the risk of the issue.

4.3 (HAL-03) MISSING USEFUL INFORMATION WITHIN SILOFACET.PLANT() FUNCTION RETURN VALUES - INFORMATIONAL (1.2)

Description:

The plant() function performs an internal deposit for the user who is planting, but the stem on which the deposit is made is not returned as a return value as the normal deposit function does.

Code Location:

Listing 7: SiloFacet.sol (Line 307)

```
306 function plant(address token) external payable returns (uint256
 ↳ beans) {
307     return _plant(msg.sender, token);
308 }
```

Listing 8: Silo.sol (Line 118)

```
 97 function _plant(address account, address token) internal returns (
 ↳ uint256 beans) {
 98     // Need to Mow for `account` before we calculate the balance
 ↳ of
 99     // Earned Beans.
100
101     // per the zero withdraw update, planting is handled
 ↳ differently
102     // depending whether or not the user plants during the vesting
 ↳  period of beanstalk.
103     // during the vesting period, the earned beans are not issued
 ↳ to the user.
104     // thus, the roots calculated for a given user is different.
105     // This is handled by the super mow function, which stores the
 ↳  difference in roots.
106     LibSilo._mow(account, token);
```

```
107     uint256 accountStalk = s.a[account].s.stalk;
108
109     // Calculate balance of Earned Beans.
110     beans = _balanceOfEarnedBeans(account, accountStalk);
111     s.a[account].deltaRoots = 0; // must be 0'd, as calling
 ↳ balanceOfEarnedBeans would give a invalid amount of beans.
112     if (beans == 0) return 0;
113
114     // Reduce the Silo's supply of Earned Beans.
115     s.earnedBeans = s.earnedBeans.sub(uint128(beans));
116
117     // Deposit Earned Beans if there are any. Note that 1 Bean = 1
 ↳  BDV.
118     LibTokenSilo.addDepositToAccount(
119         account,
120         C.beanAddress(),
121         LibTokenSilo.stemTipForToken(token),
122         beans, // amount
123         beans, // bdv
124         LibTokenSilo.Transfer.emitTransferSingle
125     );
126     s.a[account].deltaRoots = 0; // must be 0'd, as calling
 ↳ balanceOfEarnedBeans would give a invalid amount of beans.
127
128     // Earned Stalk associated with Earned Beans generate more
 ↳ Earned Beans automatically (i.e., auto compounding).
129     // Earned Stalk are minted when Earned Beans are minted during
 ↳  Sunrise. See {Sun.sol:rewardToSilo} for details.
130     // Similarly, `account` does not receive additional Roots from
 ↳  Earned Stalk during a Plant.
131     // The following lines allocate Earned Stalk that has already
 ↳ been minted to `account`.
132     uint256 stalk = beans.mul(C.getStalkPerBean());
133     s.a[account].s.stalk = accountStalk.add(stalk);
134
135
136     emit StalkBalanceChanged(account, int256(stalk), 0);
137     emit Plant(account, beans);
138 }
```

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:L/D:N/Y:N/R:P/S:U (1.2)

Recommendation:

Apart from the beans which are already being returned by the plant() function, returning the stem on which the internal deposit is made is recommended.

Remediation Plan:

SOLVED: The Beanstalk team solved the issue with the following commit ID.

Commit ID : da370ddcc86490b7b37c497b190a8bdaf62eb62c

4.4 (HAL-04) USE ++I INSTEAD OF I++ IN LOOPS FOR GAS OPTIMIZATION - INFORMATIONAL (0.0)

Description:

In the loop within the transferDeposits and safeBatchTransferFrom functions, the variable i is incremented using i++. It is known that, in loops, using ++i costs less gas per iteration than i++. This also affects variables incremented inside the loop code block.

Code Location:

Listing 9: SiloFacet.sol (Line 184)

```
176 function transferDeposits(
177     address sender,
178     address recipient,
179     address token,
180     int96[] calldata stem,
181     uint256[] calldata amounts
182 ) public payable nonReentrant returns (uint256[] memory bdvs) {
183     require(amounts.length > 0, "Silo: amounts array is empty");
184     for (uint256 i = 0; i < amounts.length; i++) {
185         require(amounts[i] > 0, "Silo: amount in array is 0");
186         if (sender != msg.sender) {
187             LibSiloPermit._spendDepositAllowance(sender, msg.
 ↳ sender, token, amounts[i]);
188         }
189     }
190
191     LibSilo._mow(sender, token);
192     // Need to update the recipient's Silo as well.
193     LibSilo._mow(recipient, token);
194     bdvs = _transferDeposits(sender, recipient, token, stem,
 ↳ amounts);
195 }
```

Listing 10: SiloFacet.sol (Line 255)

```
243 function safeBatchTransferFrom(
244     address sender,
245     address recipient,
246     uint256[] calldata depositIds,
247     uint256[] calldata amounts,
248     bytes calldata
249 ) external {
250     require(depositIds.length == amounts.length, "Silo: depositIDs
 ↳  and amounts arrays must be the same length");
251     require(recipient != address(0), "ERC1155: transfer to the
 ↳ zero address");
252     // allowance requirements are checked in transferDeposit
253     address token;
254     int96 cumulativeGrownStalkPerBDV;
255     for(uint i; i < depositIds.length; i++) {
256         (token, cumulativeGrownStalkPerBDV) =
257             LibBytes.getAddressAndStemFromBytes(
258                 bytes32(depositIds[i])
259             );
260         transferDeposit(
261             sender,
262             recipient,
263             token,
264             cumulativeGrownStalkPerBDV,
265             amounts[i]
266         );
267     }
268 }
```

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:P/S:U (0.0)

Recommendation:

It is recommended to use `++i` instead of `i++` to increment the value of an `uint` variable inside a loop. This also applies to the variables declared inside the `for` loop, not just the iterator. On the other hand, this is not applicable outside of loops.

Remediation Plan:

SOLVED: The Beanstalk team solved the issue with the following commit ID.

Commit ID : 55813422bc515a5e36469108d4c4f835158fa8fd

MANUAL TESTING

MANUAL TESTING

The main goal of the manual testing performed during this audit was to test all the functionalities regarding the Silo V3 section of the overall Beanstalk protocol, focusing on the following points/scenarios:

1. Main changes from the previous Silo version.
2. Testing on the critical protocol functionalities.
3. The short new vesting period for newly issued earned beans.
4. Instant withdraw and no claim functionality.
5. Changes made to integrate deposits as ERC1155 tokens.
6. Unripe Seed rewards functionality.
7. Check all the storage changes and potential storage collisions and general storage issues.
8. Tests focused on ConvertFacet (Converting from BEAN token deposit to CURVE LP token deposit and vice versa depending on if beanstalk is above or below peg and claiming stalk as the BDV of Unripe tokens increases during the Barn Raise).
9. Tests focused on ApprovalFacet (Approving spenders to transfer deposits for owner's deposit and signing off-chain permits).
10. Tests focused on MigrationFacet (Migrating farmer's deposits from old (seasons based) to new silo (stems based) system).
11. Tests focused on LegacyClaimWithdrawalFacet (Claiming pre-existing unclaimed Withdrawals from the Legacy system, as currently new Withdrawals cannot be created anymore).

APPENDIX

Listing 11: Test.t.sol

```
pragma solidity ^0.7.6;
pragma experimental ABIEncoderV2;

import "forge-std/Test.sol";
import "../contracts/beanstalk/Diamond.sol";
import "../contracts/beanstalk/diamond/DiamondCutFacet.sol";
import "../contracts/beanstalk/diamond/DiamondLoupeFacet.sol";
import "../contracts/beanstalk/diamond/OwnershipFacet.sol";
import "../contracts/beanstalk/silo/WhitelistFacet.sol";
import "../contracts/beanstalk/barn/UnripeFacet.sol";
import "../contracts/beanstalk/farm/TokenFacet.sol";
import "../contracts/beanstalk/silo/SiloFacet/SiloFacet.sol";
import "../contracts/beanstalk/sun/SeasonFacet/SeasonFacet.sol";
import "../contracts/beanstalk/diamond/PauseFacet.sol";
import "../contracts/beanstalk/market/MarketplaceFacet/
↳ MarketplaceFacet.sol";
import "../contracts/beanstalk/field/FundraiserFacet.sol";
import "../contracts/beanstalk/field/FieldFacet.sol";
import "../contracts/beanstalk/barn/FertilizerFacet.sol";
import "../contracts/beanstalk/farm/FarmFacet.sol";
import "../contracts/beanstalk/silo/BDVFacet.sol";
import "../contracts/beanstalk/farm/CurveFacet.sol";
import "../contracts/beanstalk/silo/ConvertFacet.sol";
import "../contracts/mocks/mockFacets/MockSeasonFacet.sol";
import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
import "@openzeppelin/contracts/utils/Strings.sol";
import "../contracts/beanstalk/silo/ApprovalFacet.sol";
import "../contracts/beanstalk/silo/MigrationFacet.sol";
import "../contracts/beanstalk/silo/SiloFacet/
↳ LegacyClaimWithdrawalFacet.sol";
import "../contracts/C.sol";

contract BeanstalkEnvironment is Test {
    using Strings for *;

    // CONTRACTS: (https://louper.dev/diamond/0
↳ xC1E088fC1323b20BCBee9bd1B9fC9546db5624C5)
    Diamond public contract_Diamond = Diamond(0
↳ xC1E088fC1323b20BCBee9bd1B9fC9546db5624C5);
    DiamondCutFacet public contract_DiamondCutFacet =
↳ DiamondCutFacet(address(contract_Diamond)); // IMPLEMENTATION: 0
↳ xdfeff7592915bea8d040499e961e332bd453c249
    DiamondLoupeFacet public contract_DiamondLoupeFacet =
```

```
↳ DiamondLoupeFacet(address(contract_Diamond)); // IMPLEMENTATION: 0
↳ xb51d5c699b749e0382e257244610039ddb272da0
38      OwnershipFacet public contract_OwnershipFacet = OwnershipFacet
↳ (address(contract_Diamond)); // IMPLEMENTATION: 0
↳ x5d45283ff53aabdb93693095039b489af8b18cf7
39      WhitelistFacet public contract_WhitelistFacet = WhitelistFacet
↳ (address(contract_Diamond)); // IMPLEMENTATION: 0
↳ xaea0e6e011106968adc7943579c829e49efddad0
40      UnripeFacet public contract_UnripeFacet = UnripeFacet(address(
↳ contract_Diamond)); // IMPLEMENTATION: 0
↳ x261b3ae660504537fbfe15b6c1c664976344eb0a
41      TokenFacet public contract_TokenFacet = TokenFacet(address(
↳ contract_Diamond)); // IMPLEMENTATION: 0
↳ x8d00ef08775872374a327355fe0fdbdece1106cf
42      SiloFacet public contract_SiloFacet = SiloFacet(address(
↳ contract_Diamond)); // IMPLEMENTATION: 0
↳ xf73db3fb33c7070db0f0ae4a76872251dca15e97 & 0
↳ xed7be52f59b4aa0c36b046e5c1f14df62aae79d6
43      SeasonFacet public contract_SeasonFacet = SeasonFacet(address(
↳ contract_Diamond)); // IMPLEMENTATION: 0
↳ x0cEFF1129091A0ffa97cC58d4D160F9676866a24
44      PauseFacet public contract_PauseFacet = PauseFacet(address(
↳ contract_Diamond)); // IMPLEMENTATION: 0
↳ xeab4398f62194948cB25F45fEE4C46Fae2e91229
45      MarketplaceFacet public contract_MarketplaceFacet =
↳ MarketplaceFacet(address(contract_Diamond)); // IMPLEMENTATION: 0
↳ x0c9F436FBEf08914c1C68fe04bD573de6e327776
46      FundraiserFacet public contract_FundraiserFacet =
↳ FundraiserFacet(address(contract_Diamond)); // IMPLEMENTATION: 0
↳ x538C76976eF45b8cA5c12662a86034434bFC7a8E
47      FieldFacet public contract_FieldFacet = FieldFacet(address(
↳ contract_Diamond)); // IMPLEMENTATION: 0
↳ x79801F5cB2592Dd2173482198385e62870a0eAe2
48      FertilizerFacet public contract_FertilizerFacet =
↳ FertilizerFacet(address(contract_Diamond)); // IMPLEMENTATION: 0
↳ xFC7Ed192a24FaB3093c8747c3DDBe6Cacd335B6C
49      FarmFacet public contract_FarmFacet = FarmFacet(address(
↳ contract_Diamond)); // IMPLEMENTATION: 0
↳ x855D37a6C3868Aa4e8F2e1a80965D08B3f10d292
50      BDVFacet public contract_BDVFacet = BDVFacet(address(
↳ contract_Diamond)); // IMPLEMENTATION: 0
↳ xc17ED2e41242063DB6b939f5601bA01374b9D44a
51      CurveFacet public contract_CurveFacet = CurveFacet(address(
↳ contract_Diamond)); // IMPLEMENTATION: 0
```

```
↳ xd231498144c5b53b65b782343CDFB366472c7bf7
52      ConvertFacet public contract_ConvertFacet = ConvertFacet(
↳ address(contract_Diamond)); // IMPLEMENTATION: 0
↳ xc2E90aCbA1DC5ec1B852592390f479012eB304C2
53
54      ERC20 public contract_BEAN = ERC20(0
↳ xBEA0000029AD1c77D3d5D23Ba2D8893dB9d1Efab);
55      ERC20 public contract_USDC = ERC20(0
↳ xA0b86991c6218b36c1d19D4a2e9Eb0cE3606eB48);
56      ERC20 public contract_USDT = ERC20(0
↳ xdAC17F958D2ee523a2206206994597C13D831ec7);
57      ERC20 public contract_WETH = ERC20(0
↳ xC02aaA39b223FE8D0A0e5C4F27eAD9083C756Cc2);
58
59      address public owner = contract_OwnershipFacet.owner();
60      address public bean_holder = 0
↳ x517B37Dc73F756192b318d3853bD4C66ba4b41cB;
61      address public unripeBean_holder = 0
↳ x9F64674CAb93986254c6329C4521c4F9737Af864;
62      address public user1 = 0
↳ x0000000000000000000000000000000000000101;
63      address public user2 = 0
↳ x0000000000000000000000000000000000000102;
64      address public user3 = 0
↳ x0000000000000000000000000000000000000103;
65      address public user4 = 0
↳ x0000000000000000000000000000000000000104;
66
67      address public zero = 0
↳ x0000000000000000000000000000000000000000;
68
69      MockSeasonFacet public mockSeasonFacet;
70      SiloFacet public siloFacet;
71
72      FarmFacet public farmFacet;
73      BDVFacet public bDVFacet;
74      ConvertFacet public convertFacet;
75
76      ApprovalFacet public approvalFacet;
77      MigrationFacet public migrationFacet;
78      LegacyClaimWithdrawalFacet public legacyClaimWithdrawalFacet;
79
80      bytes4[] selectorsSeasonFacet;
81      bytes4[] selectorsMockSeasonFacet;
```

```
82      bytes4[] selectorsSiloFacet;
83      bytes4[] selectorsSiloFacetOld;
84      bytes4[] selectorsFarmFace;
85      bytes4[] selectorsBDVFacet;
86      bytes4[] selectorsConvertFacet;
87      bytes4[] selectorsConvertFacetOld;
88      bytes4[] selectorsApprovalFacet;
89      bytes4[] selectorsMigrationFacet;
90      bytes4[] selectorsLegacyClaimWithdrawalFacet;
91
92      IDiamondCut.FacetCut[] internal cutsSeasonFacet;
93      IDiamondCut.FacetCut[] internal cutsMockSeasonFacet;
94
95      IDiamondCut.FacetCut[] internal cutsSiloFacetOld;
96      IDiamondCut.FacetCut[] internal cutsSiloFacet;
97
98      IDiamondCut.FacetCut[] internal cutsFarmFacet;
99      IDiamondCut.FacetCut[] internal cutsBDVFacet;
100     IDiamondCut.FacetCut[] internal cutsConvertFacetOld;
101     IDiamondCut.FacetCut[] internal cutsConvertFacet;
102
103     IDiamondCut.FacetCut[] internal cutsApprovalFacet;
104     IDiamondCut.FacetCut[] internal cutsMigrationFacet;
105     IDiamondCut.FacetCut[] internal cutsLegacyClaimWithdrawalFacet
 ↳ ;
106
107     int96[] stems;
108     uint256[] amounts;
109
110     function setUp() public {
111
112         vm.startPrank(bean_holder);
113
114         contract_BEAN.transfer(user1, 1_000 * 1e6);
115         contract_BEAN.transfer(user2, 1_000 * 1e6);
116         contract_BEAN.transfer(user3, 1_000 * 1e6);
117         contract_BEAN.transfer(user4, 1_000 * 1e6);
118
119         vm.stopPrank();
120
121         vm.startPrank(unripeBean_holder);
122
123         ERC20(C.UNRIPE_BEAN).transfer(user1, 1_000 * 1e6);
124         ERC20(C.UNRIPE_BEAN).transfer(user2, 1_000 * 1e6);
```

```
125         ERC20(C.UNRIPE_BEAN).transfer(user3, 1_000 * 1e6);
126         ERC20(C.UNRIPE_BEAN).transfer(user4, 1_000 * 1e6);
127
128         vm.stopPrank();
129
130         vm.startPrank(owner);
131
132         siloFacet = new SiloFacet();
133         mockSeasonFacet = new MockSeasonFacet();
134
135         farmFacet = new FarmFacet();
136         bDVFacet = new BDVFacet();
137         convertFacet = new ConvertFacet();
138
139         approvalFacet = new ApprovalFacet();
140         migrationFacet = new MigrationFacet();
141         legacyClaimWithdrawalFacet = new
 ↳ LegacyClaimWithdrawalFacet();
142
143         // ##########
144         // ##########
145         IDiamondCut.FacetCut memory cutSeasonFacet;
146         bytes4[] memory functionSelectorsSeasonFacet =
 ↳ getSelectorsSeasonFacet();
147         cutSeasonFacet = IDiamondCut.FacetCut({
148             facetAddress: zero,
149             action: IDiamondCut.FacetCutAction.Remove,
150             functionSelectors: functionSelectorsSeasonFacet
151         });
152         cutsSeasonFacet.push(cutSeasonFacet);
153         contract_DiamondCutFacet.diamondCut(cutsSeasonFacet, zero,
 ↳  bytes(""));
154         // ##########
155         // ##########
156
157         // ##########
158         // ##########
159         IDiamondCut.FacetCut memory cutMockSeasonFacet;
160         bytes4[] memory functionSelectorsMockSeasonFacet =
 ↳ getSelectorsMockSeasonFacet();
161         cutMockSeasonFacet = IDiamondCut.FacetCut({
162             facetAddress: address(mockSeasonFacet),
163             action: IDiamondCut.FacetCutAction.Add,
164             functionSelectors: functionSelectorsMockSeasonFacet
```

```
165         });
166         cutsMockSeasonFacet.push(cutMockSeasonFacet);
167         contract_DiamondCutFacet.diamondCut(cutsMockSeasonFacet,
 ↳ zero, bytes(""));
168         // ##########
169         // ##########
170 
171         // ##########
172         // ##########
173         IDiamondCut.FacetCut memory cutSiloFacetOld;
174         bytes4[] memory functionSelectorsSiloFacetOld =
 ↳ getSelectorsSiloFacetOld();
175         cutSiloFacetOld = IDiamondCut.FacetCut({
176             facetAddress: zero,
177             action: IDiamondCut.FacetCutAction.Remove,
178             functionSelectors: functionSelectorsSiloFacetOld
179         });
180         cutsSiloFacetOld.push(cutSiloFacetOld);
181         contract_DiamondCutFacet.diamondCut(cutsSiloFacetOld, zero
 ↳ , bytes(""));
182         // ##########
183         // ##########
184 
185         // ##########
186         // ##########
187         IDiamondCut.FacetCut memory cutSiloFacet;
188         bytes4[] memory functionSelectorsSiloFacet =
 ↳ getSelectorsSiloFacet();
189         cutSiloFacet = IDiamondCut.FacetCut({
190             facetAddress: address(siloFacet),
191             action: IDiamondCut.FacetCutAction.Add,
192             functionSelectors: functionSelectorsSiloFacet
193         });
194         cutsSiloFacet.push(cutSiloFacet);
195         contract_DiamondCutFacet.diamondCut(cutsSiloFacet, zero,
 ↳ bytes(""));
196         // ##########
197         // ##########
198 
199         // ##########
200         // ##########
201         IDiamondCut.FacetCut memory cutFarmFacet;
202         bytes4[] memory functionSelectorsFarmFacet =
 ↳ getSelectorsFarmFacet();
```

```
203         cutFarmFacet = IDiamondCut.FacetCut({
204             facetAddress: address(farmFacet),
205             action: IDiamondCut.FacetCutAction.Replace,
206             functionSelectors: functionSelectorsFarmFacet
207         });
208         cutsFarmFacet.push(cutFarmFacet);
209         contract_DiamondCutFacet.diamondCut(cutsFarmFacet, zero,
 ↳ bytes(""));
210         // ##########
211         // ##########
212 
213         // ##########
214         // ##########
215         IDiamondCut.FacetCut memory cutBDVFacet;
216         bytes4[] memory functionSelectorsBDVFacet =
 ↳ getSelectorsBDVFacet();
217         cutBDVFacet = IDiamondCut.FacetCut({
218             facetAddress: address(bDVFacet),
219             action: IDiamondCut.FacetCutAction.Replace,
220             functionSelectors: functionSelectorsBDVFacet
221         });
222         cutsBDVFacet.push(cutBDVFacet);
223         contract_DiamondCutFacet.diamondCut(cutsBDVFacet, zero,
 ↳ bytes(""));
224         // ##########
225         // ##########
226 
227         // ##########
228         // ##########
229         IDiamondCut.FacetCut memory cutConvertFacetOld;
230         bytes4[] memory functionSelectorsConvertFacetOld =
 ↳ getSelectorsConvertFacetOld();
231         cutConvertFacetOld = IDiamondCut.FacetCut({
232             facetAddress: zero,
233             action: IDiamondCut.FacetCutAction.Remove,
234             functionSelectors: functionSelectorsConvertFacetOld
235         });
236         cutsConvertFacetOld.push(cutConvertFacetOld);
237         contract_DiamondCutFacet.diamondCut(cutsConvertFacetOld,
 ↳ zero, bytes(""));
238         // ##########
239         // ##########
240 
241         // ##########
```

```
242         // ##########
243         IDiamondCut.FacetCut memory cutConvertFacet;
244         bytes4[] memory functionSelectorsConvertFacet =
 ↳ getSelectorsConvertFacet();
245         cutConvertFacet = IDiamondCut.FacetCut({
246             facetAddress: address(convertFacet),
247             action: IDiamondCut.FacetCutAction.Add,
248             functionSelectors: functionSelectorsConvertFacet
249         });
250         cutsConvertFacet.push(cutConvertFacet);
251         contract_DiamondCutFacet.diamondCut(cutsConvertFacet, zero
 ↳ , bytes(""));
252         // ##########
253         // ##########
254
255         // ##########
256         // ##########
257         IDiamondCut.FacetCut memory cutApprovalFacet;
258         bytes4[] memory functionSelectorsApprovalFacet =
 ↳ getSelectorsApprovalFacet();
259         cutApprovalFacet = IDiamondCut.FacetCut({
260             facetAddress: address(approvalFacet),
261             action: IDiamondCut.FacetCutAction.Add,
262             functionSelectors: functionSelectorsApprovalFacet
263         });
264         cutsApprovalFacet.push(cutApprovalFacet);
265         contract_DiamondCutFacet.diamondCut(cutsApprovalFacet,
 ↳ zero, bytes(""));
266         // ##########
267         // ##########
268
269         // ##########
270         // ##########
271         IDiamondCut.FacetCut memory cutMigrationFacet;
272         bytes4[] memory functionSelectorsMigrationFacet =
 ↳ getSelectorsMigrationFacet();
273         cutMigrationFacet = IDiamondCut.FacetCut({
274             facetAddress: address(migrationFacet),
275             action: IDiamondCut.FacetCutAction.Add,
276             functionSelectors: functionSelectorsMigrationFacet
277         });
278         cutsMigrationFacet.push(cutMigrationFacet);
279         contract_DiamondCutFacet.diamondCut(cutsMigrationFacet,
 ↳ zero, bytes(""));
```

```
280         // ##########
281         // ##########
282
283         // ##########
284         // ###########
285         IDiamondCut.FacetCut memory cutLegacyClaimWithdrawalFacet;
286         bytes4[] memory
 ↳ functionSelectorsLegacyClaimWithdrawalFacet =
 ↳ getSelectorsLegacyClaimWithdrawalFacet();
287         cutLegacyClaimWithdrawalFacet = IDiamondCut.FacetCut({
288             facetAddress: address(legacyClaimWithdrawalFacet),
289             action: IDiamondCut.FacetCutAction.Add,
290             functionSelectors:
 ↳ functionSelectorsLegacyClaimWithdrawalFacet
291         });
292         cutsLegacyClaimWithdrawalFacet.push(
 ↳ cutLegacyClaimWithdrawalFacet);
293         contract_DiamondCutFacet.diamondCut(
 ↳ cutsLegacyClaimWithdrawalFacet, zero, bytes(""));
294         // ##########
295         // ##########
296
297         siloFacet = SiloFacet(address(contract_Diamond));
298         mockSeasonFacet = MockSeasonFacet(address(contract_Diamond
 ↳ ));
299         farmFacet = FarmFacet(address(contract_Diamond));
300         bDVFacet = BDVFacet(address(contract_Diamond));
301         convertFacet = ConvertFacet(address(contract_Diamond));
302         approvalFacet = ApprovalFacet(address(contract_Diamond));
303         migrationFacet = MigrationFacet(address(contract_Diamond))
 ↳ ;
304         legacyClaimWithdrawalFacet = LegacyClaimWithdrawalFacet(
 ↳ address(contract_Diamond));
305
306         mockSeasonFacet.deployStemsUpgrade();
307
308         vm.stopPrank();
309     }
310
311     function test_deposit() public {
312         vm.startPrank(user1);
313
314         contract_BEAN.approve(address(siloFacet), 100 * 1e6);
315         siloFacet.deposit(address(contract_BEAN), 100 * 1e6,
```

```
 ↳ LibTransfer.From.EXTERNAL);
316
317         vm.stopPrank();
318     }
319
320     function test_deposit1() public {
321         vm.startPrank(user1);
322
323         _sunrise(100 * 1e6);
324
325         uint256 bdv;
326         int96 stem;
327
328         contract_BEAN.approve(address(siloFacet), 100 * 1e6);
329         (, bdv, stem) = siloFacet.deposit(address(contract_BEAN),
 ↳ 100 * 1e6, LibTransfer.From.EXTERNAL);
330
331         console.log("bdv--> ", bdv, " //stem --> ", uint256(stem))
 ↳ ;
332
333         _sunrise(100 * 1e6);
334
335         vm.stopPrank();
336     }
337
338     function test_withdrawDeposit() public {
339         vm.startPrank(user1);
340
341         mockSeasonFacet.setAbovePegE(true);
342         _sunrise(100 * 1e6);
343
344         uint256 bdv;
345         int96 stem;
346
347         console.log("contract_BEAN.balanceOf(user1) --> ",
 ↳ contract_BEAN.balanceOf(user1));
348
349         contract_BEAN.approve(address(siloFacet), 100 * 1e6);
350         (, bdv, stem) = siloFacet.deposit(address(contract_BEAN),
 ↳ 100 * 1e6, LibTransfer.From.EXTERNAL);
351
352         console.log("contract_BEAN.balanceOf(user1) --> ",
 ↳ contract_BEAN.balanceOf(user1));
353
```

```
354         console.log("bdv--> ", bdv, " //stem --> ", uint256(stem))
 ↳ ;
355
356         _sunrise(100 * 1e6);
357
358         siloFacet.withdrawDeposit(address(contract_BEAN), 0, 100 *
 ↳  1e6, LibTransfer.To.EXTERNAL);
359
360         console.log("contract_BEAN.balanceOf(user1) --> ",
 ↳ contract_BEAN.balanceOf(user1));
361
362         vm.stopPrank();
363     }
364
365     function test_withdrawDeposit1() public {
366         vm.startPrank(user1);
367
368         mockSeasonFacet.setAbovePegE(true);
369         _sunrise(100 * 1e6);
370
371         uint256 bdv;
372         int96 stem;
373
374         console.log("contract_BEAN.balanceOf(user1) --> ",
 ↳ contract_BEAN.balanceOf(user1));
375
376         contract_BEAN.approve(address(siloFacet), 100 * 1e6);
377         (, bdv, stem) = siloFacet.deposit(address(contract_BEAN),
 ↳ 100 * 1e6, LibTransfer.From.EXTERNAL);
378
379         console.log("contract_BEAN.balanceOf(user1) --> ",
 ↳ contract_BEAN.balanceOf(user1));
380
381         console.log("bdv--> ", bdv, " //stem --> ", uint256(stem))
 ↳ ;
382
383         _sunrise(100 * 1e6, 10);
384
385         siloFacet.withdrawDeposit(address(contract_BEAN), 0, 100 *
 ↳  1e6, LibTransfer.To.EXTERNAL);
386
387         console.log("contract_BEAN.balanceOf(user1) --> ",
 ↳ contract_BEAN.balanceOf(user1));
388
```

```
389         vm.stopPrank();
390     }
391
392     function test_withdrawDeposit2() public {
393         vm.startPrank(user1);
394
395         mockSeasonFacet.setAbovePegE(true);
396         _sunrise(100 * 1e6);
397
398         uint256 bdv;
399         int96 stem;
400
401         console.log("contract_BEAN.balanceOf(user1) --> ",
 ↳ contract_BEAN.balanceOf(user1));
402
403         contract_BEAN.approve(address(siloFacet), 100 * 1e6);
404         (, bdv, stem) = siloFacet.deposit(address(contract_BEAN),
 ↳ 100 * 1e6, LibTransfer.From.EXTERNAL);
405
406         console.log("contract_BEAN.balanceOf(user1) --> ",
 ↳ contract_BEAN.balanceOf(user1));
407
408         console.log("bdv--> ", bdv, " //stem --> ", uint256(stem))
 ↳ ;
409
410         _sunrise(100 * 1e6, 10);
411
412         siloFacet.plant(address(contract_BEAN));
413
414         siloFacet.withdrawDeposit(address(contract_BEAN), 0, 100 *
 ↳  1e6, LibTransfer.To.EXTERNAL);
415
416         console.log("contract_BEAN.balanceOf(user1) --> ",
 ↳ contract_BEAN.balanceOf(user1));
417
418         vm.stopPrank();
419     }
420
421     function test_withdrawDeposit30() public {
422         vm.startPrank(user1);
423
424         mockSeasonFacet.setAbovePegE(true);
425         _sunrise(100 * 1e6);
426
```

```
427         uint256 bdv;
428         int96 stem;
429
430         contract_BEAN.approve(address(siloFacet), 100 * 1e6);
431         (, bdv, stem) = siloFacet.deposit(address(contract_BEAN),
 ↳ 100 * 1e6, LibTransfer.From.EXTERNAL);
432
433         console.log("bdv--> ", bdv, " //stem --> ", uint256(stem))
 ↳ ;
434
435         _sunrise(100 * 1e6, 10);
436
437         console.log("--------- BEFORE PLANT -------");
438         accountInfo(user1);
439         siloFacet.plant(address(contract_BEAN));
440         console.log("--------- AFTER PLANT -------");
441         accountInfo(user1);
442
443
444         siloFacet.withdrawDeposit(address(contract_BEAN), 0, 100 *
 ↳  1e6, LibTransfer.To.EXTERNAL);
445
446         console.log("--------- AFTER WITHDRAWAL -------");
447         accountInfo(user1);
448
449         vm.stopPrank();
450     }
```

