Substance Exchange - Exchange V1

Smart Contract Security Assessment

Prepared by: **Halborn**

Date of Engagement: **May 25th, 2023 - June 22nd, 2023**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	06/02/2023	Miguel Jalon
0.2	Document Updates	06/02/2023	Miguel Jalon
0.3	Document Updates	06/21/2023	Miguel Jalon
0.4	Draft Version	06/22/2023	Miguel Jalon
0.5	Draft Review	06/23/2023	Grzegorz Trawinski
0.6	Draft Review	06/23/2023	Manuel Garcia Diaz
0.7	Draft Review	06/23/2023	Piotr Cielas
0.8	Draft Review	06/25/2023	Gabi Urrutia
1.0	Remediation Plan	07/17/2023	Piotr Cielas
1.1	Remediation Plan Review	07/18/2023	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
Piotr Cielas	Halborn	Piotr.Cielas@halborn.com
Grzegorz Trawinski	Halborn	Grzegorz.Trawinski@halborn.com
Manuel Garcia Diaz	Halborn	Manuel.Diaz@halborn.com
Miguel Jalon	Halborn	Miguel.Jalon@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Substance Exchange is a Perpetual Decentralized Exchange where users can interact with futures and options and also can be Liquidity Providers earning from traders.

Substance Exchange engaged Halborn to conduct a security assessment on their smart contracts beginning on May 25th, 2023 and ending on June 22nd, 2023. The security assessment was scoped to the smart contracts provided in the Substance Exchange V3 GitHub repository. Commit hashes and further details can be found in the Scope section of this report.

1.2 ASSESSMENT SUMMARY

Halborn was provided 4 weeks for the engagement and assigned a full-time security engineer to review the security of the smart contracts in scope. The engineer is a blockchain and smart contract security expert with advanced penetration testing and smart contract hacking skills, and deep knowledge of multiple blockchain protocols.

The purpose of the assessment is to:

- Identify potential security issues within the smart contracts.
- Ensure that smart contract functionality operates as intended.

In summary, Halborn identified some improvements to reduce the likelihood and impact of risks, which were addressed and accepted by Substance Exchange . The most concerning issues were found in the Chainlink integration:

- Chainlink latestrounddata might be stale or incorrect
- Chainlink Arbitrum sequencer is not verified to be online

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of this assessment. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of the code and can quickly identify items that do not follow the security best practices. The following phases and associated tools were used during the assessment:

- Research into architecture and purpose
- Smart contract manual code review and walkthrough
- Graphing out functionality and contract logic/connectivity/functions (solgraph)
- Manual assessment of use and safety for the critical Solidity variables and functions in scope to identify any arithmetic related vulnerability classes
- Manual testing by custom scripts
- Scanning of solidity files for vulnerabilities, security hot-spots or bugs. (MythX)
- Static Analysis of security for scoped contract, and imported functions. (Slither)
- Testnet deployment (Foundry, Brownie)

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)	1
	Partial (R:P)	0.5
	Full (R:F)	0.25
Scope (s)	Changed (S:C)	1.25
	Unchanged (S:U)	1

Severity Coefficient C is obtained by the following product:

$$C = rs$$

The Vulnerability Severity Score S is obtained by:

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

The score is rounded up to 1 decimal places.

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

2.4 SCOPE

Code repositories:

1. Substance Exchange V1

- Repository: SubstanceExchangeV1
- Commit ID: 136369e88c04d21a25fecbcf8a4f25d6363ee035
- Remediation Plan Commit ID: 7717277a15aef6b703a5cf9670509b2f0b1bd3fc
- Smart contracts in scope:
    1. Delegatable (src/core/Delegatable.sol)
    2. DelegationHub (src/core/DelegationHub.sol)
    3. ExchangeManager (src/core/ExchangeManager.sol)
    4. LiquidityPool (src/core/LiquidityPool.sol)
    5. SLPToken (src/core/SLPToken.sol)
    6. SubstanceProxy (src/core/SubstanceProxy.sol)
    7. SubstanceUSD (src/core/SubstanceUSD.sol)
    8. UserBalance (src/core/UserBalance.sol)
    9. BaseFuture (src/core/future/BaseFuture.sol)
    10. FutureFactory (src/core/future/FutureFactory.sol)
    11. FutureLong (src/core/future/FutureLong.sol)
    12. FutureLongV2 (src/core/future/FutureLongV2.sol)
    13. FutureShort (src/core/future/FutureShort.sol)
    14. FutureManager (src/core/future/FutureManager.sol)
    15. Option (src/core/option/Option.sol)
    16. OptionFactory (src/core/option/OptionFactory.sol)
    17. OptionManager (src/core/option/OptionManager.sol)
    18. Swap (src/core/swap/Swap.sol)
    19. SwapManager (src/core/swap/SwapManager.sol)
    20. GradualVester (src/core/token/GradualVester.sol)
    21. StakingReward (src/core/token/StakingReward.sol)
    22. SubstanceXToken (src/core/token/SubstanceXToken.sol)
    23. Struct (src/core/libraries/Struct.sol)
    24. TransferHelper (src/core/libraries/TransferHelper.sol)