Listing 12: Poc.t.sol

```
452 function test_withdrawDeposit31() public {
453         vm.startPrank(user1);
454
455         mockSeasonFacet.setAbovePegE(true);
456         _sunrise(1000 * 1e6);
457
458         uint256 bdv;
459         int96 stem;
460
461         contract_BEAN.approve(address(siloFacet), 100 * 1e6);
462         (, bdv, stem) = siloFacet.deposit(address(contract_BEAN),
 ↳ 100 * 1e6, LibTransfer.From.EXTERNAL);
463
464         console.log("bdv--> ", bdv, " //stem --> ", uint256(stem))
```

```
 ↳ ;
465
466         _sunrise(1000 * 1e6, 100);
467
468         console.log("--------- BEFORE PLANT -------");
469         accountInfo(user1);
470         siloFacet.plant(address(contract_BEAN));
471
472         console.log("--------- AFTER  PLANT , BEFORE CLAIMPLENTY
 ↳ -------");
473         accountInfo(user1);
474
475         siloFacet.claimPlenty();
476         console.log("--------- AFTER  CLAIMPLENTY -------");
477         accountInfo(user1);
478
479         siloFacet.withdrawDeposit(address(contract_BEAN), 0, 100 *
 ↳  1e6, LibTransfer.To.EXTERNAL);
480
481         console.log("--------- AFTER WITHDRAWAL -------");
482         accountInfo(user1);
483
484         vm.stopPrank();
485     }
486
487     function test_withdrawDeposit32() public {
488         vm.startPrank(user1);
489
490         mockSeasonFacet.setAbovePegE(true);
491         _sunrise(1000 * 1e6);
492
493         uint256 bdv;
494         int96 stem;
495
496         contract_BEAN.approve(address(siloFacet), 100 * 1e6);
497         (, bdv, stem) = siloFacet.deposit(address(contract_BEAN),
 ↳ 100 * 1e6, LibTransfer.From.EXTERNAL);
498
499         console.log("bdv--> ", bdv, " //stem --> ", uint256(stem))
 ↳ ;
500
501         _sunrise(1000 * 1e6, 100);
502
503         console.log("--------- BEFORE CLAIMPLENTY -------");
```

```
504         accountInfo(user1);
505         siloFacet.claimPlenty();
506 
507         console.log("--------- AFTER CLAIMPLENTY , BEFORE PLANT
 ↳ -------");
508         accountInfo(user1);
509         siloFacet.plant(address(contract_BEAN));
510         console.log("--------- AFTER PLANT -------");
511         accountInfo(user1);
512 
513         siloFacet.withdrawDeposit(address(contract_BEAN), 0, 100 *
 ↳  1e6, LibTransfer.To.EXTERNAL);
514         console.log("--------- AFTER WITHDRAWAL -------");
515 
516         accountInfo(user1);
517 
518         vm.stopPrank();
519     }
520 
521     function test_withdrawDeposit33() public {
522         vm.startPrank(user1);
523 
524         mockSeasonFacet.setAbovePegE(true);
525         _sunrise(1000 * 1e6);
526 
527         uint256 bdv;
528         int96 stem;
529 
530         contract_BEAN.approve(address(siloFacet), 100 * 1e6);
531         (, bdv, stem) = siloFacet.deposit(address(contract_BEAN),
 ↳ 100 * 1e6, LibTransfer.From.EXTERNAL);
532 
533         console.log("bdv--> ", bdv, " //stem --> ", uint256(stem))
 ↳ ;
534 
535         _sunrise(1000 * 1e6, 100);
536 
537         console.log("--------- BEFORE CLAIM PLENTY -------");
538         accountInfo(user1);
539         siloFacet.claimPlenty();
540         console.log("--------- AFTER CLAIM PLENTY -------");
541         accountInfo(user1);
542 
543         // siloFacet.plant(address(contract_BEAN));
```

```
544
545         siloFacet.withdrawDeposit(address(contract_BEAN), 0, 100 *
 ↳  1e6, LibTransfer.To.EXTERNAL);
546
547         console.log("--------- AFTER WITHDRAWAL -------");
548         accountInfo(user1);
549
550         vm.stopPrank();
551     }
552
553     function test_withdrawDeposit34() public {
554         vm.startPrank(user1);
555
556         mockSeasonFacet.setAbovePegE(true);
557         _sunrise(1000 * 1e6);
558
559         uint256 bdv;
560         int96 stem;
561
562         contract_BEAN.approve(address(siloFacet), 100 * 1e6);
563         (, bdv, stem) = siloFacet.deposit(address(contract_BEAN),
 ↳ 100 * 1e6, LibTransfer.From.EXTERNAL);
564
565         console.log("bdv--> ", bdv, " //stem --> ", uint256(stem))
 ↳ ;
566
567         _sunrise(1000 * 1e6, 100);
568
569         console.log("--------- BEFORE CLAIMPLENTY -------");
570         accountInfo(user1);
571         siloFacet.claimPlenty();
572
573         console.log("--------- AFTER CLAIMPLENTY , BEFORE PLANT
 ↳ -------");
574         accountInfo(user1);
575         uint256 beans = siloFacet.plant(address(contract_BEAN));
576         console.log("--------- AFTER PLANT -------");
577         accountInfo(user1);
578
579         siloFacet.withdrawDeposit(address(contract_BEAN), stem,
 ↳ 100 * 1e6, LibTransfer.To.EXTERNAL);
580         console.log("--------- AFTER WITHDRAWAL -------");
581
582         accountInfo(user1);
```

```
583
584         // siloFacet.withdrawDeposit(address(contract_BEAN), int96
 ↳ (siloFacet.stemTipForToken(address(contract_BEAN))), beans,
 ↳ LibTransfer.To.EXTERNAL);
585         // console.log("--------- AFTER SECOND WITHDRAWAL
 ↳ -------");
586
587         // accountInfo(user1);
588
589         vm.stopPrank();
590     }
591
592     function test_withdrawDeposit35() public {
593         vm.startPrank(user1);
594
595         mockSeasonFacet.setAbovePegE(true);
596         _sunrise(1000 * 1e6);
597
598         uint256 bdv;
599         int96 stem;
600
601         contract_BEAN.approve(address(siloFacet), 100 * 1e6);
602         (, bdv, stem) = siloFacet.deposit(address(contract_BEAN),
 ↳ 100 * 1e6, LibTransfer.From.EXTERNAL);
603
604         console.log("bdv--> ", bdv, " //stem --> ", uint256(stem))
 ↳ ;
605
606         _sunrise(1000 * 1e6, 100);
607
608         console.log("--------- BEFORE CLAIMPLENTY -------");
609         accountInfo(user1);
610         siloFacet.claimPlenty();
611
612         console.log("--------- AFTER CLAIMPLENTY , BEFORE PLANT
 ↳ -------");
613         accountInfo(user1);
614         uint256 beans = siloFacet.plant(address(contract_BEAN));
615         console.log("--------- AFTER PLANT -------");
616         accountInfo(user1);
617
618         siloFacet.withdrawDeposit(address(contract_BEAN), stem,
 ↳ 100 * 1e6, LibTransfer.To.EXTERNAL);
619         console.log("--------- AFTER WITHDRAWAL -------");
```

```
620
621         accountInfo(user1);
622
623         _sunrise(1000 * 1e6, 100);
624
625         console.log("--------- AFTER 100 SUNRISES -------");
626
627         accountInfo(user1);
628
629         // siloFacet.withdrawDeposit(address(contract_BEAN), int96
 ↳ (siloFacet.stemTipForToken(address(contract_BEAN))), beans,
 ↳ LibTransfer.To.EXTERNAL);
630         // console.log("--------- AFTER SECOND WITHDRAWAL
 ↳ -------");
631
632         // accountInfo(user1);
633
634         vm.stopPrank();
635     }
636
637     function test_withdrawDeposit36() public {
638         vm.startPrank(user1);
639
640         mockSeasonFacet.setAbovePegE(true);
641         _sunrise(1000 * 1e6);
642
643         uint256 bdv;
644         int96 stem;
645
646         contract_BEAN.approve(address(siloFacet), 100 * 1e6);
647         (, bdv, stem) = siloFacet.deposit(address(contract_BEAN),
 ↳ 100 * 1e6, LibTransfer.From.EXTERNAL);
648
649         console.log("bdv--> ", bdv, " //stem --> ", uint256(stem))
 ↳ ;
650
651         _sunrise(1000 * 1e6, 100);
652
653         console.log("--------- BEFORE MOW -------");
654         accountInfo(user1);
655         siloFacet.mow(user1, address(contract_BEAN));
656
657         console.log("--------- AFTER MOW , BEFORE PLANT -------");
658         accountInfo(user1);
```

```
659         uint256 beans = siloFacet.plant(address(contract_BEAN));
660         console.log("--------- AFTER PLANT -------");
661         accountInfo(user1);
662 
663         siloFacet.withdrawDeposit(address(contract_BEAN), stem,
 ↳ 100 * 1e6, LibTransfer.To.EXTERNAL);
664         console.log("--------- AFTER WITHDRAWAL -------");
665 
666         accountInfo(user1);
667 
668         _sunrise(1000 * 1e6, 100);
669 
670         console.log("--------- AFTER 100 SUNRISES -------");
671 
672         accountInfo(user1);
673 
674         // siloFacet.withdrawDeposit(address(contract_BEAN), int96
 ↳ (siloFacet.stemTipForToken(address(contract_BEAN))), beans,
 ↳ LibTransfer.To.EXTERNAL);
675         // console.log("--------- AFTER SECOND WITHDRAWAL
 ↳ -------");
676 
677         // accountInfo(user1);
678 
679         vm.stopPrank();
680     }
681 
682     function test_withdrawDeposit37() public {
683         vm.startPrank(user1);
684 
685         mockSeasonFacet.setAbovePegE(true);
686 
687         console.log("siloFacet.stemTipForToken(address(
 ↳ contract_BEAN)) --> ",uint128(siloFacet.stemTipForToken(address(
 ↳ contract_BEAN))));
688         _sunrise(1000 * 1e6);
689 
690         uint256 bdv;
691         int96 stem;
692 
693         contract_BEAN.approve(address(siloFacet), 100 * 1e6);
694         (, bdv, stem) = siloFacet.deposit(address(contract_BEAN),
 ↳ 100 * 1e6, LibTransfer.From.EXTERNAL);
695         console.log("bdv--> ", bdv, " //stem --> ", uint256(stem))
```

```
 ↳ ;
696
697         _sunrise(1000 * 1e6, 100);
698
699         console.log("--------- BEFORE PLANT -------");
700         accountInfo(user1);
701         uint256 beans = siloFacet.plant(address(contract_BEAN));
702         console.log("--------- AFTER PLANT -------");
703         _sunrise(1000 * 1e6, 100);
704
705         console.log("--------- AND AFTER 100 SUNRISES -------");
706
707         accountInfo(user1);
708
709         console.log("siloFacet.stemTipForToken(address(
 ↳ contract_BEAN)) --> ", uint128(siloFacet.stemTipForToken(address(
 ↳ contract_BEAN))));
710
711         siloFacet.mow(user1, address(contract_BEAN));
712         console.log("--------- AFTER MOW -------");
713         accountInfo(user1);
714
715         siloFacet.withdrawDeposit(address(contract_BEAN), stem,
 ↳ 100 * 1e6, LibTransfer.To.EXTERNAL);
716         console.log("--------- AFTER WITHDRAWAL -------");
717
718         accountInfo(user1);
719
720         // _sunrise(1000 * 1e6, 100);
721
722         // console.log("--------- AFTER 100 SUNRISES -------");
723
724         // accountInfo(user1);
725
726         // siloFacet.withdrawDeposit(address(contract_BEAN), int96
 ↳ (siloFacet.stemTipForToken(address(contract_BEAN))), beans,
 ↳ LibTransfer.To.EXTERNAL);
727         // console.log("--------- AFTER SECOND WITHDRAWAL
 ↳ -------");
728
729         // accountInfo(user1);
730
731         vm.stopPrank();
732     }
```

```
733
734     function test_transferDeposit1() public {
735         vm.startPrank(user1);
736
737         mockSeasonFacet.setAbovePegE(true);
738         _sunrise(1000 * 1e6);
739
740         uint256 bdv;
741         int96 stem;
742
743         contract_BEAN.approve(address(siloFacet), 100 * 1e6);
744         (, bdv, stem) = siloFacet.deposit(address(contract_BEAN),
 ↳ 100 * 1e6, LibTransfer.From.EXTERNAL);
745
746         console.log("bdv--> ", bdv, " //stem --> ", uint256(stem))
 ↳ ;
747
748         _sunrise(1000 * 1e6, 100);
749
750         console.log("--------- BEFORE TRANSFER -------");
751         accountInfo(user1);
752         console.log("-------------------------------------");
753         accountInfo(user2);
754
755         vm.stopPrank();
756
757         vm.startPrank(user2);
758
759         siloFacet.transferDeposit(user1, user2, address(
 ↳ contract_BEAN), stem, 100 * 1e6);
760
761         console.log("--------- AFTER TRANSFER -------");
762         accountInfo(user1);
763         console.log("-------------------------------------");
764         accountInfo(user2);
765
766         vm.stopPrank();
767     }
768
769     function test_transferDeposit2() public {
770         vm.startPrank(user1);
771
772         mockSeasonFacet.setAbovePegE(true);
773         _sunrise(1000 * 1e6);
```

```
774
775         uint256 bdv;
776         int96 stem;
777
778         contract_BEAN.approve(address(siloFacet), 100 * 1e6);
779         (, bdv, stem) = siloFacet.deposit(address(contract_BEAN),
 ↳ 100 * 1e6, LibTransfer.From.EXTERNAL);
780
781         console.log("bdv--> ", bdv, " //stem --> ", uint256(stem))
 ↳ ;
782
783         _sunrise(1000 * 1e6, 100);
784
785         console.log("--------- BEFORE TRANSFER -------");
786         accountInfo(user1);
787         console.log("-------------------------------------");
788         accountInfo(user2);
789
790         approvalFacet.approveDeposit(user2, address(contract_BEAN)
 ↳ , 100 * 1e6);
791
792         vm.stopPrank();
793
794         vm.startPrank(user2);
795
796         siloFacet.transferDeposit(user1, user2, address(
 ↳ contract_BEAN), stem, 100 * 1e6);
797
798         console.log("--------- AFTER TRANSFER -------");
799         accountInfo(user1);
800         console.log("-------------------------------------");
801         accountInfo(user2);
802
803         vm.stopPrank();
804     }
805
806     function test_convertDeposit1() public {
807         vm.startPrank(user1);
808
809         mockSeasonFacet.setAbovePegE(false);
810         _sunrise(1000 * 1e6);
811
812         uint256 bdv;
813         int96 stem;
```

```
814
815         contract_BEAN.approve(address(siloFacet), 100 * 1e6);
816         (, bdv, stem) = siloFacet.deposit(address(contract_BEAN),
 ↳ 100 * 1e6, LibTransfer.From.EXTERNAL);
817
818         console.log("bdv--> ", bdv, " //stem --> ", uint256(stem))
 ↳ ;
819
820         _sunrise(1000 * 1e6, 100);
821
822         console.log("--------- BEFORE CONVERT -------");
823         accountInfo(user1);
824
825         stems.push(stem);
826
827         amounts.push(100 * 1e6);
828
829         convertFacet.convert(abi.encode(LibConvertData.ConvertKind
 ↳ .BEANS_TO_CURVE_LP, 100 * 1e6, 0, C.curveMetapoolAddress()), stems
 ↳ , amounts);
830
831         // siloFacet.transferDeposit(user1, user2, address(
 ↳ contract_BEAN), int96(siloFacet.stemTipForToken(address(
 ↳ contract_BEAN))), 100 * 1e6);
832
833         console.log("--------- AFTER CONVERT -------");
834         accountInfo(user1);
835
836         vm.stopPrank();
837     }
838
839     function test_enrootDeposit1() public {
840         vm.startPrank(user1);
841
842         mockSeasonFacet.setAbovePegE(true);
843         _sunrise(1000 * 1e6);
844
845         uint256 bdv;
846         int96 stem;
847
848         // ERC20(C.UNRIPE_BEAN).approve(address(siloFacet), 100 *
 ↳ 1e6);
849         // (, bdv, stem) = siloFacet.deposit(C.UNRIPE_BEAN, 100 *
 ↳ 1e6, LibTransfer.From.EXTERNAL);
```

```
850
851         // console.log("bdv--> ", bdv, " //stem --> ", uint256(
 ↳ stem));
852
853         _sunrise(1000 * 1e6, 100);
854
855         console.log("--------- BEFORE enrootDeposit -------");
856         accountInfo(user1);
857
858         convertFacet.enrootDeposit(C.UNRIPE_BEAN, int96(siloFacet.
 ↳ stemTipForToken(C.UNRIPE_BEAN)), 100 * 1e6);
859
860         console.log("--------- AFTER enrootDeposit -------");
861         accountInfo(user1);
862
863         vm.stopPrank();
864     }
865
866     function test_X_sunriseCheck0() public {
867         vm.startPrank(user1);
868
869         contract_BEAN.approve(address(siloFacet), 100e6);
870         siloFacet.deposit(address(contract_BEAN), 100e6,
 ↳ LibTransfer.From.EXTERNAL);
871
872         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
873         mockSeasonFacet.siloSunrise(1000);
874
875         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
876         mockSeasonFacet.siloSunrise(1000);
877
878         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
879         mockSeasonFacet.siloSunrise(1000);
880
881         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
882         mockSeasonFacet.siloSunrise(1000);
883
884         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
885     }
```

```
886
887     function test_X_sunriseCheck1() public {
888         vm.startPrank(user1);
889
890         contract_BEAN.approve(address(siloFacet), 100e6);
891         siloFacet.deposit(address(contract_BEAN), 100e6,
 ↳ LibTransfer.From.EXTERNAL);
892
893         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
894         mockSeasonFacet.siloSunrise(1000);
895         vm.warp(block.timestamp + 3600);
896
897         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
898         mockSeasonFacet.siloSunrise(1000);
899         vm.warp(block.timestamp + 3600);
900
901         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
902         mockSeasonFacet.siloSunrise(1000);
903         vm.warp(block.timestamp + 3600);
904
905         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
906         mockSeasonFacet.siloSunrise(1000);
907         vm.warp(block.timestamp + 3600);
908
909         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
910     }
911
912
913     function test_X_sunriseCheck2() public {
914         vm.startPrank(user1);
915
916         contract_BEAN.approve(address(siloFacet), 100e6);
917         siloFacet.deposit(address(contract_BEAN), 100e6,
 ↳ LibTransfer.From.EXTERNAL);
918
919         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
920         vm.roll(block.number + 25);
921         vm.warp(block.timestamp + 3600);
```

```
922         mockSeasonFacet.siloSunrise(1000);
923
924         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
925         vm.roll(block.number + 25);
926         vm.warp(block.timestamp + 3600);
927         mockSeasonFacet.siloSunrise(1000);
928
929         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
930         vm.roll(block.number + 25);
931         vm.warp(block.timestamp + 3600);
932         mockSeasonFacet.siloSunrise(1000);
933
934         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
935         vm.roll(block.number + 25);
936         vm.warp(block.timestamp + 3600);
937         mockSeasonFacet.siloSunrise(1000);
938
939         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
940     }
941
942     function test_X_sunriseCheck3() public {
943         vm.startPrank(user1);
944
945         contract_BEAN.approve(address(siloFacet), 100e6);
946         siloFacet.deposit(address(contract_BEAN), 100e6,
 ↳ LibTransfer.From.EXTERNAL);
947
948         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
949         mockSeasonFacet.siloSunrise(1000);
950         vm.roll(block.number + 25);
951         vm.warp(block.timestamp + 3600);
952
953         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
954         mockSeasonFacet.siloSunrise(1000);
955         vm.roll(block.number + 25);
956         vm.warp(block.timestamp + 3600);
957
958         console.log("USER1 GROWN STALK  ----> ", siloFacet.
```

```
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
959         mockSeasonFacet.siloSunrise(1000);
960         vm.roll(block.number + 25);
961         vm.warp(block.timestamp + 3600);
962
963         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
964         mockSeasonFacet.siloSunrise(1000);
965         vm.roll(block.number + 25);
966         vm.warp(block.timestamp + 3600);
967
968         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
969     }
```

Listing 13: Poc.t.sol

```
971 function test_X_sunriseCheck4() public {
972         vm.startPrank(user1);
973
974         contract_BEAN.approve(address(siloFacet), 100e6);
975         siloFacet.deposit(address(contract_BEAN), 100e6,
 ↳ LibTransfer.From.EXTERNAL);
976
977         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
978         mockSeasonFacet.lightSunrise();
979
980         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
981         mockSeasonFacet.lightSunrise();
982
983         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
984         mockSeasonFacet.lightSunrise();
985
986         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
987         mockSeasonFacet.lightSunrise();
988
989         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
990     }
991
```

```
992     function test_X_deposit() public {
993         vm.startPrank(user1);
994 
995         contract_BEAN.approve(address(siloFacet), 100e6);
996         siloFacet.deposit(address(contract_BEAN), 100e6,
 ↳ LibTransfer.From.EXTERNAL);
997 
998         console.log("DECIMALS            ---->  X000000000000");
999         console.log("USER 1 STALK        ----> ", siloFacet.
 ↳ balanceOfStalk(user1));
1000     }
1001 
1002         function test_X_deposit05() public {
1003         vm.startPrank(user1);
1004 
1005         contract_BEAN.approve(address(siloFacet), 100e6);
1006         siloFacet.deposit(address(contract_BEAN), 100e6,
 ↳ LibTransfer.From.EXTERNAL);
1007 
1008         vm.warp(block.timestamp + 3600);
1009         vm.roll(block.number + 25);
1010         mockSeasonFacet.lightSunrise();
1011 
1012         vm.warp(block.timestamp + 3600);
1013         vm.roll(block.number + 25);
1014         mockSeasonFacet.lightSunrise();
1015 
1016         vm.warp(block.timestamp + 3600);
1017         vm.roll(block.number + 25);
1018         mockSeasonFacet.lightSunrise();
1019 
1020         vm.warp(block.timestamp + 3600);
1021         vm.roll(block.number + 25);
1022         mockSeasonFacet.lightSunrise();
1023 
1024         vm.warp(block.timestamp + 3600);
1025         vm.roll(block.number + 25);
1026         mockSeasonFacet.lightSunrise();
1027 
1028         vm.warp(block.timestamp + 3600);
1029         vm.roll(block.number + 25);
1030         mockSeasonFacet.lightSunrise();
1031 
1032         vm.warp(block.timestamp + 3600);
```

```
033         vm.roll(block.number + 25);
034         mockSeasonFacet.lightSunrise();
035 
036         vm.warp(block.timestamp + 3600);
037         vm.roll(block.number + 25);
038         mockSeasonFacet.lightSunrise();
039 
040         console.log("DECIMALS           ---->  X000000000000");
041         console.log("USER1 STALK        ----> ", siloFacet.
 ↳ balanceOfStalk(user1));
042         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
043     }
044 
045     function test_X_deposit06() public {
046         vm.startPrank(user1);
047 
048         contract_BEAN.approve(address(siloFacet), 100e6);
049         siloFacet.deposit(address(contract_BEAN), 100e6,
 ↳ LibTransfer.From.EXTERNAL);
050 
051         mockSeasonFacet.lightSunrise();
052         vm.warp(block.timestamp + 3600);
053         vm.roll(block.number + 25);
054 
055         mockSeasonFacet.lightSunrise();
056         vm.warp(block.timestamp + 3600);
057         vm.roll(block.number + 25);
058 
059         mockSeasonFacet.lightSunrise();
060         vm.warp(block.timestamp + 3600);
061         vm.roll(block.number + 25);
062 
063         mockSeasonFacet.lightSunrise();
064         vm.warp(block.timestamp + 3600);
065         vm.roll(block.number + 25);
066 
067         mockSeasonFacet.lightSunrise();
068         vm.warp(block.timestamp + 3600);
069         vm.roll(block.number + 25);
070 
071         mockSeasonFacet.lightSunrise();
072         vm.warp(block.timestamp + 3600);
073         vm.roll(block.number + 25);
```

```
074
075         mockSeasonFacet.lightSunrise();
076         vm.warp(block.timestamp + 3600);
077         vm.roll(block.number + 25);
078
079         mockSeasonFacet.lightSunrise();
080         vm.warp(block.timestamp + 3600);
081         vm.roll(block.number + 25);
082
083         mockSeasonFacet.lightSunrise();
084         vm.warp(block.timestamp + 3600);
085         vm.roll(block.number + 25);
086
087         console.log("DECIMALS           ---->  X000000000000");
088         console.log("USER1 STALK        ----> ", siloFacet.
 ↳ balanceOfStalk(user1));
089         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
090     }
091
092
093     function test_X_withdrawDeposit01() public {
094         vm.startPrank(user1);
095
096         contract_BEAN.approve(address(siloFacet), 100e6);
097         siloFacet.deposit(address(contract_BEAN), 100e6,
 ↳ LibTransfer.From.EXTERNAL);
098         siloFacet.withdrawDeposit(address(contract_BEAN), 0, 100e6
 ↳ , LibTransfer.To.EXTERNAL);
099
100         console.log("DECIMALS            ---->  X000000000000");
101         console.log("USER 1 STALK        ----> ", siloFacet.
 ↳ balanceOfStalk(user1));
102     }
103
104
105     function test_X_withdrawDeposit02() public {
106         vm.startPrank(user1);
107
108         contract_BEAN.approve(address(siloFacet), 100e6);
109         siloFacet.deposit(address(contract_BEAN), 100e6,
 ↳ LibTransfer.From.EXTERNAL);
110         siloFacet.withdrawDeposit(address(contract_BEAN), 0, 50e6,
 ↳  LibTransfer.To.EXTERNAL);
```

```
111
112         console.log("DECIMALS             ---->  X000000000000");
113         console.log("USER 1 STALK         ----> ", siloFacet.
 ↳ balanceOfStalk(user1));
114     }
115
116
117     function test_X_deposit03() public {
118         _sunriseInvert(10000 * 1e6, 4 * 7 * 24);
119         vm.startPrank(user1);
120
121         contract_BEAN.approve(address(siloFacet), 100e6);
122         siloFacet.deposit(address(contract_BEAN), 100e6,
 ↳ LibTransfer.From.EXTERNAL);
123
124         _sunrise(10000 * 1e6);
125         // _sunrise(10000 * 1e6, 4 * 7 * 24);
126
127         console.log("DECIMALS            ---->  X000000000000");
128         console.log("USER1 STALK         ----> ", siloFacet.
 ↳ balanceOfStalk(user1));
129         console.log("USER1 GROWN STALK   ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
130         console.log("siloFacet.balanceOfEarnedStalk(user) -> ",
 ↳ siloFacet.balanceOfEarnedStalk(user1));
131     }
132
133
134     function test_X_deposit04() public {
135         _sunriseInvert(10000 * 1e6, 4 * 7 * 24);
136         vm.startPrank(user1);
137
138         contract_BEAN.approve(address(siloFacet), 100e6);
139         siloFacet.deposit(address(contract_BEAN), 100e6,
 ↳ LibTransfer.From.EXTERNAL);
140
141         _sunriseInvert(10000 * 1e6, 4 * 7 * 24);
142
143         console.log("DECIMALS            ---->  X000000000000");
144         console.log("USER1 STALK         ----> ", siloFacet.
 ↳ balanceOfStalk(user1));
145         console.log("USER1 GROWN STALK   ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
146     }
```

```
147
148     function test_X_deposit0302() public {
149         vm.startPrank(user1);
150
151         contract_BEAN.approve(address(siloFacet), 100e6);
152         siloFacet.deposit(address(contract_BEAN), 100e6,
 ↳ LibTransfer.From.EXTERNAL);
153
154         _sunrise(10000 * 1e6);
155         // _sunrise(10000 * 1e6, 4 * 7 * 24);
156
157         console.log("------------ GROWN STALK BY THROUGH SUNRISES
 ↳ ------------------");
158         _sunrise(10000 * 1e6);
159         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
160         _sunrise(10000 * 1e6);
161         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
162         _sunrise(10000 * 1e6);
163         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
164         _sunrise(10000 * 1e6);
165         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
166         _sunrise(10000 * 1e6);
167         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
168         _sunrise(10000 * 1e6);
169         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
170         _sunrise(10000 * 1e6);
171         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
172         _sunrise(10000 * 1e6);
173         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
174         _sunrise(10000 * 1e6);
175         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
176         _sunrise(10000 * 1e6);
177         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
178         _sunrise(10000 * 1e6);
```

```
179         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
180         _sunrise(10000 * 1e6);
181         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
182         _sunrise(10000 * 1e6);
183         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
184         _sunrise(10000 * 1e6);
185         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
186         _sunrise(10000 * 1e6);
187         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
188         _sunrise(10000 * 1e6);
189         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
190         _sunrise(10000 * 1e6);
191         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
192         _sunrise(10000 * 1e6);
193         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
194         _sunrise(10000 * 1e6);
195         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
196         _sunrise(10000 * 1e6);
197         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
198
199     }
200
201     function test_X_deposit0312() public {
202         vm.startPrank(user1);
203
204         contract_BEAN.approve(address(siloFacet), 100e6);
205         siloFacet.deposit(address(contract_BEAN), 100e6,
 ↳ LibTransfer.From.EXTERNAL);
206
207         _sunrise(10000 * 1e6);
208         // _sunrise(10000 * 1e6, 4 * 7 * 24);
209
210         console.log("------------ GROWN STALK BY THROUGH SUNRISES
 ↳ ------------------");
```

```
211         _sunrise(10000 * 1e6, 10);
212         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
213         _sunrise(10000 * 1e6, 10);
214         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
215         _sunrise(10000 * 1e6, 10);
216         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
217         _sunrise(10000 * 1e6, 10);
218         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
219         _sunrise(10000 * 1e6, 10);
220         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
221         _sunrise(10000 * 1e6, 10);
222         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
223         _sunrise(10000 * 1e6, 10);
224         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
225         _sunrise(10000 * 1e6, 10);
226         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
227         _sunrise(10000 * 1e6, 10);
228         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
229         _sunrise(10000 * 1e6, 10);
230         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
231         _sunrise(10000 * 1e6, 10);
232         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
233         _sunrise(10000 * 1e6, 10);
234         console.log("USER1 GROWN STALK  ----> ", siloFacet.
 ↳ balanceOfGrownStalk(user1, address(contract_BEAN)));
235     }
236
237     function test_X1_depositChecks() public {
238         vm.startPrank(user1);
239
240         contract_BEAN.approve(address(siloFacet), 100e6);
241         siloFacet.deposit(address(contract_BEAN), 100e6,
 ↳ LibTransfer.From.EXTERNAL);
```

```
242         console.log("EXECUTING DEPOSIT...");
243         accountInfo2(user1);
244 
245         console.log("SUNRAISE...");
246         _sunrise(1000e6);
247         accountInfo2(user1);
248 
249         console.log("SUNRAISE...");
250         _sunrise(1000e6);
251         accountInfo2(user1);
252 
253         console.log("SUNRAISE...");
254         _sunrise(1000e6);
255         accountInfo2(user1);
256 
257         console.log("SUNRAISE...");
258         _sunrise(1000e6);
259         accountInfo2(user1);
260     }
261 
262     function test_X1_depositChecksAndRoll() public {
263         vm.startPrank(user1);
264 
265         contract_BEAN.approve(address(siloFacet), 100e6);
266         siloFacet.deposit(address(contract_BEAN), 100e6,
 ↳ LibTransfer.From.EXTERNAL);
267         console.log("EXECUTING DEPOSIT...");
268         accountInfo2(user1);
269 
270         console.log("SUNRAISE...");
271         _sunrise(1000e6);
272         vm.roll(block.number + 10);
273         accountInfo2(user1);
274 
275         console.log("SUNRAISE...");
276         _sunrise(1000e6);
277         vm.roll(block.number + 10);
278         accountInfo2(user1);
279 
280         console.log("SUNRAISE...");
281         _sunrise(1000e6);
282         vm.roll(block.number + 10);
283         accountInfo2(user1);
284 
```

```
285         console.log("SUNRAISE...");
286         _sunrise(1000e6);
287         vm.roll(block.number + 10);
288         accountInfo2(user1);
289     }
290
291     function test_X1_depositPlantChecks() public {
292         vm.startPrank(user1);
293
294         contract_BEAN.approve(address(siloFacet), 100e6);
295         siloFacet.deposit(address(contract_BEAN), 100e6,
 ↳ LibTransfer.From.EXTERNAL);
296         console.log("EXECUTING DEPOSIT...");
297         accountInfo2(user1);
298
299         console.log("SUNRAISE...");
300         _sunrise(1000000e6);
301         accountInfo2(user1);
302
303         console.log("SUNRAISE...");
304         _sunrise(1000000e6);
305         accountInfo2(user1);
306
307         console.log("SUNRAISE...");
308         _sunrise(1000000e6);
309         accountInfo2(user1);
310
311         console.log("SUNRAISE...");
312         _sunrise(1000000e6);
313         accountInfo2(user1);
314
315         console.log("PLANT...");
316         siloFacet.plant(address(contract_BEAN));        // VULN,
 ↳ EARN STALK IS BIT CORRECTLY ADDED BEFOR MOW
317         accountInfo2(user1);
318
319         console.log("SUNRAISE...");
320         _sunrise(1000000e6);
321         accountInfo2(user1);
322     }
323
324     function test_X1_depositPlantWithdrawalChecksPartial() public
 ↳ {
325         vm.startPrank(user1);
```

```
326
327         contract_BEAN.approve(address(siloFacet), 100e6);
328         siloFacet.deposit(address(contract_BEAN), 100e6,
 ↳ LibTransfer.From.EXTERNAL);
329         console.log("EXECUTING DEPOSIT...");
330         accountInfo2(user1);
331
332         console.log("SUNRAISE...");
333         _sunrise(1000e6);
334         accountInfo2(user1);
335
336         console.log("SUNRAISE...");
337         _sunrise(1000e6);
338         accountInfo2(user1);
339
340         console.log("SUNRAISE...");
341         _sunrise(1000e6);
342         accountInfo2(user1);
343
344         console.log("SUNRAISE...");
345         _sunrise(1000e6);
346         accountInfo2(user1);
347
348         console.log("PLANT...");
349         siloFacet.plant(address(contract_BEAN));
350         accountInfo2(user1);
351
352         console.log("SUNRAISE...");
353         _sunrise(1000e6);
354         accountInfo2(user1);
355
356         console.log("SUNRAISE...");
357         _sunrise(1000e6);
358         accountInfo2(user1);
359
360         console.log("SUNRAISE...");
361         _sunrise(1000e6);
362         accountInfo2(user1);
363
364         console.log("SUNRAISE...");
365         _sunrise(1000e6);
366         accountInfo2(user1);
367
368         console.log("HALF WITHDRAWAL...");
```

```
369         siloFacet.withdrawDeposit(address(contract_BEAN), 0, 50e6,
 ↳  LibTransfer.To.EXTERNAL);
370         accountInfo2(user1);
371     }
372
373     function test_X1_depositPlantWithdrawalChecksTotal() public {
374         vm.startPrank(user1);
375
376         contract_BEAN.approve(address(siloFacet), 100e6);
377         siloFacet.deposit(address(contract_BEAN), 100e6,
 ↳ LibTransfer.From.EXTERNAL);
378         console.log("EXECUTING DEPOSIT...");
379         accountInfo2(user1);
380
381         console.log("SUNRAISE...");
382         _sunrise(1000e6);
383         accountInfo2(user1);
384
385         console.log("SUNRAISE...");
386         _sunrise(1000e6);
387         accountInfo2(user1);
388
389         console.log("SUNRAISE...");
390         _sunrise(1000e6);
391         accountInfo2(user1);
392
393         console.log("SUNRAISE...");
394         _sunrise(1000e6);
395         accountInfo2(user1);
396
397         console.log("PLANT...");
398         siloFacet.plant(address(contract_BEAN));
399         accountInfo2(user1);
400
401         console.log("SUNRAISE...");
402         _sunrise(1000e6);
403         accountInfo2(user1);
404
405         console.log("SUNRAISE...");
406         _sunrise(1000e6);
407         accountInfo2(user1);
408
409         console.log("SUNRAISE...");
410         _sunrise(1000e6);
```

```
411         accountInfo2(user1);
412
413         console.log("SUNRAISE...");
414         _sunrise(1000e6);
415         accountInfo2(user1);
416
417         console.log("HALF WITHDRAWAL...");
418         siloFacet.withdrawDeposit(address(contract_BEAN), 0, 100e6
 ↳ , LibTransfer.To.EXTERNAL);
419         accountInfo2(user1);
420     }
421
422     function test_X1_depositPlantWithdrawalChecksAfterSunrise()
 ↳ public {
423         console.log("SUNRAISE...");
424         _sunrise(1000e6);
425         accountInfo2(user1);
426         vm.startPrank(user1);
427
428         contract_BEAN.approve(address(siloFacet), 100e6);
429         siloFacet.deposit(address(contract_BEAN), 100e6,
 ↳ LibTransfer.From.EXTERNAL);
430         console.log("EXECUTING DEPOSIT...");
431         accountInfo2(user1);
432
433         console.log("SUNRAISE...");
434         _sunrise(1000e6);
435         accountInfo2(user1);
436
437         console.log("SUNRAISE...");
438         _sunrise(1000e6);
439         accountInfo2(user1);
440
441         console.log("SUNRAISE...");
442         _sunrise(1000e6);
443         accountInfo2(user1);
444
445         console.log("SUNRAISE...");
446         _sunrise(1000e6);
447         accountInfo2(user1);
448
449         console.log("PLANT...");
450         siloFacet.plant(address(contract_BEAN));
451         accountInfo2(user1);
```

```
452
453         console.log("SUNRAISE...");
454         _sunrise(1000e6);
455         accountInfo2(user1);
456
457         console.log("SUNRAISE...");
458         _sunrise(1000e6);
459         accountInfo2(user1);
460
461         console.log("SUNRAISE...");
462         _sunrise(1000e6);
463         accountInfo2(user1);
464
465         console.log("SUNRAISE...");
466         _sunrise(1000e6);
467         accountInfo2(user1);
468
469         console.log("HALF WITHDRAWAL...");
470         vm.expectRevert("Silo: Crate balance too low.");
471         siloFacet.withdrawDeposit(address(contract_BEAN), 0, 50e6,
 ↳  LibTransfer.To.EXTERNAL);
472         accountInfo2(user1);
473     }
474
475     function test_X1_depositPlantWithdrawalChecksExceed() public {
476         vm.startPrank(user1);
477
478         contract_BEAN.approve(address(siloFacet), 100e6);
479         siloFacet.deposit(address(contract_BEAN), 100e6,
 ↳ LibTransfer.From.EXTERNAL);
480         console.log("EXECUTING DEPOSIT...");
481         accountInfo2(user1);
482
483         console.log("SUNRAISE...");
484         _sunrise(1000e6);
485         accountInfo2(user1);
486
487         console.log("SUNRAISE...");
488         _sunrise(1000e6);
489         accountInfo2(user1);
490
491         console.log("SUNRAISE...");
492         _sunrise(1000e6);
493         accountInfo2(user1);
```

```
494
495         console.log("SUNRAISE...");
496         _sunrise(1000e6);
497         accountInfo2(user1);
498
499         console.log("PLANT...");
500         siloFacet.plant(address(contract_BEAN));
501         accountInfo2(user1);
502
503         console.log("SUNRAISE...");
504         _sunrise(1000e6);
505         accountInfo2(user1);
506
507         console.log("SUNRAISE...");
508         _sunrise(1000e6);
509         accountInfo2(user1);
510
511         console.log("SUNRAISE...");
512         _sunrise(1000e6);
513         accountInfo2(user1);
514
515         console.log("SUNRAISE...");
516         _sunrise(1000e6);
517         accountInfo2(user1);
518
519         console.log("HALF WITHDRAWAL...");
520         vm.expectRevert("Silo: Crate balance too low.");
521         siloFacet.withdrawDeposit(address(contract_BEAN), 0, 100
 ↳ _007000, LibTransfer.To.EXTERNAL);
522         accountInfo2(user1);
523     }
524
525     function test_X2_mow() public {
526         vm.startPrank(user1);
527
528         contract_BEAN.approve(address(siloFacet), 100e6);
529         siloFacet.deposit(address(contract_BEAN), 100e6,
 ↳ LibTransfer.From.EXTERNAL);
530         console.log("-----------------------------");
531         console.log("EXECUTING DEPOSIT...");
532         console.log("-----------------------------");
533         accountInfo2(user1);
534
535         console.log("-----------------------------");
```

```
536         console.log("10 SUNRAISES...");
537         console.log("------------------------------");
538         _sunrise(1000e6, 10);
539         accountInfo2(user1);
540 
541         console.log("------------------------------");
542         console.log("MOWING...");
543         console.log("------------------------------");
544         siloFacet.mow(user1, address(contract_BEAN));
545         accountInfo2(user1);
546 
547         console.log("------------------------------");
548         console.log("10 SUNRAISES...");
549         console.log("------------------------------");
550         _sunrise(1000e6, 10);
551         accountInfo2(user1);
552 
553         console.log("------------------------------");
554         console.log("PLANTING...");
555         console.log("------------------------------");
556         siloFacet.plant(address(contract_BEAN));
557         accountInfo2(user1);
558     }
559 
560     // purpose of this tests is to check the difference of general
 ↳  autocompounding vs manual plant and vs manual mow
561 
562     function test_X3_autocoumpound() public {
563         vm.startPrank(user1);
564 
565         contract_BEAN.approve(address(siloFacet), 100e6);
566         siloFacet.deposit(address(contract_BEAN), 100e6,
 ↳ LibTransfer.From.EXTERNAL);
567         console.log("------------------------------");
568         console.log("EXECUTING DEPOSIT...");
569         console.log("------------------------------");
570         accountInfo2(user1);
571 
572         console.log("------------------------------");
573         console.log("10000 SUNRAISES...");
574         console.log("------------------------------");
575         _sunrise(1000e6, 10000);
576         accountInfo2(user1);
577 
```

```
578         console.log("-----------------------------");
579         console.log("FINAL RESULT...");
580         console.log("-----------------------------");
581         console.log("STALK PERFORMANCE IN 10000 SEASONS ----> ",
 ↳ siloFacet.balanceOfStalk(user1) - 100e10);
582         console.log("STALK PERFORMANCE WITHOUT DECIMALS ----> ", (
 ↳ siloFacet.balanceOfStalk(user1) - 100e10) / 1e10);
583     }
584
585     function test_X3_manualMow() public {
586         vm.startPrank(user1);
587
588         contract_BEAN.approve(address(siloFacet), 100e6);
589         siloFacet.deposit(address(contract_BEAN), 100e6,
 ↳ LibTransfer.From.EXTERNAL);
590         console.log("-----------------------------");
591         console.log("EXECUTING DEPOSIT...");
592         console.log("-----------------------------");
593         accountInfo2(user1);
594
595         console.log("-----------------------------");
596         console.log("10000 SUNRAISES AND 1000 MOWS...");
597         console.log("-----------------------------");
598         for (uint i; i < 1000; ++i){
599             _sunrise(1000e6, 10);
600             siloFacet.mow(user1, address(contract_BEAN));
601         }
602         accountInfo2(user1);
603
604         console.log("-----------------------------");
605         console.log("FINAL RESULT...");
606         console.log("-----------------------------");
607         console.log("STALK PERFORMANCE IN 10000 SEASONS ----> ",
 ↳ siloFacet.balanceOfStalk(user1) - 100e10);
608         console.log("STALK PERFORMANCE WITHOUT DECIMALS ----> ", (
 ↳ siloFacet.balanceOfStalk(user1) - 100e10) / 1e10);
609     }
```