Out-of-scope:

- third-party libraries and dependencies
- economic attacks

EXECUTIVE OVERVIEW

3. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	2	5	2

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
(HAL-01) CHAINLINK LATESTROUNDDATA MIGHT BE STALE OR INCORRECT	Medium (6.7)	SOLVED - 07/11/2023
(HAL-02) MISSING CHAINLINK ARBITRUM SEQUENCER HEALTH CHECK	Medium (6.7)	RISK ACCEPTED
(HAL-03) USING TRANSFER INSTEAD OF SAFETRANSFER	Low (3.1)	RISK ACCEPTED
(HAL-04) FEEONTRANSFER AND BURNONTRANSFER TOKENS ARE NOT SUPPORTED	Low (3.1)	RISK ACCEPTED
(HAL-05) CENTRALIZATION RISK: PRODUCT MANAGER CAN WITHDRAW ARBITRARY AMOUNTS FROM LIQUIDITY POOLS	Low (2.5)	RISK ACCEPTED
(HAL-06) CENTRALIZATION RISK: PRODUCT MANAGER CAN ALTER TOKEN RESERVES INDICATORS	Low (2.5)	RISK ACCEPTED
(HAL-07) POTENTIAL ACCESS CONTROL BYPASS	Low (2.5)	RISK ACCEPTED
(HAL-08) USING ERC721A INSTEAD OF ERC721 FOR MINTING ONLY 1 NFT AT A TIME	Informational (1.2)	SOLVED - 07/10/2023
(HAL-09) MISSING FEE RATES SANITY CHECKS	Informational (0.2)	ACKNOWLEDGED

FINDINGS & TECH DETAILS

4.1 (HAL-01) CHAINLINK LATESTROUNDDATA MIGHT BE STALE OR INCORRECT - MEDIUM (6.7)

Description:

Substance Exchange uses Chainlink as its price oracle. When buying or selling sUSD, the SubstanceUSD contract queries Chainlink for the underlying token price using the latestRoundData() function. This function returns uint80 roundId, int256 answer, uint256 startedAt, uint256 updatedAt and uint80 answeredInRound. roundId denotes the identifier of the most recent update round, answer is the price of the asset, startedAt is the timestamp at which the round started and updatedAt is the timestamp at which the feed was updated. The getPrice() function does not check if the feed was updated at the most recent round nor does it verify the update timestamp against the current time, and this can result in accepting stale data which may threaten the stability of the exchange in a volatile market.

Code Location:

SubstanceUSD.sol#L92

Listing 1: SubstanceUSD.sol (Line 92)

```
87     function getPrice(address token, bool min) public view returns
 ↳ (uint256 price) {
88         address oracle = underlyingToken[token].oracle;
89         if (oracle == address(0)) {
90             revert SubstanceUSD__InvalidToken();
91         }
92         (, int256 oraclePrice, , , ) = AggregatorV3Interface(
 ↳ oracle).latestRoundData();
93         if (oraclePrice <= 0) {
94             revert SubstanceUSD__InvalidOraclePrice();
95         }
96         uint8 pDecimals = AggregatorV3Interface(oracle).decimals()
 ↳ ;
```

```
        price = (uint256(oraclePrice) * PRECISION) / (10**pDecimals);
        price = min ? Math.min(PRECISION, price) : Math.max(PRECISION, price);
    }
```

BVSS:

AO:A/AC:L/AX:M/C:N/I:N/A:C/D:N/Y:N/R:N/S:U (6.7)

Recommendation:

It is recommended to establish confidence intervals for the roundId and updatedAt parameters and reject any Chainlink data feed response which falls outside those ranges.