Listing 14: Poc.t.sol

```
611 function test_X3_manualPlant() public {
612         vm.startPrank(user1);
613
614         contract_BEAN.approve(address(siloFacet), 100e6);
```

```
615         siloFacet.deposit(address(contract_BEAN), 100e6,
 ↳ LibTransfer.From.EXTERNAL);
616         console.log("-----------------------------");
617         console.log("EXECUTING DEPOSIT...");
618         console.log("-----------------------------");
619         accountInfo2(user1);
620
621         console.log("-----------------------------");
622         console.log("10000 SUNRAISES AND 1000 PLANTS...");
623         console.log("-----------------------------");
624         for (uint i; i < 1000; ++i){
625             _sunrise(1000e6, 10);
626             siloFacet.plant(address(contract_BEAN));
627         }
628         accountInfo2(user1);
629
630         console.log("-----------------------------");
631         console.log("FINAL RESULT...");
632         console.log("-----------------------------");
633         console.log("STALK PERFORMANCE IN 10000 SEASONS ----> ",
 ↳ siloFacet.balanceOfStalk(user1) - 100e10);
634         console.log("STALK PERFORMANCE WITHOUT DECIMALS ----> ", (
 ↳ siloFacet.balanceOfStalk(user1) - 100e10) / 1e10);
635     }
636
637     // RFC1_00 CHECKS IF THE EARNED BEANS WORKS THE SAME WAY
 ↳ DEPENDING OF WHEN USER IS DEPOSITING
638     function test_X4_RFC1_000() public {
639         mockSeasonFacet.siloSunrise(1000e6);
640
641         vm.startPrank(user1);
642
643         // DEPOSIT JUST AFTER THE SUINRAISE
644         contract_BEAN.approve(address(siloFacet), 100e6);
645         siloFacet.deposit(address(contract_BEAN), 100e6,
 ↳ LibTransfer.From.EXTERNAL);
646
647         console.log("-----------------------------");
648         console.log("JUST AFTER DEPOSIT");
649         console.log("-----------------------------");
650         accountInfo2(user1);
651
652         // AFTER 20 BLOCKS
653         vm.roll(block.number + 20);
```

```
654         vm.warp(block.timestamp + 100);
655
656         console.log("-----------------------------");
657         console.log("20 BLOCKS LATER...");
658         console.log("-----------------------------");
659         accountInfo2(user1);
660
661         // AFTER 5 BLOCKS
662         vm.roll(block.number + 5);
663         vm.warp(block.timestamp + 100);
664
665         console.log("-----------------------------");
666         console.log("5 BLOCKS LATER...");
667         console.log("-----------------------------");
668         accountInfo2(user1);
669
670         // AFTER 1 BLOCKS
671         vm.roll(block.number + 1);
672         vm.warp(block.timestamp + 100);
673
674         console.log("-----------------------------");
675         console.log("1 BLOCKS LATER...");
676         console.log("-----------------------------");
677         accountInfo2(user1);
678
679
680         // AFTER 50 BLOCKS
681         vm.roll(block.number + 50);
682         vm.warp(block.timestamp + 100);
683
684         console.log("-----------------------------");
685         console.log("50 BLOCKS LATER...");
686         console.log("-----------------------------");
687         accountInfo2(user1);
688
689         // AFTER SUNRISE
690         mockSeasonFacet.siloSunrise(1000e6);
691
692         console.log("-----------------------------");
693         console.log("AFTER SUNRAISE");
694         console.log("-----------------------------");
695         accountInfo2(user1);
696
697         // AFTER 20 BLOCKS
```

```
698         vm.roll(block.number + 20);
699         vm.warp(block.timestamp + 100);
700
701         console.log("------------------------------");
702         console.log("20 BLOCKS LATER...");
703         console.log("------------------------------");
704         accountInfo2(user1);
705
706         // AFTER 5 BLOCKS
707         vm.roll(block.number + 5);
708         vm.warp(block.timestamp + 100);
709
710         console.log("------------------------------");
711         console.log("5 BLOCKS LATER...");
712         console.log("------------------------------");
713         accountInfo2(user1);
714
715         // AFTER 1 BLOCKS
716         vm.roll(block.number + 1);
717         vm.warp(block.timestamp + 100);
718
719         console.log("------------------------------");
720         console.log("1 BLOCKS LATER...");
721         console.log("------------------------------");
722         accountInfo2(user1);
723
724
725         // AFTER 50 BLOCKS
726         vm.roll(block.number + 50);
727         vm.warp(block.timestamp + 100);
728
729         console.log("------------------------------");
730         console.log("50 BLOCKS LATER...");
731         console.log("------------------------------");
732         accountInfo2(user1);
733     }
734
735     function test_X4_RFC1_001() public {
736         mockSeasonFacet.siloSunrise(1000e6);
737         vm.roll(block.number + 24);
738         vm.warp(block.timestamp + 100);
739
740         vm.startPrank(user1);
741
```

```
742         // DEPOSIT 24 BLOCKS AFTER LAST SUNRAISE
743         contract_BEAN.approve(address(siloFacet), 100e6);
744         siloFacet.deposit(address(contract_BEAN), 100e6,
 ↳ LibTransfer.From.EXTERNAL);
745
746         console.log("------------------------------");
747         console.log("DEPOSIT 24 BLOCKS AFTER LAST SUNRAISE");
748         console.log("------------------------------");
749         accountInfo2(user1);
750
751         // AFTER 20 BLOCKS
752         vm.roll(block.number + 20);
753         vm.warp(block.timestamp + 100);
754
755         console.log("------------------------------");
756         console.log("20 BLOCKS LATER...");
757         console.log("------------------------------");
758         accountInfo2(user1);
759
760         // AFTER 5 BLOCKS
761         vm.roll(block.number + 5);
762         vm.warp(block.timestamp + 100);
763
764         console.log("------------------------------");
765         console.log("5 BLOCKS LATER...");
766         console.log("------------------------------");
767         accountInfo2(user1);
768
769         // AFTER 1 BLOCKS
770         vm.roll(block.number + 1);
771         vm.warp(block.timestamp + 100);
772
773         console.log("------------------------------");
774         console.log("1 BLOCKS LATER...");
775         console.log("------------------------------");
776         accountInfo2(user1);
777
778         // AFTER 50 BLOCKS
779         vm.roll(block.number + 50);
780         vm.warp(block.timestamp + 100);
781
782         console.log("------------------------------");
783         console.log("50 BLOCKS LATER...");
784         console.log("------------------------------");
```

```
785         accountInfo2(user1);
786
787         // AFTER SUNRISE
788         mockSeasonFacet.siloSunrise(1000e6);
789
790         console.log("------------------------------");
791         console.log("AFTER SUNRAISE");
792         console.log("------------------------------");
793         accountInfo2(user1);
794
795         // AFTER 20 BLOCKS
796         vm.roll(block.number + 20);
797         vm.warp(block.timestamp + 100);
798
799         console.log("------------------------------");
800         console.log("20 BLOCKS LATER...");
801         console.log("------------------------------");
802         accountInfo2(user1);
803
804         // AFTER 5 BLOCKS
805         vm.roll(block.number + 5);
806         vm.warp(block.timestamp + 100);
807
808         console.log("------------------------------");
809         console.log("5 BLOCKS LATER...");
810         console.log("------------------------------");
811         accountInfo2(user1);
812
813         // AFTER 1 BLOCKS
814         vm.roll(block.number + 1);
815         vm.warp(block.timestamp + 100);
816
817         console.log("------------------------------");
818         console.log("1 BLOCKS LATER...");
819         console.log("------------------------------");
820         accountInfo2(user1);
821
822
823         // AFTER 50 BLOCKS
824         vm.roll(block.number + 50);
825         vm.warp(block.timestamp + 100);
826
827         console.log("------------------------------");
828         console.log("50 BLOCKS LATER...");
```

```
829         console.log("------------------------------");
830         accountInfo2(user1);
831
832
833         // AFTER ANOTHER SUNRISE
834         mockSeasonFacet.siloSunrise(1000e6);
835
836         console.log("------------------------------");
837         console.log("AFTER 2ND SUNRAISE");
838         console.log("------------------------------");
839         accountInfo2(user1);
840
841         // AFTER 20 BLOCKS
842         vm.roll(block.number + 20);
843         vm.warp(block.timestamp + 100);
844
845         console.log("------------------------------");
846         console.log("20 BLOCKS LATER...");
847         console.log("------------------------------");
848         accountInfo2(user1);
849
850         // AFTER 5 BLOCKS
851         vm.roll(block.number + 5);
852         vm.warp(block.timestamp + 100);
853
854         console.log("------------------------------");
855         console.log("5 BLOCKS LATER...");
856         console.log("------------------------------");
857         accountInfo2(user1);
858
859         // AFTER 1 BLOCKS
860         vm.roll(block.number + 1);
861         vm.warp(block.timestamp + 100);
862
863         console.log("------------------------------");
864         console.log("1 BLOCKS LATER...");
865         console.log("------------------------------");
866         accountInfo2(user1);
867
868
869         // AFTER 50 BLOCKS
870         vm.roll(block.number + 50);
871         vm.warp(block.timestamp + 100);
872
```

```
873         console.log("-----------------------------");
874         console.log("50 BLOCKS LATER...");
875         console.log("-----------------------------");
876         accountInfo2(user1);
877     }
878
879     function test_X4_RFC1_002() public {
880         mockSeasonFacet.siloSunrise(1000e6);
881         vm.roll(block.number + 25);
882         vm.warp(block.timestamp + 100);
883
884         vm.startPrank(user1);
885
886         // DEPOSIT 25 BLOCKS AFTER LAST SUNRAISE
887         contract_BEAN.approve(address(siloFacet), 100e6);
888         siloFacet.deposit(address(contract_BEAN), 100e6,
 ↳ LibTransfer.From.EXTERNAL);
889
890         console.log("-----------------------------");
891         console.log("DEPOSIT 26 BLOCKS AFTER LAST SUNRAISE");
892         console.log("-----------------------------");
893         accountInfo2(user1);
894
895         // AFTER 20 BLOCKS
896         vm.roll(block.number + 20);
897         vm.warp(block.timestamp + 100);
898
899         console.log("-----------------------------");
900         console.log("20 BLOCKS LATER...");
901         console.log("-----------------------------");
902         accountInfo2(user1);
903
904         // AFTER 5 BLOCKS
905         vm.roll(block.number + 5);
906         vm.warp(block.timestamp + 100);
907
908         console.log("-----------------------------");
909         console.log("5 BLOCKS LATER...");
910         console.log("-----------------------------");
911         accountInfo2(user1);
912
913         // AFTER 1 BLOCKS
914         vm.roll(block.number + 1);
915         vm.warp(block.timestamp + 100);
```

```
916
917         console.log("-----------------------------");
918         console.log("1 BLOCKS LATER...");
919         console.log("-----------------------------");
920         accountInfo2(user1);
921
922         // AFTER 50 BLOCKS
923         vm.roll(block.number + 50);
924         vm.warp(block.timestamp + 100);
925
926         console.log("-----------------------------");
927         console.log("50 BLOCKS LATER...");
928         console.log("-----------------------------");
929         accountInfo2(user1);
930
931         // AFTER SUNRISE
932         mockSeasonFacet.siloSunrise(1000e6);
933
934         console.log("-----------------------------");
935         console.log("AFTER SUNRAISE");
936         console.log("-----------------------------");
937         accountInfo2(user1);
938
939         // AFTER 20 BLOCKS
940         vm.roll(block.number + 20);
941         vm.warp(block.timestamp + 100);
942
943         console.log("-----------------------------");
944         console.log("20 BLOCKS LATER...");
945         console.log("-----------------------------");
946         accountInfo2(user1);
947
948         // AFTER 5 BLOCKS
949         vm.roll(block.number + 5);
950         vm.warp(block.timestamp + 100);
951
952         console.log("-----------------------------");
953         console.log("5 BLOCKS LATER...");
954         console.log("-----------------------------");
955         accountInfo2(user1);
956
957         // AFTER 1 BLOCKS
958         vm.roll(block.number + 1);
959         vm.warp(block.timestamp + 100);
```

```
1960
1961          console.log("------------------------------");
1962          console.log("1 BLOCKS LATER...");
1963          console.log("------------------------------");
1964          accountInfo2(user1);
1965
1966
1967          // AFTER 50 BLOCKS
1968          vm.roll(block.number + 50);
1969          vm.warp(block.timestamp + 100);
1970
1971          console.log("------------------------------");
1972          console.log("50 BLOCKS LATER...");
1973          console.log("------------------------------");
1974          accountInfo2(user1);
1975
1976          // AFTER ANOTHER SUNRISE
1977          mockSeasonFacet.siloSunrise(1000e6);
1978
1979          console.log("------------------------------");
1980          console.log("AFTER 2ND SUNRAISE");
1981          console.log("------------------------------");
1982          accountInfo2(user1);
1983
1984          // AFTER 20 BLOCKS
1985          vm.roll(block.number + 20);
1986          vm.warp(block.timestamp + 100);
1987
1988          console.log("------------------------------");
1989          console.log("20 BLOCKS LATER...");
1990          console.log("------------------------------");
1991          accountInfo2(user1);
1992
1993          // AFTER 5 BLOCKS
1994          vm.roll(block.number + 5);
1995          vm.warp(block.timestamp + 100);
1996
1997          console.log("------------------------------");
1998          console.log("5 BLOCKS LATER...");
1999          console.log("------------------------------");
2000          accountInfo2(user1);
2001
2002          // AFTER 1 BLOCKS
2003          vm.roll(block.number + 1);
```

```
2004         vm.warp(block.timestamp + 100);
2005
2006         console.log("------------------------------");
2007         console.log("1 BLOCKS LATER...");
2008         console.log("------------------------------");
2009         accountInfo2(user1);
2010
2011
2012         // AFTER 50 BLOCKS
2013         vm.roll(block.number + 50);
2014         vm.warp(block.timestamp + 100);
2015
2016         console.log("------------------------------");
2017         console.log("50 BLOCKS LATER...");
2018         console.log("------------------------------");
2019         accountInfo2(user1);
2020     }
2021
2022     // FUNCTIONS
2023     // DEPOSIT  ------------> contract_BEAN.approve(address(
  ↳ siloFacet), 100e6); siloFacet.deposit(address(contract_BEAN), 100
  ↳ e6, LibTransfer.From.EXTERNAL);
2024     // PLANT    ------------> siloFacet.plant(address(
  ↳ contract_BEAN));
2025     // MOW      ------------> siloFacet.mow(user1, address(
  ↳ contract_BEAN));
2026     // WITHDRAW ------------> siloFacet.withdrawDeposit(address(
  ↳ contract_BEAN), 0, 100_007000, LibTransfer.To.EXTERNAL);
2027     // SUNRISE  ------------> _sunrise(1000e6, 10);
2028     // CUST SUNRISE --------> mockSeasonFacet.siloSunrise(1000e6);
2029     // ROLL     ------------> vm.roll(block.number + 25);
2030     // LOG      ------------> console.log
  ↳ ("------------------------------");
2031     //                        console.log("EXECUTING DEPOSIT...");
2032     //                        console.log
  ↳ ("------------------------------");
2033     // ACCOUNT INFO --------> accountInfo2(user1);
2034
2035     // RFC1_01 HERE IS CHECKED HOW PLANT() WORKS DEPENDING ON THE
  ↳ VESTING TIME
2036     function test_X4_RFC1_010() public {
2037         mockSeasonFacet.siloSunrise(1000e6);
2038
2039         vm.startPrank(user1);
```

```
2040          contract_BEAN.approve(address(siloFacet), 100e6);
2041          siloFacet.deposit(address(contract_BEAN), 100e6,
  ↳ LibTransfer.From.EXTERNAL);
2042          vm.stopPrank();
2043
2044          vm.roll(block.number + 24);
2045
2046          mockSeasonFacet.siloSunrise(1000e6);
2047          siloFacet.plant(address(contract_BEAN));
2048      }
2049
2050      function test_X4_RFC1_011() public {
2051          mockSeasonFacet.siloSunrise(1000e6);
2052
2053          vm.startPrank(user1);
2054          contract_BEAN.approve(address(siloFacet), 100e6);
2055          siloFacet.deposit(address(contract_BEAN), 100e6,
  ↳ LibTransfer.From.EXTERNAL);
2056          vm.stopPrank();
2057
2058          vm.roll(block.number + 24);
2059
2060          mockSeasonFacet.siloSunrise(1000e6);
2061          vm.roll(block.number + 24);
2062
2063
2064          siloFacet.plant(address(contract_BEAN));
2065      }
2066
2067      function test_X4_RFC1_012() public {
2068          mockSeasonFacet.siloSunrise(1000e6);
2069
2070          vm.startPrank(user1);
2071          contract_BEAN.approve(address(siloFacet), 100e6);
2072          siloFacet.deposit(address(contract_BEAN), 100e6,
  ↳ LibTransfer.From.EXTERNAL);
2073          vm.stopPrank();
2074
2075          vm.roll(block.number + 24);
2076
2077          mockSeasonFacet.siloSunrise(1000e6);
2078          vm.roll(block.number + 25);
2079          siloFacet.plant(address(contract_BEAN));
2080      }
```

```
2081
2082     function test_X4_RFC1_013() public {
2083         mockSeasonFacet.siloSunrise(1000e6);
2084
2085         vm.startPrank(user1);
2086         contract_BEAN.approve(address(siloFacet), 100e6);
2087         siloFacet.deposit(address(contract_BEAN), 100e6,
  ↳ LibTransfer.From.EXTERNAL);
2088         vm.stopPrank();
2089
2090         vm.roll(block.number + 24);
2091
2092         mockSeasonFacet.siloSunrise(1000e6);
2093         vm.roll(block.number + 26);
2094         siloFacet.plant(address(contract_BEAN));
2095     }
2096
2097     function test_SetUpState() public view {
2098         console.log("\n\ntest_env_SetUpState()");
2099         console.log("___________________");
2100         console.log("Beanstalk owner -> %s", BeanstalkOwner());
2101         console.log("contract_SiloFacet.getTotalDeposited(address(
  ↳ contract_BEAN)) -> %s", siloFacet.getTotalDeposited(address(
  ↳ contract_BEAN)));
2102         console.log("contract_SiloFacet.getTotalWithdrawn(address(
  ↳ contract_BEAN)) -> %s", siloFacet.getTotalWithdrawn(address(
  ↳ contract_BEAN)));
2103         console.log("contract_SiloFacet.totalEarnedBeans() -> %s",
  ↳  siloFacet.totalEarnedBeans());
2104         console.log("contract_SiloFacet.totalRoots() -> %s",
  ↳ siloFacet.totalRoots());
2105         console.log("contract_SiloFacet.totalStalk() -> %s",
  ↳ siloFacet.totalStalk());
2106         console.log("contract_FieldFacet.totalSoil() -> %s",
  ↳ contract_FieldFacet.totalSoil());
2107         console.log("___________________\n\n");
2108     }
2109
2110     // Helpers
2111     function BeanstalkOwner() public view returns (address){
2112         return contract_OwnershipFacet.owner();
2113     }
2114
2115     function accountInfo(address user) internal {
```

```
2116         console.log("contract_BEAN.balanceOf(user) --> ",
  ↳ contract_BEAN.balanceOf(user));
2117         console.log("ERC20(C.UNRIPE_BEAN).balanceOf(user) --> ",
  ↳ ERC20(C.UNRIPE_BEAN).balanceOf(user));
2118         console.log("siloFacet.balanceOfStalk(user) -> ",
  ↳ siloFacet.balanceOfStalk(user));
2119         console.log("siloFacet.balanceOfRoots(user) -> ",
  ↳ siloFacet.balanceOfRoots(user));
2120         console.log("siloFacet.balanceOfGrownStalk(user, address(
  ↳ contract_BEAN))) -> ", siloFacet.balanceOfGrownStalk(user, address
  ↳ (contract_BEAN)));
2121         console.log("siloFacet.balanceOfEarnedBeans(user) -> ",
  ↳ siloFacet.balanceOfEarnedBeans(user));
2122         console.log("siloFacet.balanceOfEarnedStalk(user) -> ",
  ↳ siloFacet.balanceOfEarnedStalk(user));
2123     }
2124
2125     function accountInfo2(address user) internal {
2126         console.log("BEAN BALANCE ------------> ", contract_BEAN.
  ↳ balanceOf(user));
2127         console.log("STALK BALANCE ------------> ", siloFacet.
  ↳ balanceOfStalk(user));
2128         //console.log("ROOTS BALANCE ------------> ", siloFacet.
  ↳ balanceOfRoots(user));
2129         console.log("GROWN STALK FOR BEANS ----> ", siloFacet.
  ↳ balanceOfGrownStalk(user, address(contract_BEAN)));
2130         console.log("EARNED BEANS -------------> ", siloFacet.
  ↳ balanceOfEarnedBeans(user));
2131         console.log("EARNED STALK -------------> ", siloFacet.
  ↳ balanceOfEarnedStalk(user));
2132     }
2133
2134     function _sunrise(uint256 amount) internal {
2135         vm.roll(block.number + 25);
2136         vm.warp(block.timestamp + 3600);
2137         mockSeasonFacet.siloSunrise(amount);
2138     }
2139
2140     function _sunrise(uint256 amount, uint256 seasons) internal {
2141         for (uint256 i; i < seasons; ++i) {
2142             vm.warp(block.timestamp + 3600);
2143             mockSeasonFacet.siloSunrise(amount);
2144         }
2145     }
```

```
2146
2147     function _sunriseInvert(uint256 amount) internal {
2148         vm.roll(block.number + 25);
2149         mockSeasonFacet.siloSunrise(amount);
2150         vm.warp(block.timestamp + 3600);
2151         vm.roll(block.number + 25);
2152     }
2153
2154     function _sunriseInvert(uint256 amount, uint256 seasons)
  ↳ internal {
2155         for (uint256 i; i < seasons; ++i) {
2156             vm.roll(block.number + 25);
2157             mockSeasonFacet.siloSunrise(amount);
2158             vm.warp(block.timestamp + 3600);
2159             vm.roll(block.number + 25);
2160         }
2161     }
2162
2163     function getSelector(string memory _func) public pure returns
  ↳ (bytes4) {
2164         return bytes4(keccak256(bytes(_func)));
2165     }
2166
2167     function getSelectorsApprovalFacet() public returns (bytes4[]
  ↳ memory) {
2168         selectorsApprovalFacet.push(getSelector("approveDeposit(
  ↳ address,address,uint256)"));
2169         selectorsApprovalFacet.push(getSelector("
  ↳ increaseDepositAllowance(address,address,uint256)"));
2170         selectorsApprovalFacet.push(getSelector("
  ↳ decreaseDepositAllowance(address,address,uint256)"));
2171         // selectorsApprovalFacet.push(getSelector("permitDeposits
  ↳ (address,address,address[],uint256[],uint256,uint8,bytes32,bytes32
  ↳ )"));
2172         // selectorsApprovalFacet.push(getSelector("permitDeposit(
  ↳ address,address,address,uint256,uint256,uint8,bytes32,bytes32)"));
2173         selectorsApprovalFacet.push(getSelector("
  ↳ depositPermitNonces(address)"));
2174         selectorsApprovalFacet.push(getSelector("
  ↳ depositPermitDomainSeparator()"));
2175         // selectorsApprovalFacet.push(getSelector("
  ↳ depositAllowance(address,address,address)"));
2176         // selectorsApprovalFacet.push(getSelector("
  ↳ setApprovalForAll(address,bool)"));
```

```
2177         // selectorsApprovalFacet.push(getSelector("
  ↳ isApprovedForAll(address,address)"));
2178         return selectorsApprovalFacet;
2179     }
2180 
2181     function getSelectorsMigrationFacet() public returns (bytes4[]
  ↳  memory) {
2182         selectorsMigrationFacet.push(getSelector("mowAndMigrate(
  ↳ address,address[],uint32[][],uint256[][],uint256,uint256,bytes32
  ↳ [])"));
2183         selectorsMigrationFacet.push(getSelector("
  ↳ mowAndMigrateNoDeposits(address)"));
2184         return selectorsMigrationFacet;
2185     }
2186 
2187     function getSelectorsLegacyClaimWithdrawalFacet() public
  ↳ returns (bytes4[] memory) {
2188         selectorsLegacyClaimWithdrawalFacet.push(getSelector("
  ↳ claimWithdrawal(address,uint32,uint8)"));
2189         selectorsLegacyClaimWithdrawalFacet.push(getSelector("
  ↳ claimWithdrawals(address,uint32[],uint8)"));
2190         // selectorsLegacyClaimWithdrawalFacet.push(getSelector("
  ↳ getWithdrawal(address,address,uint32)"));
2191         // selectorsLegacyClaimWithdrawalFacet.push(getSelector("
  ↳ getTotalWithdrawn(address)"));
2192         return selectorsLegacyClaimWithdrawalFacet;
2193     }
2194 
2195     function getSelectorsFarmFacet() public returns (bytes4[]
  ↳ memory) {
2196         selectorsFarmFace.push(getSelector("farm(bytes[])"));
2197         selectorsFarmFace.push(getSelector("advancedFarm((bytes,
  ↳ bytes)[])"));
2198         return selectorsFarmFace;
2199     }
2200 
2201     function getSelectorsBDVFacet() public returns (bytes4[]
  ↳ memory) {
2202         selectorsBDVFacet.push(getSelector("curveToBDV(uint256)"))
  ↳ ;
2203         selectorsBDVFacet.push(getSelector("beanToBDV(uint256)"));
2204         selectorsBDVFacet.push(getSelector("unripeLPToBDV(uint256)
  ↳ "));
2205         selectorsBDVFacet.push(getSelector("unripeBeanToBDV(
```

```
↳ uint256)"));
2206          selectorsBDVFacet.push(getSelector("bdv(address,uint256)")
↳ );
2207          return selectorsBDVFacet;
2208      }
2209
2210      function getSelectorsConvertFacet() public returns (bytes4[]
↳ memory) {
2211          selectorsConvertFacet.push(getSelector("convert(bytes,
↳ int96[],uint256[])"));
2212          selectorsConvertFacet.push(getSelector("enrootDeposit(
↳ address,int96,uint256)"));
2213          selectorsConvertFacet.push(getSelector("enrootDeposits(
↳ address,int96[],uint256[])"));
2214          selectorsConvertFacet.push(getSelector("getMaxAmountIn(
↳ address,address)"));
2215          selectorsConvertFacet.push(getSelector("getAmountOut(
↳ address,address,uint256)"));
2216          return selectorsConvertFacet;
2217      }
2218
2219      function getSelectorsConvertFacetOld() public returns (bytes4
↳ [] memory) {
2220          selectorsConvertFacetOld.push(getSelector("convert(bytes,
↳ uint32[],uint256[])"));
2221          selectorsConvertFacetOld.push(getSelector("getMaxAmountIn(
↳ address,address)"));
2222          selectorsConvertFacetOld.push(getSelector("getAmountOut(
↳ address,address,uint256)"));
2223          return selectorsConvertFacetOld;
2224      }
2225
2226      function getSelectorsSiloFacet() public returns (bytes4[]
↳ memory) {
2227          selectorsSiloFacet.push(getSelector("deposit(address,
↳ uint256,uint8)"));
2228          selectorsSiloFacet.push(getSelector("withdrawDeposit(
↳ address,int96,uint256,uint8)"));
2229          selectorsSiloFacet.push(getSelector("withdrawDeposits(
↳ address,int96[],uint256[],uint8)"));
2230          selectorsSiloFacet.push(getSelector("transferDeposit(
↳ address,address,address,int96,uint256)"));
2231          selectorsSiloFacet.push(getSelector("transferDeposits(
↳ address,address,address,int96[],uint256[])"));
```

```
2232          selectorsSiloFacet.push(getSelector("safeTransferFrom(
  ↳ address,address,uint256,uint256,bytes)"));
2233          selectorsSiloFacet.push(getSelector("safeBatchTransferFrom
  ↳ (address,address,uint256[],uint256[],bytes)"));
2234          selectorsSiloFacet.push(getSelector("mow(address,address)"
  ↳ ));
2235          selectorsSiloFacet.push(getSelector("mowMultiple(address,
  ↳ address[])"));
2236          selectorsSiloFacet.push(getSelector("plant(address)"));
2237          selectorsSiloFacet.push(getSelector("claimPlenty()"));
2238          selectorsSiloFacet.push(getSelector("balanceOfStalk(
  ↳ address)"));
2239          selectorsSiloFacet.push(getSelector("balanceOfRoots(
  ↳ address)"));
2240          selectorsSiloFacet.push(getSelector("balanceOfGrownStalk(
  ↳ address,address)"));
2241          selectorsSiloFacet.push(getSelector("balanceOfEarnedBeans(
  ↳ address)"));
2242          selectorsSiloFacet.push(getSelector("balanceOfEarnedStalk(
  ↳ address)"));
2243          // selectorsSiloFacet.push(getSelector("lastSeasonOfPlenty
  ↳ ()"));
2244          selectorsSiloFacet.push(getSelector("balanceOfPlenty(
  ↳ address)"));
2245          selectorsSiloFacet.push(getSelector("balanceOfRainRoots(
  ↳ address)"));
2246          selectorsSiloFacet.push(getSelector("balanceOfSop(address)
  ↳ "));
2247          selectorsSiloFacet.push(getSelector("stemTipForToken(
  ↳ address)"));
2248          selectorsSiloFacet.push(getSelector("seasonToStem(address,
  ↳ uint32)"));
2249          selectorsSiloFacet.push(getSelector("stemStartSeason()"));
2250          return selectorsSiloFacet;
2251      }
2252
2253      function getSelectorsSiloFacetOld() public returns (bytes4[]
  ↳ memory) {
2254          selectorsSiloFacetOld.push(getSelector("deposit(address,
  ↳ uint256,uint8)"));
2255          selectorsSiloFacetOld.push(getSelector("withdrawDeposit(
  ↳ address,uint32,uint256)"));
2256          selectorsSiloFacetOld.push(getSelector("withdrawDeposits(
  ↳ address,uint32[],uint256[])"));
```

```
2257          selectorsSiloFacetOld.push(getSelector("claimWithdrawal(
  ↳ address,uint32,uint8)"));
2258          selectorsSiloFacetOld.push(getSelector("claimWithdrawals(
  ↳ address,uint32[],uint8)"));
2259          selectorsSiloFacetOld.push(getSelector("transferDeposit(
  ↳ address,address,address,uint32,uint256)"));
2260          selectorsSiloFacetOld.push(getSelector("transferDeposits(
  ↳ address,address,address,uint32[],uint256[])"));
2261          selectorsSiloFacetOld.push(getSelector("approveDeposit(
  ↳ address,address,uint256)"));
2262          selectorsSiloFacetOld.push(getSelector("
  ↳ increaseDepositAllowance(address,address,uint256)"));
2263          selectorsSiloFacetOld.push(getSelector("
  ↳ decreaseDepositAllowance(address,address,uint256)"));
2264          selectorsSiloFacetOld.push(getSelector("permitDeposits(
  ↳ address,address,address[],uint256[],uint256,uint8,bytes32,bytes32)
  ↳ "));
2265          selectorsSiloFacetOld.push(getSelector("permitDeposit(
  ↳ address,address,address,uint256,uint256,uint8,bytes32,bytes32)"));
2266          selectorsSiloFacetOld.push(getSelector("
  ↳ depositPermitNonces(address)"));
2267          selectorsSiloFacetOld.push(getSelector("
  ↳ depositPermitDomainSeparator()"));
2268          selectorsSiloFacetOld.push(getSelector("update(address)"))
  ↳ ;
2269          selectorsSiloFacetOld.push(getSelector("plant()"));
2270          selectorsSiloFacetOld.push(getSelector("claimPlenty()"));
2271          selectorsSiloFacetOld.push(getSelector("enrootDeposits(
  ↳ address,uint32[],uint256[])"));
2272          selectorsSiloFacetOld.push(getSelector("enrootDeposit(
  ↳ address,uint32,uint256)"));
2273          selectorsSiloFacetOld.push(getSelector("balanceOfStalk(
  ↳ address)"));
2274          selectorsSiloFacetOld.push(getSelector("balanceOfRoots(
  ↳ address)"));
2275          selectorsSiloFacetOld.push(getSelector("
  ↳ balanceOfGrownStalk(address)"));
2276          selectorsSiloFacetOld.push(getSelector("
  ↳ balanceOfEarnedBeans(address)"));
2277          selectorsSiloFacetOld.push(getSelector("
  ↳ balanceOfEarnedStalk(address)"));
2278          selectorsSiloFacetOld.push(getSelector("balanceOfRainRoots
  ↳ (address)"));
2279          selectorsSiloFacetOld.push(getSelector("balanceOfSop(
```

```
↳ address)"));
2280          selectorsSiloFacetOld.push(getSelector("balanceOfPlenty(
↳ address)"));
2281          selectorsSiloFacetOld.push(getSelector("balanceOfSeeds(
↳ address)"));
2282          selectorsSiloFacetOld.push(getSelector("
↳ balanceOfEarnedSeeds(address)"));
2283          return selectorsSiloFacetOld;
2284      }
2285
2286      function getSelectorsSeasonFacet() public returns (bytes4[]
↳ memory) {
2287          selectorsSeasonFacet.push(getSelector("sunrise()"));
2288          selectorsSeasonFacet.push(getSelector("season()"));
2289          selectorsSeasonFacet.push(getSelector("paused()"));
2290          selectorsSeasonFacet.push(getSelector("time()"));
2291          selectorsSeasonFacet.push(getSelector("seasonTime()"));
2292          return selectorsSeasonFacet;
2293      }
2294
2295      function getSelectorsMockSeasonFacet() public returns (bytes4
↳ [] memory) {
2296          selectorsMockSeasonFacet.push(getSelector("
↳ reentrancyGuardTest()"));
2297          selectorsMockSeasonFacet.push(getSelector("rsetYieldE(
↳ uint256)"));
2298          selectorsMockSeasonFacet.push(getSelector("siloSunrise(
↳ uint256)"));
2299          selectorsMockSeasonFacet.push(getSelector("mockStepSilo(
↳ uint256)"));
2300          selectorsMockSeasonFacet.push(getSelector("rainSunrise()")
↳ );
2301          selectorsMockSeasonFacet.push(getSelector("rainSunrises(
↳ uint256)"));
2302          selectorsMockSeasonFacet.push(getSelector("droughtSunrise
↳ ()"));
2303          selectorsMockSeasonFacet.push(getSelector("rainSiloSunrise
↳ (uint256)"));
2304          selectorsMockSeasonFacet.push(getSelector("
↳ droughtSiloSunrise(uint256)"));
2305          selectorsMockSeasonFacet.push(getSelector("sunSunrise(
↳ int256,uint256)"));
2306          selectorsMockSeasonFacet.push(getSelector("
↳ sunTemperatureSunrise(int256,uint256,uint32)"));
```

```
2307          selectorsMockSeasonFacet.push(getSelector("lightSunrise()"
  ↳ ));
2308          selectorsMockSeasonFacet.push(getSelector("fastForward(
  ↳ uint32)"));
2309          selectorsMockSeasonFacet.push(getSelector("teleportSunrise
  ↳ (uint32)"));
2310          selectorsMockSeasonFacet.push(getSelector("farmSunrise()")
  ↳ );
2311          selectorsMockSeasonFacet.push(getSelector("farmSunrises(
  ↳ uint256)"));
2312          selectorsMockSeasonFacet.push(getSelector("setMaxTempE(
  ↳ uint32)"));
2313          selectorsMockSeasonFacet.push(getSelector("setAbovePegE(
  ↳ bool)"));
2314          selectorsMockSeasonFacet.push(getSelector("setLastDSoilE(
  ↳ uint128)"));
2315          selectorsMockSeasonFacet.push(getSelector("setNextSowTimeE
  ↳ (uint32)"));
2316          selectorsMockSeasonFacet.push(getSelector("setLastSowTimeE
  ↳ (uint32)"));
2317          selectorsMockSeasonFacet.push(getSelector("setSoilE(
  ↳ uint256)"));
2318          selectorsMockSeasonFacet.push(getSelector("resetAccount(
  ↳ address)"));
2319          selectorsMockSeasonFacet.push(getSelector("
  ↳ resetAccountToken(address,address)"));
2320          selectorsMockSeasonFacet.push(getSelector("resetState()"))
  ↳ ;
2321          selectorsMockSeasonFacet.push(getSelector("stepWeatherE(
  ↳ int256,uint128)"));
2322          selectorsMockSeasonFacet.push(getSelector("
  ↳ setCurrentSeasonE(uint32)"));
2323          selectorsMockSeasonFacet.push(getSelector("
  ↳ stepWeatherWithParams(uint256,uint256,uint128,uint128,int256,bool,
  ↳ bool)"));
2324          selectorsMockSeasonFacet.push(getSelector("
  ↳ resetSeasonStart(uint256)"));
2325          selectorsMockSeasonFacet.push(getSelector("captureE()"));
2326          selectorsMockSeasonFacet.push(getSelector("captureCurveE()
  ↳ "));
2327          selectorsMockSeasonFacet.push(getSelector("
  ↳ updateTWAPCurveE()"));
2328          selectorsMockSeasonFacet.push(getSelector("curveOracle()")
  ↳ );
```

```
2329          selectorsMockSeasonFacet.push(getSelector("resetPools(
  ↳ address[])"));
2330          selectorsMockSeasonFacet.push(getSelector("
  ↳ rewardToFertilizerE(uint256)"));
2331          selectorsMockSeasonFacet.push(getSelector("getEthPrice()")
  ↳ );
2332          selectorsMockSeasonFacet.push(getSelector("lastDSoil()"));
2333          selectorsMockSeasonFacet.push(getSelector("lastSowTime()")
  ↳ );
2334          selectorsMockSeasonFacet.push(getSelector("thisSowTime()")
  ↳ );
2335          selectorsMockSeasonFacet.push(getSelector("getT()"));
2336          selectorsMockSeasonFacet.push(getSelector("
  ↳ deployStemsUpgrade()"));
2337          return selectorsMockSeasonFacet;
2338      }
2339  }
```