Reference:

For further details, see Chainlink's latestRoundData might return stale or incorrect results

Remediation Plan::

SOLVED: The Substance Exchange team solved this issue in commit 7717277a.

4.2 (HAL-02) MISSING CHAINLINK ARBITRUM SEQUENCER HEALTH CHECK - MEDIUM (6.7)

Description:

Arbitrum is a L2 blockchain leveraging Optimistic Rollups to integrate with the underlying L1. A node called sequencer is tasked with submitting user transactions to the L1 and if it fails, communication between the two is impossible. The exchange does not verify if the sequencer is online, which may lead to unexpected behavior if submitting transactions to the Ethereum mainnet is blocked.

Code Location:

SubstanceUSD.sol#L92

Listing 2: SubstanceUSD.sol (Line 92)

```
87      function getPrice(address token, bool min) public view returns (uint256 price) {
88          address oracle = underlyingToken[token].oracle;
89          if (oracle == address(0)) {
90              revert SubstanceUSD__InvalidToken();
91          }
92          (, int256 oraclePrice, , , ) = AggregatorV3Interface(oracle).latestRoundData();
93          if (oraclePrice <= 0) {
94              revert SubstanceUSD__InvalidOraclePrice();
95          }
96          uint8 pDecimals = AggregatorV3Interface(oracle).decimals();
97          price = (uint256(oraclePrice) * PRECISION) / (10**pDecimals);
98          price = min ? Math.min(PRECISION, price) : Math.max(PRECISION, price);
99      }
```

BVSS:

AO:A/AC:L/AX:M/C:N/I:N/A:C/D:N/Y:N/R:N/S:U (6.7)

Recommendation:

It is recommended to verify the status of the Arbitrum sequencer before updating the contract state.

References:

Chainlink L2 Sequencer Uptime Feeds

Remediation Plan:

RISK ACCEPTED: The Substance Exchange team accepted the risk of this issue.

4.3 (HAL-03) USING TRANSFER INSTEAD OF SAFETRANSFER - LOW (3.1)

Description:

Using transfer() instead of safeTransfer() when interacting with ERC20 tokens is not recommended because transfer() does not provide the same level of error handling and safety measures.

Code Location:

The following contracts use transfer() function:

- ExchangeManager.sol
- LiquidityPool.sol
- SubstanceUSD.sol
- UserBalance.sol
- BaseFuture.sol
- FutureManager.sol
- Option.sol
- OptionManager.sol
- SwapManager.sol
- StakingReward.sol

BVSS:

AO:A/AC:L/AX:L/C:N/I:L/A:N/D:L/Y:N/R:N/S:U (3.1)

Recommendation:

It is recommended to use the SafeERC20 contract and the safeTransfer() function for token transfers.

Remediation Plan:

RISK ACCEPTED: The Substance Exchange team accepted the risk of this issue.

4.4 (HAL-04) FEEONTRANSFER AND BURNONTRANSFER TOKENS ARE NOT SUPPORTED - LOW (3.1)

Description:

Whenever a transfer of tokens is executed (in a swap, a deposit, or while adding liquidity), there’s no check if the amount sent is equal to the amount actually received by the contract.

The safeTransferFrom function calls transferFrom internally in the token contract to execute the transfer. However, the balance is not verified before and after the transfer and the actual amount transferred may not be the same as the amount received in the case of a fee applied in the token contract. In the case of using a token of this kind, the liquidity providers may not be able to withdraw all of their liquidity.

Code Location:

UserBalance.sol#L78

Listing 3: UserBalance.sol (Line 82)

```
78     function userDeposit(address _token, uint256 _amount) external {
79         _validTokenAddress(_token);
80         address user = msgSender();
81         IERC20(_token).safeTransferFrom(user, address(this), _amount);
82         userBalance[user][_token] += _amount;
83         emit Deposit(user, _token, _amount);
84     }
```

BVSS:

AO:A/AC:L/AX:L/C:N/I:M/A:N/D:M/Y:N/R:P/S:U (3.1)

Recommendation:

It is recommended to check the balance before and after the transfer to be sure the amount added to the user balance is actually the amount received by the protocol.