AUTOMATED TESTING

7.1 STATIC ANALYSIS REPORT

Description:

Halborn used automated testing techniques to enhance the coverage of certain areas of the scoped contracts. Among the tools used was Slither, a Solidity static analysis framework. After Halborn verified all the contracts in the repository and was able to compile them correctly into their ABI and binary formats, Slither was run on the all-scoped 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:

protocol/contracts/beanstalk/silo/ApprovalFacet.sol

```
LibCurve.getY(uint256,uint256[2],uint256,uint256) (contracts/libraries/Curve/LibCurve.sol#49-81) performs a multiplication on the result of a division:
        -c = (c * D) / (_x * N_COINS) (contracts/libraries/Curve/LibCurve.sol#67)
        -c = (c * D * A_PRECISION) / (Ann * N_COINS) (contracts/libraries/Curve/LibCurve.sol#70)
LibCurve.getD(uint256[2],uint256) (contracts/libraries/Curve/LibCurve.sol#83-112) performs a multiplication on the result of a division:
        -D_P = (D_P * D) / (xp[_j] * N_COINS) (contracts/libraries/Curve/LibCurve.sol#101)
        -D = ((Ann * S) / A_PRECISION + D_P * N_COINS) * D) / (((Ann - A_PRECISION) * D) / A_PRECISION + (N_COINS + 1) * D_P) (contracts/libraries/Curve/LibCurve.sol#104-106)
LibCurve.getY0(uint256,uint256,uint256[2],uint256) (contracts/libraries/Curve/LibCurve.sol#114-145) performs a multiplication on the result of a division:
        -c = (c * D) / (_x * N_COINS) (contracts/libraries/Curve/LibCurve.sol#131)
        -c = (c * D * A_PRECISION) / (Ann * N_COINS) (contracts/libraries/Curve/LibCurve.sol#134)
LibPRBMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/LibPRBMath.sol#185-263) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/LibPRBMath.sol#231)
        -inverse = (3 * denominator) ^ 2 (contracts/libraries/LibPRBMath.sol#246)
LibPRBMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/LibPRBMath.sol#185-263) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/LibPRBMath.sol#231)
        -inverse *= 2 - denominator * inverse (contracts/libraries/LibPRBMath.sol#250)
LibPRBMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/LibPRBMath.sol#185-263) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/LibPRBMath.sol#231)
        -inverse *= 2 - denominator * inverse (contracts/libraries/LibPRBMath.sol#251)
LibPRBMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/LibPRBMath.sol#185-263) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/LibPRBMath.sol#231)
        -inverse *= 2 - denominator * inverse (contracts/libraries/LibPRBMath.sol#252)
LibPRBMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/LibPRBMath.sol#185-263) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/LibPRBMath.sol#231)
        -inverse *= 2 - denominator * inverse (contracts/libraries/LibPRBMath.sol#253)
LibPRBMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/LibPRBMath.sol#185-263) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/LibPRBMath.sol#231)
        -inverse *= 2 - denominator * inverse (contracts/libraries/LibPRBMath.sol#254)
LibPRBMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/LibPRBMath.sol#185-263) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/LibPRBMath.sol#231)
        -inverse *= 2 - denominator * inverse (contracts/libraries/LibPRBMath.sol#255)
LibPRBMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/LibPRBMath.sol#185-263) performs a multiplication on the result of a division:
        -prod0 = prod0 / twos (contracts/libraries/LibPRBMath.sol#234)
        -result = prod0 * inverse (contracts/libraries/LibPRBMath.sol#261)
LibUnripeSilo._removeUnripeLPDeposit(address,uint32,uint256) (contracts/libraries/Silo/LibUnripeSilo.sol#212-289) performs a multiplication on the result of a division:
        -removed = amount.mul(bdv1).div(amount1) (contracts/libraries/Silo/LibUnripeSilo.sol#230)
        -s.a[account].lp.depositSeeds[season] = s.a[account].lp.depositSeeds[season].sub(removed.mul(4)) (contracts/libraries/Silo/LibUnripeSilo.sol#231-235)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

Contract locking ether found:
        Contract ApprovalFacet (contracts/beanstalk/silo/ApprovalFacet.sol#24-167) has payable functions:
         - ApprovalFacet.approveDeposit(address,address,uint256) (contracts/beanstalk/silo/ApprovalFacet.sol#47-51)
         - ApprovalFacet.permitDeposits(address,address,address[],uint256[],uint256[],uint8[],bytes32[],bytes32[]) (contracts/beanstalk/silo/ApprovalFacet.sol#108-113)
         - ApprovalFacet.permitDeposit(address,address,address,uint256,uint256,uint8,bytes32,bytes32) (contracts/beanstalk/silo/ApprovalFacet.sol#127-130)
        But does not have a function to withdraw the ether
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#contracts-that-lock-ether

Reentrancy in Silo._claimPlenty(address) (contracts/beanstalk/silo/SiloFacet/Silo.sol#147-154):
        External calls:
        - C.threeCrv().safeTransfer(account,plenty) (contracts/beanstalk/silo/SiloFacet/Silo.sol#150)
        State variables written after the call(s):
        - delete s.a[account].sop.plenty (contracts/beanstalk/silo/SiloFacet/Silo.sol#151)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

LibCurve.getD(uint256[2],uint256)._j (contracts/libraries/Curve/LibCurve.sol#100) is a local variable never initialized
LibCurve.getY0(uint256,uint256,uint256[2],uint256)._i_scope_0 (contracts/libraries/Curve/LibCurve.sol#138) is a local variable never initialized
LibLegacyTokenSilo._mowAndMigrate(address,address[],uint32[][],uint256[][]).migrateData (contracts/libraries/Silo/LibLegacyTokenSilo.sol#367) is a local variable never initialized
LibDiamond.addFunctions(address,bytes4[]).selectorIndex (contracts/libraries/LibDiamond.sol#129) is a local variable never initialized
ApprovalFacet.permitDeposits(address,address,address[],uint256[],uint256,uint8,bytes32,bytes32).i (contracts/beanstalk/silo/ApprovalFacet.sol#110) is a local variable never initialized
LibLegacyTokenSilo._claimWithdrawals(address,address,uint32[]).i (contracts/libraries/Silo/LibLegacyTokenSilo.sol#587) is a local variable never initialized
LibDiamond.diamondCut(IDiamondCut.FacetCut[],address,bytes).facetIndex (contracts/libraries/LibDiamond.sol#184) is a local variable never initialized
LibCurve.getY(uint256,uint256[2],uint256,uint256)._i_scope_0 (contracts/libraries/Curve/LibCurve.sol#74) is a local variable never initialized
LibSilo._removeDepositsFromAccount(address,address,int96[],uint256[]).i (contracts/libraries/Silo/LibSilo.sol#555) is a local variable never initialized
LibDiamond.removeFunctions(address,bytes4[]).selectorIndex (contracts/libraries/LibDiamond.sol#162) is a local variable never initialized
LibCurve.getD(uint256[2],uint256)._i (contracts/libraries/Curve/LibCurve.sol#91) is a local variable never initialized
LibCurve.getYD(uint256,uint256,uint256[2],uint256)._i (contracts/libraries/Curve/LibCurve.sol#127) is a local variable never initialized
LibDiamond.replaceFunctions(address,bytes4[]).selectorIndex (contracts/libraries/LibDiamond.sol#147) is a local variable never initialized
LibCurve.getY(uint256,uint256[2],uint256,uint256)._i (contracts/libraries/Curve/LibCurve.sol#62) is a local variable never initialized
TokenSilo._transferDeposits(address,address,address,int96[],uint256[]).i (contracts/beanstalk/silo/SiloFacet/TokenSilo.sol#335) is a local variable never initialized
LibCurve.getD(uint256[2],uint256)._i_scope_0 (contracts/libraries/Curve/LibCurve.sol#98) is a local variable never initialized
LibLegacyTokenSilo._mowAndMigrate(address,address[],uint32[][],uint256[][]).perTokenData (contracts/libraries/Silo/LibLegacyTokenSilo.sol#371) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

LibUnripeSilo.unripeLPDeposit(address,uint32).bdv (contracts/libraries/Silo/LibUnripeSilo.sol#306) is written in both
```

```
        (amount,bdv) = getBeanEthUnripeLP(account,season) (contracts/libraries/Silo/LibUnripeSilo.sol#312)
        bdv = uint256(s.a[account].legacyDeposits[C.unripeLPAddress()][season].bdv).add(legBdv) (contracts/libraries/Silo/LibUnripeSilo.sol#329-331)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#write-after-write

ApprovalFacet.permitDeposits(address,address,address[],uint256[],uint256,uint8,bytes32,bytes32).s (contracts/beanstalk/silo/ApprovalFacet.sol#108) shadows:
        - ReentrancyGuard.s (contracts/beanstalk/ReentrancyGuard.sol#17) (state variable)
ApprovalFacet.permitDeposit(address,address,address,uint256,uint256,uint8,bytes32,bytes32).s (contracts/beanstalk/silo/ApprovalFacet.sol#127) shadows:
        - ReentrancyGuard.s (contracts/beanstalk/ReentrancyGuard.sol#17) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

Variable 'LibLegacyTokenSilo._mowAndMigrate(address,address[],uint32[][],uint256[][]).perDepositData (contracts/libraries/Silo/LibLegacyTokenSilo.sol#376)' in LibLegacyTokenSilo._mowAndMigrate(address,address[],uin
/libraries/Silo/LibLegacyTokenSilo.sol#345-434) potentially used before declaration: j < seasons[i].length (contracts/libraries/Silo/LibLegacyTokenSilo.sol#375)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

Reentrancy in Silo._claimPlenty(address) (contracts/beanstalk/silo/SiloFacet/Silo.sol#147-154):
        External calls:
        - C.threeCrv().safeTransfer(account,plenty) (contracts/beanstalk/silo/SiloFacet/Silo.sol#150)
        Event emitted after the call(s):
        - ClaimPlenty(account,plenty) (contracts/beanstalk/silo/SiloFacet/Silo.sol#153)
Reentrancy in LibTokenSilo.deposit(address,address,int96,uint256) (contracts/libraries/Silo/LibTokenSilo.sol#108-116):
        External calls:
        - bdv = beanDenominatedValue(token,amount) (contracts/libraries/Silo/LibTokenSilo.sol#114)
                - (success,data) = address(this).call(callData) (contracts/libraries/Silo/LibTokenSilo.sol#286-288)
        Event emitted after the call(s):
        - AddDeposit(account,token,stem,amount,bdv) (contracts/libraries/Silo/LibTokenSilo.sol#203)
                - depositWithBDV(account,token,stem,amount,bdv) (contracts/libraries/Silo/LibTokenSilo.sol#115)
        - TransferSingle(msg.sender,address(0),account,uint256(depositId),amount) (contracts/libraries/Silo/LibTokenSilo.sol#195-201)
                - depositWithBDV(account,token,stem,amount,bdv) (contracts/libraries/Silo/LibTokenSilo.sol#115)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

LibSiloPermit.permit(address,address,address,uint256,uint256,uint8,bytes32,bytes32) (contracts/libraries/Silo/LibSiloPermit.sol#57-82) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= deadline,Silo: permit expired deadline) (contracts/libraries/Silo/LibSiloPermit.sol#67)
LibSiloPermit.permits(address,address,address[],uint256[],uint256,uint8,bytes32,bytes32) (contracts/libraries/Silo/LibSiloPermit.sol#93-118) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= deadline,Silo: permit expired deadline) (contracts/libraries/Silo/LibSiloPermit.sol#103)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

ECDSA.recover(bytes32,bytes) (lib/openzeppelin-contracts/contracts/cryptography/ECDSA.sol#26-47) uses assembly
        - INLINE ASM (lib/openzeppelin-contracts/contracts/cryptography/ECDSA.sol#40-44)
Address.isContract(address) (lib/openzeppelin-contracts/contracts/utils/Address.sol#26-35) uses assembly
        - INLINE ASM (lib/openzeppelin-contracts/contracts/utils/Address.sol#33)
Address._verifyCallResult(bool,bytes,string) (lib/openzeppelin-contracts/contracts/utils/Address.sol#171-188) uses assembly
        - INLINE ASM (lib/openzeppelin-contracts/contracts/utils/Address.sol#180-183)
LibAppStorage.diamondStorage() (contracts/libraries/LibAppStorage.sol#15-19) uses assembly
        - INLINE ASM (contracts/libraries/LibAppStorage.sol#16-18)
LibBytes.toUint8(bytes,uint256) (contracts/libraries/LibBytes.sol#18-28) uses assembly
        - INLINE ASM (contracts/libraries/LibBytes.sol#23-25)
LibBytes.toUint32(bytes,uint256) (contracts/libraries/LibBytes.sol#34-44) uses assembly
        - INLINE ASM (contracts/libraries/LibBytes.sol#39-41)
LibBytes.toUint256(bytes,uint256) (contracts/libraries/LibBytes.sol#50-60) uses assembly
        - INLINE ASM (contracts/libraries/LibBytes.sol#55-57)
LibDiamond.diamondStorage() (contracts/libraries/LibDiamond.sol#44-49) uses assembly
        - INLINE ASM (contracts/libraries/LibDiamond.sol#46-48)
LibDiamond.enforceHasContractCode(address,string) (contracts/libraries/LibDiamond.sol#234-240) uses assembly
        - INLINE ASM (contracts/libraries/LibDiamond.sol#236-238)
LibInternal.mow(address,address) (contracts/libraries/LibInternal.sol#19-35) uses assembly
        - INLINE ASM (contracts/libraries/LibInternal.sol#22-24)
LibPRBMath.mulDivFixedPoint(uint256,uint256) (contracts/libraries/LibPRBMath.sol#59-96) uses assembly
        - INLINE ASM (contracts/libraries/LibPRBMath.sol#62-66)
        - INLINE ASM (contracts/libraries/LibPRBMath.sol#74-77)
        - INLINE ASM (contracts/libraries/LibPRBMath.sol#84-95)
LibPRBMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/LibPRBMath.sol#185-263) uses assembly
        - INLINE ASM (contracts/libraries/LibPRBMath.sol#195-199)
        - INLINE ASM (contracts/libraries/LibPRBMath.sol#215-222)
        - INLINE ASM (contracts/libraries/LibPRBMath.sol#229-238)
LibTokenSilo.beanDenominatedValue(address,uint256) (contracts/libraries/Silo/LibTokenSilo.sol#274-300) uses assembly
        - INLINE ASM (contracts/libraries/Silo/LibTokenSilo.sol#292-294)
        - INLINE ASM (contracts/libraries/Silo/LibTokenSilo.sol#297-299)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity are used:
        - Version used: ['=0.7.6', '>=0.6.0<0.8.0', '>=0.6.2<0.8.0']
        - >=0.6.0<0.8.0 (lib/openzeppelin-contracts/contracts/cryptography/ECDSA.sol#3)
```

protocol/contracts/beanstalk/silo/BDVFacet.sol

```
LibCurve.getY(uint256,uint256[2],uint256,uint256) (contracts/libraries/Curve/LibCurve.sol#49-81) performs a multiplication on the result of a division:
        -c = (c * D) / (_x * N_COINS) (contracts/libraries/Curve/LibCurve.sol#67)
        -c = (c * D * A_PRECISION) / (Ann * N_COINS) (contracts/libraries/Curve/LibCurve.sol#70)
LibCurve.getD(uint256[2],uint256) (contracts/libraries/Curve/LibCurve.sol#83-112) performs a multiplication on the result of a division:
        -D_P = (D_P * D) / (xp[_j] * N_COINS) (contracts/libraries/Curve/LibCurve.sol#101)
        -D = (((Ann * S) / A_PRECISION + D_P * N_COINS) * D) / (((Ann - A_PRECISION) * D) / A_PRECISION + (N_COINS + 1) * D_P) (contracts/libraries/Curve/LibCurve.sol#104-106)
LibCurve.getYD(uint256,uint256[2],uint256,uint256) (contracts/libraries/Curve/LibCurve.sol#114-145) performs a multiplication on the result of a division:
        -c = (c * D) / (_x * N_COINS) (contracts/libraries/Curve/LibCurve.sol#131)
        -c = (c * D * A_PRECISION) / (Ann * N_COINS) (contracts/libraries/Curve/LibCurve.sol#134)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

LibCurve.getY(uint256,uint256[2],uint256,uint256)._i_scope_0 (contracts/libraries/Curve/LibCurve.sol#74) is a local variable never initialized
LibCurve.getYD(uint256,uint256,uint256[2],uint256)._i_scope_0 (contracts/libraries/Curve/LibCurve.sol#138) is a local variable never initialized
LibCurve.getD(uint256[2],uint256)._j (contracts/libraries/Curve/LibCurve.sol#100) is a local variable never initialized
LibCurve.getD(uint256[2],uint256)._i (contracts/libraries/Curve/LibCurve.sol#91) is a local variable never initialized
LibCurve.getD(uint256[2],uint256)._i_scope_0 (contracts/libraries/Curve/LibCurve.sol#98) is a local variable never initialized
LibCurve.getYD(uint256,uint256,uint256[2],uint256)._i (contracts/libraries/Curve/LibCurve.sol#127) is a local variable never initialized
LibCurve.getY(uint256,uint256[2],uint256,uint256)._i (contracts/libraries/Curve/LibCurve.sol#62) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

LibAppStorage.diamondStorage() (contracts/libraries/LibAppStorage.sol#15-19) uses assembly
        - INLINE ASM (contracts/libraries/LibAppStorage.sol#16-18)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity are used:
        - Version used: ['=0.7.6', '>=0.6.0<0.8.0']
        - >=0.6.0<0.8.0 (lib/openzeppelin-contracts/contracts/math/SafeMath.sol#3)
        - >=0.6.0<0.8.0 (lib/openzeppelin-contracts/contracts/token/ERC20/IERC20.sol#3)
        - >=0.6.0<0.8.0 (lib/openzeppelin-contracts/contracts/utils/Counters.sol#3)
        - =0.7.6 (contracts/C.sol#3)
        - ABIEncoderV2 (contracts/C.sol#4)
        - =0.7.6 (contracts/beanstalk/AppStorage.sol#3)
        - ABIEncoderV2 (contracts/beanstalk/AppStorage.sol#4)
        - =0.7.6 (contracts/beanstalk/silo/BDVFacet.sol#5)
        - ABIEncoderV2 (contracts/beanstalk/silo/BDVFacet.sol#6)
        - =0.7.6 (contracts/interfaces/IBean.sol#3)
        - ABIEncoderV2 (contracts/interfaces/IBean.sol#4)
        - ABIEncoderV2 (contracts/interfaces/ICurve.sol#2)
        - =0.7.6 (contracts/interfaces/ICurve.sol#3)
        - ABIEncoderV2 (contracts/interfaces/IDiamondCut.sol#2)
        - =0.7.6 (contracts/interfaces/IDiamondCut.sol#3)
        - ABIEncoderV2 (contracts/interfaces/IFertilizer.sol#2)
        - =0.7.6 (contracts/interfaces/IFertilizer.sol#3)
        - ABIEncoderV2 (contracts/interfaces/IProxyAdmin.sol#2)
        - =0.7.6 (contracts/interfaces/IProxyAdmin.sol#3)
        - =0.7.6 (contracts/libraries/Curve/LibBeanMetaCurve.sol#3)
        - ABIEncoderV2 (contracts/libraries/Curve/LibBeanMetaCurve.sol#4)
        - =0.7.6 (contracts/libraries/Curve/LibCurve.sol#3)
        - ABIEncoderV2 (contracts/libraries/Curve/LibCurve.sol#4)
        - =0.7.6 (contracts/libraries/Curve/LibMetaCurve.sol#3)
        - ABIEncoderV2 (contracts/libraries/Curve/LibMetaCurve.sol#4)
        - =0.7.6 (contracts/libraries/Decimal.sol#5)
        - ABIEncoderV2 (contracts/libraries/Decimal.sol#6)
        - =0.7.6 (contracts/libraries/LibAppStorage.sol#3)
        - ABIEncoderV2 (contracts/libraries/LibAppStorage.sol#4)
        - =0.7.6 (contracts/libraries/LibUnripe.sol#3)
        - ABIEncoderV2 (contracts/libraries/LibUnripe.sol#4)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

C.UniV3EthUsdc() (contracts/C.sol#179-181) is never used and should be removed
C.bean() (contracts/C.sol#147-149) is never used and should be removed
C.beanAddress() (contracts/C.sol#103-105) is never used and should be removed
C.curve3PoolAddress() (contracts/C.sol#171-173) is never used and should be removed
C.curveMetapoolAddress() (contracts/C.sol#107-109) is never used and should be removed
C.curveZap() (contracts/C.sol#163-165) is never used and should be removed
C.curveZapAddress() (contracts/C.sol#167-169) is never used and should be removed
C.dollarPerUnripeLP() (contracts/C.sol#207-209) is never used and should be removed
C.exploitAddLPRatio() (contracts/C.sol#211-213) is never used and should be removed
C.fertilizer() (contracts/C.sol#183-185) is never used and should be removed
C.fertilizerAddress() (contracts/C.sol#187-189) is never used and should be removed
C.fertilizerAdmin() (contracts/C.sol#191-193) is never used and should be removed
C.getBlockLengthSeconds() (contracts/C.sol#79-81) is never used and should be removed
C.getChainId() (contracts/C.sol#83-85) is never used and should be removed
```

protocol/contracts/beanstalk/silo/ConvertFacet.sol

```
ConvertFacet.enrootDeposits(address,int96[],uint256[]) (contracts/beanstalk/silo/ConvertFacet.sol#179-222) performs a multiplication on the result of a division:
        -bdv = amounts[i].mul(newBDV).div(ar.tokensRemoved) (contracts/beanstalk/silo/ConvertFacet.sol#196)
        -newStalk = newStalk.add(bdv.mul(_stalkPerBdv).add(LibSilo.stalkReward(stems[i],_lastStem,uint128(bdv)))) (contracts/beanstalk/silo/ConvertFacet.sol#206-214)
LibCurve.getY(uint256,uint256[2],uint256,uint256) (contracts/libraries/Curve/LibCurve.sol#49-81) performs a multiplication on the result of a division:
        -c = (c * D) / (_x * N_COINS) (contracts/libraries/Curve/LibCurve.sol#67)
        -c = (c * D * A_PRECISION) / (Ann * N_COINS) (contracts/libraries/Curve/LibCurve.sol#70)
LibCurve.getD(uint256[2],uint256) (contracts/libraries/Curve/LibCurve.sol#83-112) performs a multiplication on the result of a division:
        -D_P = (D_P * D) / (xp[_j] * N_COINS) (contracts/libraries/Curve/LibCurve.sol#101)
        -D = (((Ann * S) / A_PRECISION + D_P * N_COINS) * D) / (((Ann - A_PRECISION) * D) / A_PRECISION + (N_COINS + 1) * D_P) (contracts/libraries/Curve/LibCurve.sol#104-106)
LibCurve.getYD(uint256,uint256,uint256[2],uint256) (contracts/libraries/Curve/LibCurve.sol#114-145) performs a multiplication on the result of a division:
        -c = (c * D) / (_x * N_COINS) (contracts/libraries/Curve/LibCurve.sol#131)
        -c = (c * D * A_PRECISION) / (Ann * N_COINS) (contracts/libraries/Curve/LibCurve.sol#134)
LibPRBMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/LibPRBMath.sol#185-263) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/LibPRBMath.sol#231)
        -inverse = (3 * denominator) ^ 2 (contracts/libraries/LibPRBMath.sol#246)
LibPRBMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/LibPRBMath.sol#185-263) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/LibPRBMath.sol#231)
        -inverse *= 2 - denominator * inverse (contracts/libraries/LibPRBMath.sol#250)
LibPRBMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/LibPRBMath.sol#185-263) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/LibPRBMath.sol#231)
        -inverse *= 2 - denominator * inverse (contracts/libraries/LibPRBMath.sol#251)
LibPRBMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/LibPRBMath.sol#185-263) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/LibPRBMath.sol#231)
        -inverse *= 2 - denominator * inverse (contracts/libraries/LibPRBMath.sol#252)
LibPRBMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/LibPRBMath.sol#185-263) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/LibPRBMath.sol#231)
        -inverse *= 2 - denominator * inverse (contracts/libraries/LibPRBMath.sol#253)
LibPRBMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/LibPRBMath.sol#185-263) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/LibPRBMath.sol#231)
        -inverse *= 2 - denominator * inverse (contracts/libraries/LibPRBMath.sol#254)
LibPRBMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/LibPRBMath.sol#185-263) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/LibPRBMath.sol#231)
        -inverse *= 2 - denominator * inverse (contracts/libraries/LibPRBMath.sol#255)
LibPRBMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/LibPRBMath.sol#185-263) performs a multiplication on the result of a division:
        -prod0 = prod0 / twos (contracts/libraries/LibPRBMath.sol#234)
        -result = prod0 * inverse (contracts/libraries/LibPRBMath.sol#261)
LibUnripeSilo._removeUnripeLPDeposit(address,uint32,uint256) (contracts/libraries/Silo/LibUnripeSilo.sol#212-289) performs a multiplication on the result of a division:
        -removed = amount.mul(bdv1).div(amount1) (contracts/libraries/Silo/LibUnripeSilo.sol#230)
        -s.a[account].lp.depositSeeds[season] = s.a[account].lp.depositSeeds[season].sub(removed.mul(4)) (contracts/libraries/Silo/LibUnripeSilo.sol#231-235)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

Reentrancy in Silo._claimPlenty(address) (contracts/beanstalk/silo/SiloFacet/Silo.sol#147-154):
        External calls:
        - C.threeCrv().safeTransfer(account,plenty) (contracts/beanstalk/silo/SiloFacet/Silo.sol#150)
        State variables written after the call(s):
        - delete s.a[account].sop.plenty (contracts/beanstalk/silo/SiloFacet/Silo.sol#151)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

LibCurve.getD(uint256[2],uint256)._i_scope_0 (contracts/libraries/Curve/LibCurve.sol#98) is a local variable never initialized
LibLegacyTokenSilo._mowAndMigrate(address,address[],uint32[][],uint256[][]).perTokenData (contracts/libraries/Silo/LibLegacyTokenSilo.sol#371) is a local variable never initialized
LibCurve.getYD(uint256,uint256,uint256[2],uint256)._i (contracts/libraries/Curve/LibCurve.sol#127) is a local variable never initialized
LibCurve.getY(uint256,uint256[2],uint256,uint256)._i (contracts/libraries/Curve/LibCurve.sol#62) is a local variable never initialized
LibCurve.getD(uint256[2],uint256)._i (contracts/libraries/Curve/LibCurve.sol#91) is a local variable never initialized
LibDiamond.removeFunctions(address,bytes4[]).selectorIndex (contracts/libraries/LibDiamond.sol#162) is a local variable never initialized
LibSilo._removeDepositsFromAccount(address,address,int96[],uint256[]).i (contracts/libraries/Silo/LibSilo.sol#555) is a local variable never initialized
ConvertFacet.enrootDeposits(address,int96[],uint256[]).i (contracts/beanstalk/silo/ConvertFacet.sol#195) is a local variable never initialized
LibDiamond.replaceFunctions(address,bytes4[]).selectorIndex (contracts/libraries/LibDiamond.sol#147) is a local variable never initialized
LibCurve.getYD(uint256,uint256,uint256[2],uint256)._i_scope_0 (contracts/libraries/Curve/LibCurve.sol#138) is a local variable never initialized
ConvertFacet._withdrawTokens(address,int96[],uint256[],uint256).a (contracts/beanstalk/silo/ConvertFacet.sol#234) is a local variable never initialized
LibDiamond.addFunctions(address,bytes4[]).selectorIndex (contracts/libraries/LibDiamond.sol#129) is a local variable never initialized
LibLegacyTokenSilo._claimWithdrawals(address,address,uint32[]).i (contracts/libraries/Silo/LibLegacyTokenSilo.sol#587) is a local variable never initialized
LibLegacyTokenSilo._mowAndMigrate(address,address[],uint32[][],uint256[][]).migrateData (contracts/libraries/Silo/LibLegacyTokenSilo.sol#367) is a local variable never initialized
LibCurve.getD(uint256[2],uint256)._j (contracts/libraries/Curve/LibCurve.sol#100) is a local variable never initialized
LibDiamond.diamondCut(IDiamondCut.FacetCut[],address,bytes).facetIndex (contracts/libraries/LibDiamond.sol#184) is a local variable never initialized
LibCurve.getY(uint256,uint256[2],uint256,uint256)._i_scope_0 (contracts/libraries/Curve/LibCurve.sol#74) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

LibUnripeSilo.unripeLPDeposit(address,uint32).bdv (contracts/libraries/Silo/LibUnripeSilo.sol#306) is written in both
        (amount,bdv) = getBeanEthUnripeLP(account,season) (contracts/libraries/Silo/LibUnripeSilo.sol#312)
        bdv = uint256(s.a[account].legacyDeposits[C.unripeLPAddress()][season].bdv).add(legBdv) (contracts/libraries/Silo/LibUnripeSilo.sol#329-331)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#write-after-write
```

protocol/contracts/beanstalk/silo/MigrationFacet.sol

```
LibCurve.getD(uint256[2],uint256) (contracts/libraries/Curve/LibCurve.sol#83-112) performs a multiplication on the result of a division:
        -D_P = (D_P * D) / (xp[_j] * N_COINS) (contracts/libraries/Curve/LibCurve.sol#101)
        -D = (((Ann * S) / A_PRECISION + D_P * N_COINS) * D) / (((Ann - A_PRECISION) * D) / A_PRECISION + (N_COINS + 1) * D_P) (contracts/libraries/Curve/LibCurve.sol#104-106)
LibCurve.getYD(uint256,uint256,uint256[2],uint256) (contracts/libraries/Curve/LibCurve.sol#114-145) performs a multiplication on the result of a division:
        -c = (c * D) / (_x * N_COINS) (contracts/libraries/Curve/LibCurve.sol#131)
        -c = (c * D * A_PRECISION) / (Ann * N_COINS) (contracts/libraries/Curve/LibCurve.sol#134)
LibPRBMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/LibPRBMath.sol#185-263) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/LibPRBMath.sol#231)
        -inverse = (3 * denominator) ^ 2 (contracts/libraries/LibPRBMath.sol#246)
LibPRBMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/LibPRBMath.sol#185-263) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/LibPRBMath.sol#231)
        -inverse *= 2 - denominator * inverse (contracts/libraries/LibPRBMath.sol#250)
LibPRBMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/LibPRBMath.sol#185-263) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/LibPRBMath.sol#231)
        -inverse *= 2 - denominator * inverse (contracts/libraries/LibPRBMath.sol#251)
LibPRBMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/LibPRBMath.sol#185-263) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/LibPRBMath.sol#231)
        -inverse *= 2 - denominator * inverse (contracts/libraries/LibPRBMath.sol#252)
LibPRBMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/LibPRBMath.sol#185-263) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/LibPRBMath.sol#231)
        -inverse *= 2 - denominator * inverse (contracts/libraries/LibPRBMath.sol#253)
LibPRBMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/LibPRBMath.sol#185-263) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/LibPRBMath.sol#231)
        -inverse *= 2 - denominator * inverse (contracts/libraries/LibPRBMath.sol#254)
LibPRBMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/LibPRBMath.sol#185-263) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/LibPRBMath.sol#231)
        -inverse *= 2 - denominator * inverse (contracts/libraries/LibPRBMath.sol#255)
LibPRBMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/LibPRBMath.sol#185-263) performs a multiplication on the result of a division:
        -prod0 = prod0 / twos (contracts/libraries/LibPRBMath.sol#234)
        -result = prod0 * inverse (contracts/libraries/LibPRBMath.sol#261)
LibUnripeSilo._removeUnripeLPDeposit(address,uint32,uint256) (contracts/libraries/Silo/LibUnripeSilo.sol#212-289) performs a multiplication on the result of a division:
        -removed = amount.mul(bdv1).div(amount1) (contracts/libraries/Silo/LibUnripeSilo.sol#230)
        -s.a[account].lp.depositSeeds[season] = s.a[account].lp.depositSeeds[season].sub(removed.mul(4)) (contracts/libraries/Silo/LibUnripeSilo.sol#231-235)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

Contract locking ether found:
        Contract MigrationFacet (contracts/beanstalk/silo/MigrationFacet.sol#23-69) has payable functions:
         - MigrationFacet.mowAndMigrate(address,address[],uint32[][],uint256[][],uint256,uint256,bytes32[]) (contracts/beanstalk/silo/MigrationFacet.sol#42-53)
         - MigrationFacet.mowAndMigrateNoDeposits(address) (contracts/beanstalk/silo/MigrationFacet.sol#63-65)
        But does not have a function to withdraw the ether
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#contracts-that-lock-ether

Reentrancy in Silo._claimPlenty(address) (contracts/beanstalk/silo/SiloFacet/Silo.sol#147-154):
        External calls:
        - C.threeCrv().safeTransfer(account,plenty) (contracts/beanstalk/silo/SiloFacet/Silo.sol#150)
        State variables written after the call(s):
        - delete s.a[account].sop.plenty (contracts/beanstalk/silo/SiloFacet/Silo.sol#151)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

LibCurve.getY(uint256,uint256[2],uint256,uint256)._i (contracts/libraries/Curve/LibCurve.sol#62) is a local variable never initialized
LibLegacyTokenSilo._claimWithdrawals(address,address,uint32[])._i (contracts/libraries/Silo/LibLegacyTokenSilo.sol#587) is a local variable never initialized
LibDiamond.removeFunctions(address,bytes4[]).selectorIndex (contracts/libraries/LibDiamond.sol#162) is a local variable never initialized
LibDiamond.addFunctions(address,bytes4[]).selectorIndex (contracts/libraries/LibDiamond.sol#129) is a local variable never initialized
LibCurve.getD(uint256[2],uint256)._i (contracts/libraries/Curve/LibCurve.sol#91) is a local variable never initialized
LibSilo._removeDepositsFromAccount(address,address,int96[],uint256[]).i (contracts/libraries/Silo/LibSilo.sol#555) is a local variable never initialized
LibCurve.getYD(uint256,uint256,uint256[2],uint256)._i (contracts/libraries/Curve/LibCurve.sol#127) is a local variable never initialized
LibCurve.getY(uint256,uint256[2],uint256,uint256)._i_scope_0 (contracts/libraries/Curve/LibCurve.sol#74) is a local variable never initialized
TokenSilo._transferDeposits(address,address,address,int96[],uint256[]).i (contracts/beanstalk/silo/SiloFacet/TokenSilo.sol#335) is a local variable never initialized
LibCurve.getD(uint256[2],uint256)._j (contracts/libraries/Curve/LibCurve.sol#100) is a local variable never initialized
LibCurve.getD(uint256[2],uint256)._i_scope_0 (contracts/libraries/Curve/LibCurve.sol#98) is a local variable never initialized
LibCurve.getYD(uint256,uint256,uint256[2],uint256)._i_scope_0 (contracts/libraries/Curve/LibCurve.sol#138) is a local variable never initialized
LibDiamond.replaceFunctions(address,bytes4[]).selectorIndex (contracts/libraries/LibDiamond.sol#147) is a local variable never initialized
LibLegacyTokenSilo._mowAndMigrate(address,address[],uint32[][],uint256[][]).perTokenData (contracts/libraries/Silo/LibLegacyTokenSilo.sol#371) is a local variable never initialized
LibDiamond.diamondCut(IDiamondCut.FacetCut[],address,bytes).facetIndex (contracts/libraries/LibDiamond.sol#184) is a local variable never initialized
LibLegacyTokenSilo._mowAndMigrate(address,address[],uint32[][],uint256[][]).migrateData (contracts/libraries/Silo/LibLegacyTokenSilo.sol#367) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

LibUnripeSilo.unripeLPDeposit(address,uint32).bdv (contracts/libraries/Silo/LibUnripeSilo.sol#306) is written in both
        (amount,bdv) = getBeanEthUnripeLP(account,season) (contracts/libraries/Silo/LibUnripeSilo.sol#312)
        bdv = uint256(s.a[account].legacyDeposits[C.unripeLPAddress()][season].bdv).add(legBdv) (contracts/libraries/Silo/LibUnripeSilo.sol#329-331)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#write-after-write
```