Listing 4: UserBalance.sol (Line 82)

```solidity
78    function userDeposit(address _token, uint256 _amount) external {
79        _validTokenAddress(_token);
80        address user = msgSender();
81        uint256 balanceBefore; = IERC20(_token).balanceOf(this);
82        IERC20(_token).safeTransferFrom(user, address(this), _amount);
83        _amount = IERC20(_token).balanceOf(this) - balanceBefore;
84        userBalance[user][_token] += _amount;
85        emit Deposit(user, _token, _amount);
86    }
```

Remediation Plan:

RISK ACCEPTED: The Substance Exchange team accepted the risk of this issue.

4.5 (HAL-05) CENTRALIZATION RISK: PRODUCT MANAGER CAN WITHDRAW ARBITRARY AMOUNTS FROM LIQUIDITY POOLS - LOW (2.5)

Description:

The LiquidityPool contract implements the external transfer function, which allows any account with the ProductManger role to withdraw arbitrary amounts of tokens from existing liquidity pools. In case such an account is compromised, the entire protocol liquidity is at risk.

Code Location:

LiquidityPool.sol#L499

Listing 5: LiquidityPool.sol (Line 501)

```
499     function transfer(address _token, address _dist, uint256
  ↳ _amount) external isProductManager {
500         poolAmount[_token] -= _amount;
501         IERC20(_token).transfer(_dist, _amount);
502     }
```

BVSS:

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

Recommendation:

It is recommended to reconsider the need for the transfer function and, if deemed necessary for the operations of the protocol, implement a decentralized solution like a multi-signature wallet to govern the exchange.

Remediation Plan:

RISK ACCEPTED: The Substance Exchange team accepted the risk of this issue.

4.6 (HAL-06) CENTRALIZATION RISK: PRODUCT MANAGER CAN ALTER TOKEN RESERVES INDICATORS - LOW (2.5)

Description:

The LiquidityPool contract implements the increaseLiquidity() and decreaseLiquidity() functions. If called by the Product Manager (which is a role assigned by the contract owner) they can alter the values reported by the token reserves trackers without actually touching the reserves, putting the exchange out of balance. This directly affects protocol accounting and may have negative consequences on the protocol and its users.

A good example of this could be a liquidity provider withdrawing their liquidity from the pool, where the transaction would revert.

Code Location:

LiquidityPool.sol#L489

Listing 6: LiquidityPool.sol (Line 491)

```
489     function increaseLiquidity(address _token, uint256 _amount)
  ↳ external isProductManager {
490         _validTokenCheck(_token);
491         poolAmount[_token] += _amount;
492     }
```

LiquidityPool.sol#L494

Listing 7: LiquidityPool.sol (Line 496)

```
494     function decreaseLiquidity(address _token, uint256 _amount)
  ↳ external isProductManager {
495         _validTokenCheck(_token);
496         poolAmount[_token] -= _amount;
```

```
    }
```

BVSS:

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

Recommendation:

It is recommended to reconsider the need for this function, and if deemed necessary force the execution of token transfers for the reserves to reflect the reported balances.

Remediation Plan:

RISK ACCEPTED: The Substance Exchange team accepted the risk of this issue.

4.7 (HAL-07) POTENTIAL ACCESS CONTROL BYPASS - LOW (2.5)

Description:

The ProductManager role is used to grant access from some contracts to other contract’s functions. There exists a scenario in which any user can be the owner of the entire protocol by means of using the hub contract's delegate calls to call any other protocol contract if the Delegation Hub contract is assigned the ProductManager role for those other contracts, effectively granting anyone privileged access to many sensitive functions.

Code Location:

Listing 8: OwnableUpgradeable.sol

```
    function transferOwnership(address newOwner) public virtual onlyOwner {
        require(newOwner != address(0), "Ownable: new owner is the zero address");
        _transferOwnership(newOwner);
    }
```

BVSS:

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

Recommendation:

It is recommended to add a requirement that the Hub contract cannot be assigned the ProductManager role.

Remediation Plan:

RISK ACCEPTED: The Substance Exchange team accepted the risk of this issue.

4.8 (HAL-08) USING ERC721A INSTEAD OF ERC721 FOR MINTING ONLY 1 NFT AT A TIME - INFORMATIONAL (1.2)

Description:

The StakingReward contract uses the ERC721A standard to store the users' staking weight and the corresponding rewards. ERC721A is designed to allow multiple mints at the same time with so-called batch transfers (the more tokens minted at the same time, the more gas efficient the operation is) and Substance Exchange is not implementing this core functionality in their protocol. In ERC721A, NFT transfers are more expensive because of the way NFT owner accounts are stored.

Code Location:

StakingReward.sol#L21

Listing 9: StakingReward.sol

```
21 contract StakingReward is Ownable, Delegatable, ERC721A {
```

Proof of Concept:

For the purposes of this PoC, two different types of NFTs were created, one based on the ERC721A standard and one based on ERC721.

Listing 10: HalbornERC721A.sol

```solidity
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.17;

import "../lib/erc721a/contracts/ERC721A.sol";

contract HalbornERC721A is ERC721A{

    constructor() ERC721A("Substance Exchange Stake Azuki", "SEXSTAKE") {}

    function mint(uint256 _quantity) external payable {
        _mint(msg.sender, _quantity);
    }

    function transfer(address _to, uint256 tokenId) external {
        transferFrom(msg.sender, _to, tokenId);
    }
}
```

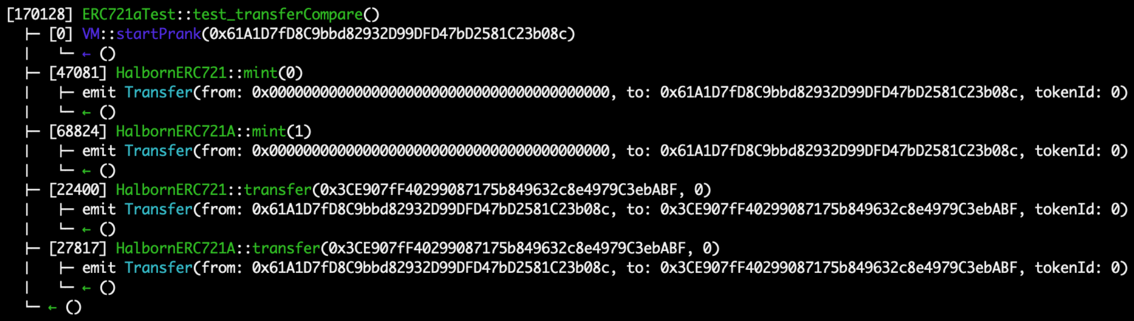
Listing 11: HalbornERC721.sol

```solidity
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.17;

import "../lib/openzeppelin-contracts/contracts/token/ERC721/ERC721.sol";

contract HalbornERC721 is ERC721{

    constructor() ERC721("Substance Exchange Stake OZ", "SEXSTAKE") {}

    function mint(uint256 _tokenId) public{
        _mint(msg.sender, _tokenId);
    }

    function transfer(address _to, uint256 _tokenId) public {
        _transfer(msg.sender, _to, _tokenId);
    }
}
```

The following scenario was simulated:

- Minting an ERC721 (ozNFT) with tokenId = 0
- Minting an ERC721A (azukiNFT) with quantity param = 1
- Transfer the ozNFT (tokenId = 0) from Alice to Bobby
- Transfer the azukiNFT (tokenId = 0) from Alice to Bobby

Scenario	Gas consumed
ozNFT mint	47081
azukiNFT mint	68824
ozNFT transfer	22400
azukiNFT transfer	27817



```
[170128] ERC721aTest::test_transferCompare()
  ├─ [0] VM::startPrank(0x61A1D7fD8C9bbd82932D99DFD47bD2581C23b08c)
  │   └─ ← ()
  ├─ [47081] HalbornERC721::mint(0)
  │   ├─ emit Transfer(from: 0x0000000000000000000000000000000000000000, to: 0x61A1D7fD8C9bbd82932D99DFD47bD2581C23b08c, tokenId: 0)
  │   └─ ← ()
  ├─ [68824] HalbornERC721A::mint(1)
  │   ├─ emit Transfer(from: 0x0000000000000000000000000000000000000000, to: 0x61A1D7fD8C9bbd82932D99DFD47bD2581C23b08c, tokenId: 0)
  │   └─ ← ()
  ├─ [22400] HalbornERC721::transfer(0x3CE907fF40299087175b849632c8e4979C3ebABF, 0)
  │   ├─ emit Transfer(from: 0x61A1D7fD8C9bbd82932D99DFD47bD2581C23b08c, to: 0x3CE907fF40299087175b849632c8e4979C3ebABF, tokenId: 0)
  │   └─ ← ()
  ├─ [27817] HalbornERC721A::transfer(0x3CE907fF40299087175b849632c8e4979C3ebABF, 0)
  │   ├─ emit Transfer(from: 0x61A1D7fD8C9bbd82932D99DFD47bD2581C23b08c, to: 0x3CE907fF40299087175b849632c8e4979C3ebABF, tokenId: 0)
  │   └─ ← ()
  └─ ← ()
```