protocol/contracts/beanstalk/silo/WhitelistFacet.sol

```
LibPRBMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/LibPRBMath.sol#185-263) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/LibPRBMath.sol#231)
        -inverse = (3 * denominator) ^ 2 (contracts/libraries/LibPRBMath.sol#246)
LibPRBMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/LibPRBMath.sol#185-263) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/LibPRBMath.sol#231)
        -inverse *= 2 - denominator * inverse (contracts/libraries/LibPRBMath.sol#250)
LibPRBMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/LibPRBMath.sol#185-263) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/LibPRBMath.sol#231)
        -inverse *= 2 - denominator * inverse (contracts/libraries/LibPRBMath.sol#251)
LibPRBMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/LibPRBMath.sol#185-263) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/LibPRBMath.sol#231)
        -inverse *= 2 - denominator * inverse (contracts/libraries/LibPRBMath.sol#252)
LibPRBMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/LibPRBMath.sol#185-263) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/LibPRBMath.sol#231)
        -inverse *= 2 - denominator * inverse (contracts/libraries/LibPRBMath.sol#253)
LibPRBMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/LibPRBMath.sol#185-263) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/LibPRBMath.sol#231)
        -inverse *= 2 - denominator * inverse (contracts/libraries/LibPRBMath.sol#254)
LibPRBMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/LibPRBMath.sol#185-263) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/LibPRBMath.sol#231)
        -inverse *= 2 - denominator * inverse (contracts/libraries/LibPRBMath.sol#255)
LibPRBMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/LibPRBMath.sol#185-263) performs a multiplication on the result of a division:
        -prod0 = prod0 / twos (contracts/libraries/LibPRBMath.sol#234)
        -result = prod0 * inverse (contracts/libraries/LibPRBMath.sol#261)
LibUnripeSilo._removeUnripeLPDeposit(address,uint32,uint256) (contracts/libraries/Silo/LibUnripeSilo.sol#212-289) performs a multiplication on the result of a division:
        -removed = amount.mul(bdv1).div(amount1) (contracts/libraries/Silo/LibUnripeSilo.sol#230)
        -s.a[account].lp.depositSeeds[season] = s.a[account].lp.depositSeeds[season].sub(removed.mul(4)) (contracts/libraries/Silo/LibUnripeSilo.sol#231-235)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

Contract locking ether found:
        Contract WhitelistFacet (contracts/beanstalk/silo/WhitelistFacet.sol#16-62) has payable functions:
         - WhitelistFacet.dewhitelistToken(address) (contracts/beanstalk/silo/WhitelistFacet.sol#32-35)
         - WhitelistFacet.whitelistToken(address,bytes4,uint32,uint32) (contracts/beanstalk/silo/WhitelistFacet.sol#37-50)
         - WhitelistFacet.updateStalkPerBdvPerSeasonForToken(address,uint32) (contracts/beanstalk/silo/WhitelistFacet.sol#52-61)
        But does not have a function to withdraw the ether
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#contracts-that-lock-ether

LibDiamond.addFunctions(address,bytes4[]).selectorIndex (contracts/libraries/LibDiamond.sol#129) is a local variable never initialized
LibDiamond.diamondCut(IDiamondCut.FacetCut[],address,bytes).facetIndex (contracts/libraries/LibDiamond.sol#184) is a local variable never initialized
LibDiamond.removeFunctions(address,bytes4[]).selectorIndex (contracts/libraries/LibDiamond.sol#162) is a local variable never initialized
LibLegacyTokenSilo._claimWithdrawals(address,address,uint32[]).i (contracts/libraries/Silo/LibLegacyTokenSilo.sol#587) is a local variable never initialized
LibDiamond.replaceFunctions(address,bytes4[]).selectorIndex (contracts/libraries/LibDiamond.sol#147) is a local variable never initialized
LibLegacyTokenSilo._mowAndMigrate(address,address[],uint32[][],uint256[][]).perTokenData (contracts/libraries/Silo/LibLegacyTokenSilo.sol#371) is a local variable never initialized
LibSilo._removeDepositsFromAccount(address,address,int96[],uint256[]).i (contracts/libraries/Silo/LibSilo.sol#555) is a local variable never initialized
LibLegacyTokenSilo._mowAndMigrate(address,address[],uint32[][],uint256[][]).migrateData (contracts/libraries/Silo/LibLegacyTokenSilo.sol#367) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

LibUnripeSilo.unripeLPDeposit(address,uint32).bdv (contracts/libraries/Silo/LibUnripeSilo.sol#306) is written in both
        (amount,bdv) = getBeanEthUnripeLP(account,season) (contracts/libraries/Silo/LibUnripeSilo.sol#312)
        bdv = uint256(s.a[account].legacyDeposits[C.unripeLPAddress()][season].bdv).add(legBdv) (contracts/libraries/Silo/LibUnripeSilo.sol#329-331)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#write-after-write

Variable 'LibLegacyTokenSilo._mowAndMigrate(address,address[],uint32[][],uint256[][]).perDepositData (contracts/libraries/Silo/LibLegacyTokenSilo.sol#376)' in LibLegacyTokenSilo._mowAndMigrate(address,address[],uin
/libraries/Silo/LibLegacyTokenSilo.sol#345-434) potentially used before declaration: j < seasons[i].length (contracts/libraries/Silo/LibLegacyTokenSilo.sol#375)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

Reentrancy in LibTokenSilo.deposit(address,address,int96,uint256) (contracts/libraries/Silo/LibTokenSilo.sol#108-116):
        External calls:
        - bdv = beanDenominatedValue(token,amount) (contracts/libraries/Silo/LibTokenSilo.sol#114)
                - (success,data) = address(this).call(callData) (contracts/libraries/Silo/LibTokenSilo.sol#286-288)
        Event emitted after the call(s):
        - AddDeposit(account,token,stem,amount,bdv) (contracts/libraries/Silo/LibTokenSilo.sol#203)
                - depositWithBDV(account,token,stem,amount,bdv) (contracts/libraries/Silo/LibTokenSilo.sol#115)
        - TransferSingle(msg.sender,address(0),account,uint256(depositId),amount) (contracts/libraries/Silo/LibTokenSilo.sol#195-201)
                - depositWithBDV(account,token,stem,amount,bdv) (contracts/libraries/Silo/LibTokenSilo.sol#115)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

LibAppStorage.diamondStorage() (contracts/libraries/LibAppStorage.sol#15-19) uses assembly
        - INLINE ASM (contracts/libraries/LibAppStorage.sol#16-18)
LibBytes.toUint8(bytes,uint256) (contracts/libraries/LibBytes.sol#18-28) uses assembly
        - INLINE ASM (contracts/libraries/LibBytes.sol#23-25)
LibBytes.toUint32(bytes,uint256) (contracts/libraries/LibBytes.sol#34-44) uses assembly
        - INLINE ASM (contracts/libraries/LibBytes.sol#39-41)
```

protocol/contracts/beanstalk/silo/SiloFacet/SiloFacet.sol

```
LibPRBMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/LibPRBMath.sol#185-263) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/LibPRBMath.sol#231)
        -inverse = (3 * denominator) ^ 2 (contracts/libraries/LibPRBMath.sol#246)
LibPRBMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/LibPRBMath.sol#185-263) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/LibPRBMath.sol#231)
        -inverse *= 2 - denominator * inverse (contracts/libraries/LibPRBMath.sol#250)
LibPRBMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/LibPRBMath.sol#185-263) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/LibPRBMath.sol#231)
        -inverse *= 2 - denominator * inverse (contracts/libraries/LibPRBMath.sol#251)
LibPRBMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/LibPRBMath.sol#185-263) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/LibPRBMath.sol#231)
        -inverse *= 2 - denominator * inverse (contracts/libraries/LibPRBMath.sol#252)
LibPRBMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/LibPRBMath.sol#185-263) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/LibPRBMath.sol#231)
        -inverse *= 2 - denominator * inverse (contracts/libraries/LibPRBMath.sol#253)
LibPRBMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/LibPRBMath.sol#185-263) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/LibPRBMath.sol#231)
        -inverse *= 2 - denominator * inverse (contracts/libraries/LibPRBMath.sol#254)
LibPRBMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/LibPRBMath.sol#185-263) performs a multiplication on the result of a division:
        -denominator = denominator / twos (contracts/libraries/LibPRBMath.sol#231)
        -inverse *= 2 - denominator * inverse (contracts/libraries/LibPRBMath.sol#255)
LibPRBMath.mulDiv(uint256,uint256,uint256) (contracts/libraries/LibPRBMath.sol#185-263) performs a multiplication on the result of a division:
        -prod0 = prod0 / twos (contracts/libraries/LibPRBMath.sol#234)
        -result = prod0 * inverse (contracts/libraries/LibPRBMath.sol#261)
LibUnripeSilo._removeUnripeLPDeposit(address,uint32,uint256) (contracts/libraries/Silo/LibUnripeSilo.sol#212-289) performs a multiplication on the result of a division:
        -removed = amount.mul(bdv1).div(amount1) (contracts/libraries/Silo/LibUnripeSilo.sol#230)
        -s.a[account].lp.depositSeeds[season] = s.a[account].lp.depositSeeds[season].sub(removed.mul(4)) (contracts/libraries/Silo/LibUnripeSilo.sol#231-235)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

LibTransfer.sendToken(IERC20,uint256,address,LibTransfer.To) (contracts/libraries/Token/LibTransfer.sol#73-83) uses a dangerous strict equality:
        - amount == 0 (contracts/libraries/Token/LibTransfer.sol#79)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in Silo._claimPlenty(address) (contracts/beanstalk/silo/SiloFacet/Silo.sol#147-154):
        External calls:
        - C.threeCrv().safeTransfer(account,plenty) (contracts/beanstalk/silo/SiloFacet/Silo.sol#150)
        State variables written after the call(s):
        - delete s.a[account].sop.plenty (contracts/beanstalk/silo/SiloFacet/Silo.sol#151)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

SiloFacet.mowMultiple(address,address[]).i (contracts/beanstalk/silo/SiloFacet/SiloFacet.sol#270) is a local variable never initialized
LibSilo._removeDepositsFromAccount(address,address,int96[],uint256[]).i (contracts/libraries/Silo/LibSilo.sol#555) is a local variable never initialized
LibLegacyTokenSilo._claimWithdrawals(address,address,uint32[]).i (contracts/libraries/Silo/LibLegacyTokenSilo.sol#587) is a local variable never initialized
LibLegacyTokenSilo._mowAndMigrate(address,address[],uint32[][],uint256[][]).perTokenData (contracts/libraries/Silo/LibLegacyTokenSilo.sol#371) is a local variable never initialized
SiloFacet.safeBatchTransferFrom(address,address,uint256[],uint256[],bytes).i (contracts/beanstalk/silo/SiloFacet/SiloFacet.sol#243) is a local variable never initialized
TokenSilo._transferDeposits(address,address,address,int96[],uint256[]).i (contracts/beanstalk/silo/SiloFacet/TokenSilo.sol#335) is a local variable never initialized
LibLegacyTokenSilo._mowAndMigrate(address,address[],uint32[][],uint256[][]).migrateData (contracts/libraries/Silo/LibLegacyTokenSilo.sol#367) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

LibUnripeSilo.unripeLPDeposit(address,uint32).bdv (contracts/libraries/Silo/LibUnripeSilo.sol#306) is written in both
        (amount,bdv) = getBeanEthUnripeLP(account,season) (contracts/libraries/Silo/LibUnripeSilo.sol#312)
        bdv = uint256(s.a[account].legacyDeposits[C.unripeLPAddress()][season].bdv).add(legBdv) (contracts/libraries/Silo/LibUnripeSilo.sol#329-331)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#write-after-write

Variable 'LibLegacyTokenSilo._mowAndMigrate(address,address[],uint32[][],uint256[][]).perDepositData (contracts/libraries/Silo/LibLegacyTokenSilo.sol#376)' in LibLegacyTokenSilo._mowAndMigrate(address,address[],uin
/libraries/Silo/LibLegacyTokenSilo.sol#345-434) potentially used before declaration: j < seasons[i].length (contracts/libraries/Silo/LibLegacyTokenSilo.sol#375)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

Reentrancy in Silo._claimPlenty(address) (contracts/beanstalk/silo/SiloFacet/Silo.sol#147-154):
        External calls:
        - C.threeCrv().safeTransfer(account,plenty) (contracts/beanstalk/silo/SiloFacet/Silo.sol#150)
        Event emitted after the call(s):
        - ClaimPlenty(account,plenty) (contracts/beanstalk/silo/SiloFacet/Silo.sol#153)
Reentrancy in SiloFacet.deposit(address,uint256,LibTransfer.From) (contracts/beanstalk/silo/SiloFacet/SiloFacet.sol#50-68):
        External calls:
        - amount = LibTransfer.receiveToken(IERC20(token),_amount,msg.sender,mode) (contracts/beanstalk/silo/SiloFacet/SiloFacet.sol#61-66)
        - (bdv,stem) = _deposit(msg.sender,token,amount) (contracts/beanstalk/silo/SiloFacet/SiloFacet.sol#67)
                - stalk = LibTokenSilo.deposit(account,token,stem = LibTokenSilo.stemTipForToken(token),amount) (contracts/beanstalk/silo/SiloFacet/TokenSilo.sol#162-167)
                - (success,data) = address(this).call(callData) (contracts/libraries/Silo/LibTokenSilo.sol#286-288)
        Event emitted after the call(s):
        - AddDeposit(account,token,stem,amount,bdv) (contracts/libraries/Silo/LibTokenSilo.sol#203)
                - (bdv,stem) = _deposit(msg.sender,token,amount) (contracts/beanstalk/silo/SiloFacet/SiloFacet.sol#67)
        - StalkBalanceChanged(account,int256(stalk),int256(roots)) (contracts/libraries/Silo/LibSilo.sol#164)
```

protocol/contracts/beanstalk/AppStorage.sol

```
Different versions of Solidity are used:
        - Version used: ['=0.7.6', '>=0.6.0<0.8.0']
        - >=0.6.0<0.8.0 (lib/openzeppelin-contracts/contracts/math/SafeMath.sol#3)
        - >=0.6.0<0.8.0 (lib/openzeppelin-contracts/contracts/token/ERC20/IERC20.sol#3)
        - >=0.6.0<0.8.0 (lib/openzeppelin-contracts/contracts/utils/Counters.sol#3)
        - =0.7.6 (contracts/beanstalk/AppStorage.sol#3)
        - ABIEncoderV2 (contracts/beanstalk/AppStorage.sol#4)
        - ABIEncoderV2 (contracts/interfaces/IDiamondCut.sol#2)
        - =0.7.6 (contracts/interfaces/IDiamondCut.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Counters.current(Counters.Counter) (lib/openzeppelin-contracts/contracts/utils/Counters.sol#28-30) is never used and should be removed
Counters.decrement(Counters.Counter) (lib/openzeppelin-contracts/contracts/utils/Counters.sol#37-39) is never used and should be removed
Counters.increment(Counters.Counter) (lib/openzeppelin-contracts/contracts/utils/Counters.sol#32-35) is never used and should be removed
SafeMath.add(uint256,uint256) (lib/openzeppelin-contracts/contracts/math/SafeMath.sol#85-89) is never used and should be removed
SafeMath.div(uint256,uint256) (lib/openzeppelin-contracts/contracts/math/SafeMath.sol#135-138) is never used and should be removed
SafeMath.div(uint256,uint256,string) (lib/openzeppelin-contracts/contracts/math/SafeMath.sol#190-193) is never used and should be removed
SafeMath.mod(uint256,uint256) (lib/openzeppelin-contracts/contracts/math/SafeMath.sol#152-155) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (lib/openzeppelin-contracts/contracts/math/SafeMath.sol#210-213) is never used and should be removed
SafeMath.mul(uint256,uint256) (lib/openzeppelin-contracts/contracts/math/SafeMath.sol#116-121) is never used and should be removed
SafeMath.sub(uint256,uint256) (lib/openzeppelin-contracts/contracts/math/SafeMath.sol#101-104) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (lib/openzeppelin-contracts/contracts/math/SafeMath.sol#170-173) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (lib/openzeppelin-contracts/contracts/math/SafeMath.sol#24-28) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (lib/openzeppelin-contracts/contracts/math/SafeMath.sol#60-63) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (lib/openzeppelin-contracts/contracts/math/SafeMath.sol#70-73) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (lib/openzeppelin-contracts/contracts/math/SafeMath.sol#45-53) is never used and should be removed
SafeMath.trySub(uint256,uint256) (lib/openzeppelin-contracts/contracts/math/SafeMath.sol#35-38) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version>=0.6.0<0.8.0 (lib/openzeppelin-contracts/contracts/math/SafeMath.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (lib/openzeppelin-contracts/contracts/token/ERC20/IERC20.sol#3) is too complex
Pragma version>=0.6.0<0.8.0 (lib/openzeppelin-contracts/contracts/utils/Counters.sol#3) is too complex
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

- As a result of the tests carried out with the Slither tool, some results were obtained and reviewed by Halborn. Based on the results reviewed, some vulnerabilities were determined to be false positives. The actual vulnerabilities found by Slither are already included in the report findings.

7.2 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 all the contracts and sent the compiled results to the analyzers to locate any vulnerabilities.

MythX results:

Report for contracts/beanstalk/silo/SiloFacet/SiloExit.sol
https://dashboard.mythx.io/#/console/analyses/469834e9-5131-41db-bf06-d3f44374ed71
https://dashboard.mythx.io/#/console/analyses/3f667a62-6511-4d6b-b23e-a4689bedefe0
https://dashboard.mythx.io/#/console/analyses/ad57b6b0-5b2f-4254-a1e7-48b840b3d104
https://dashboard.mythx.io/#/console/analyses/f9b884bb-af59-440c-92ec-512ea3822cd1

Line	SWC Title	Severity	Short Description
171	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randonmness.

Report for contracts/beanstalk/silo/SiloFacet/SiloFacet.sol
https://dashboard.mythx.io/#/console/analyses/469834e9-5131-41db-bf06-d3f44374ed71

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

Report for contracts/libraries/Silo/LibSilo.sol
https://dashboard.mythx.io/#/console/analyses/f9b884bb-af59-440c-92ec-512ea3822cd1
https://dashboard.mythx.io/#/console/analyses/3a2d70c3-0bef-44c9-9b0d-20c968ababf9
https://dashboard.mythx.io/#/console/analyses/469834e9-5131-41db-bf06-d3f44374ed71
https://dashboard.mythx.io/#/console/analyses/ad57b6b0-5b2f-4254-a1e7-48b840b3d104
https://dashboard.mythx.io/#/console/analyses/3f667a62-6511-4d6b-b23e-a4689bedefe0

Line	SWC Title	Severity	Short Description
237	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randonmness.
612	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randonmness.

Report for contracts/libraries/Silo/LibTokenSilo.sol
https://dashboard.mythx.io/#/console/analyses/469834e9-5131-41db-bf06-d3f44374ed71

Line	SWC Title	Severity	Short Description
286	(SWC-123) Requirement Violation	Low	Requirement violation.

Report for contracts/libraries/LibInternal.sol
https://dashboard.mythx.io/#/console/analyses/ad57b6b0-5b2f-4254-a1e7-48b840b3d104
https://dashboard.mythx.io/#/console/analyses/3f667a62-6511-4d6b-b23e-a4689bedefe0
https://dashboard.mythx.io/#/console/analyses/f9b884bb-af59-440c-92ec-512ea3822cd1

Line	SWC Title	Severity	Short Description
20	(SWC-109) Uninitialized Storage Pointer	Medium	Dangerous use of uninitialized storage variables.

Report for contracts/beanstalk/silo/WhitelistFacet.sol
https://dashboard.mythx.io/#/console/analyses/3a2d70c3-0bef-44c9-9b0d-20c968ababf9

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

- No major issues found by Mythx.

THANK YOU FOR CHOOSING

// HALBORN