BVSS:

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

Recommendation:

It is recommended to use the standard ERC721 from Openzeppelin instead of ERC721A for the StakingRewards if only one NFT is processed at a time.

Reference:

The ERC721A specification can be reviewed here and a detailed comparison with the ERC721 standard is available here.

Remediation Plan:

SOLVED: The Substance Exchange team solved this issue by replacing the ERC721A standard with ERC721 in commit e45003fe.

4.9 (HAL-09) MISSING FEE RATES SANITY CHECKS - INFORMATIONAL (0.2)

Description:

The exchange charges handling fees on certain operations. Fee rates are set by the contract owner and can be updated anytime. None of the setter functions does the sanity check of the provided fee rates, which may lead to contract owners introducing prohibitive or zero fees accidentally or by design.

Code Location:

SubstanceUSD.sol#L54

Listing 12: SubstanceUSD.sol

```
    function setFee(uint256 _mintFee, uint256 _burnFee) external onlyOwner {
        mintFee = _mintFee;
        burnFee = _burnFee;
    }
```

SwapManager.sol#L45

Listing 13: SwapManager.sol

```
    function setMinExecutionFee(uint256 _minExecutionFee) external onlyOwner {
        minExecutionFee = _minExecutionFee;
    }
```

OptionManager.sol#L59

Listing 14: OptionManager.sol

```
    function setMinExecutionFee(uint256 _minExecutionFee) external onlyOwner {
        minExecutionFee = _minExecutionFee;
    }
```

BaseFuture.sol#L167

Listing 15: BaseFuture.sol

```
    function setMinExecutionFee(uint256 _minExecutionFee) external onlyOwner {
        minExecutionFee = _minExecutionFee;
    }
```

BVSS:

AO:S/AC:L/AX:L/C:N/I:N/A:L/D:N/Y:L/R:F/S:U (0.2)

Recommendation:

It is recommended to restrict the fee rates to fixed ranges.

Remediation Plan:

ACKNOWLEDGED: The Substance Exchange team acknowledged this issue.

RETESTING

The issue described in this section was brought to Halborn’s attention by the Substance Exchange team during the engagement.

5.1 SUBSTANCE01 - USERS CAN OPEN FUTURE POSITIONS WITHOUT PROVIDING THE COLLATERAL ACCORDINGLY

Description:

The trader is able to open a trade with zero collateral. The code in BaseFuture and in FutureLongV2 is not updating the User’s position info prior to checking for liquidation. Therefore, if a position is opened that should have been instantly liquidated in the same transaction (i.e 0 collateral) the liquidation check returns false and then the position info being updated. Thus, opening a position with no collateral.

Code Location:

FutureLongV2.sol#L68

Listing 16: FutureLongV2.sol

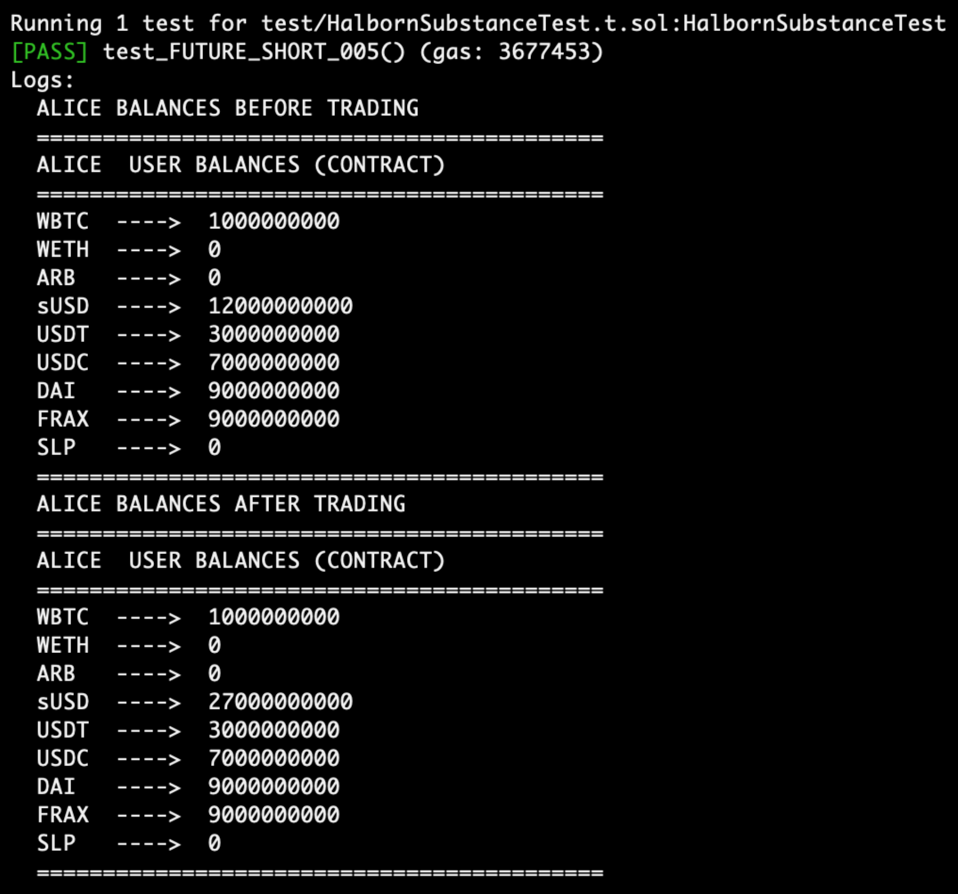
```solidity
    function increasePosition(
        address _user,
        uint256 _price,
        uint256 _increaseTokenSize,
        uint256 _increaseCollateral,
        Struct.FutureFeeInfo memory _feeInfo
    ) external override onlyManager returns (Struct.UpdatePositionResult memory result) {
        // step1, firstly add increased collateral into user's position and charge fees.
        Struct.Position storage position = s_position[_user];
        uint256 originCollateral = position.collateral;
        position.collateral += _increaseCollateral;
        {
            (uint256 _teamGainFeeUSD, uint256 _lpGainFeeUSD) = _chargeFees(
                _increaseTokenSize,
                _price,
                _feeInfo.txFeeRatio,
                _feeInfo.priceImpactRatio,
                _user
            );
            result.teamGainUSD += _teamGainFeeUSD;
            result.lpGainUSD += _lpGainFeeUSD;
        }

        // step2, update user's position storage
        (bool liquidated, uint256 userLostUSD) = _liquidationCheck(_user, _price);

        // step3, check liquidation or update store
        if (liquidated) {
            result.teamGainUSD += userLostUSD;
            result.unlockedTokenSize = originCollateral * maxProfitRatio;
            _emitLiquidatePosition(_user);
            _resetPosition(_user);
        } else {
            if (_maxProfitCheck(_user, _price)) {
                result.userBalanceGainUSD += position.collateral;
                result.userBalanceGainToken += position.collateral * maxProfitRatio;
                result.unlockedTokenSize = originCollateral * maxProfitRatio;
                _emitMaxProfitForceClosePosition(_user, result);
                _resetPosition(_user);
            } else {
                if (position.collateral >= originCollateral) {
                    result.lockedTokenSize = (position.collateral - originCollateral) * maxProfitRatio;
                } else {
                    result.unlockedTokenSize = (originCollateral - position.collateral) * maxProfitRatio;
                }
                position.tokenSize += _increaseTokenSize;
                position.openCost += getUSDValue(_increaseTokenSize, _price);

                sizeGlobal += _increaseTokenSize;
                costGlobal += getUSDValue(_increaseTokenSize,
 ↳ _price);
            }
        }

        _emitUpdatePosition(_user);
        _emitIncreasePosition(_user, result);

        return result;
    }
```

Proof of Concept:

STRATEGY:

- Alice has 12k$
- Open a position with 0 collateral for 1 BTC (owing to protocol 30k)$(apositionshouldnotbeopenedwith0collateral)-ThevalueofBTCisdecreasing50$
- Alice is closing the position
- Expected output: 0
- Result 27k$ (12k$ from before + 15k$ from the trade)

RETESTING



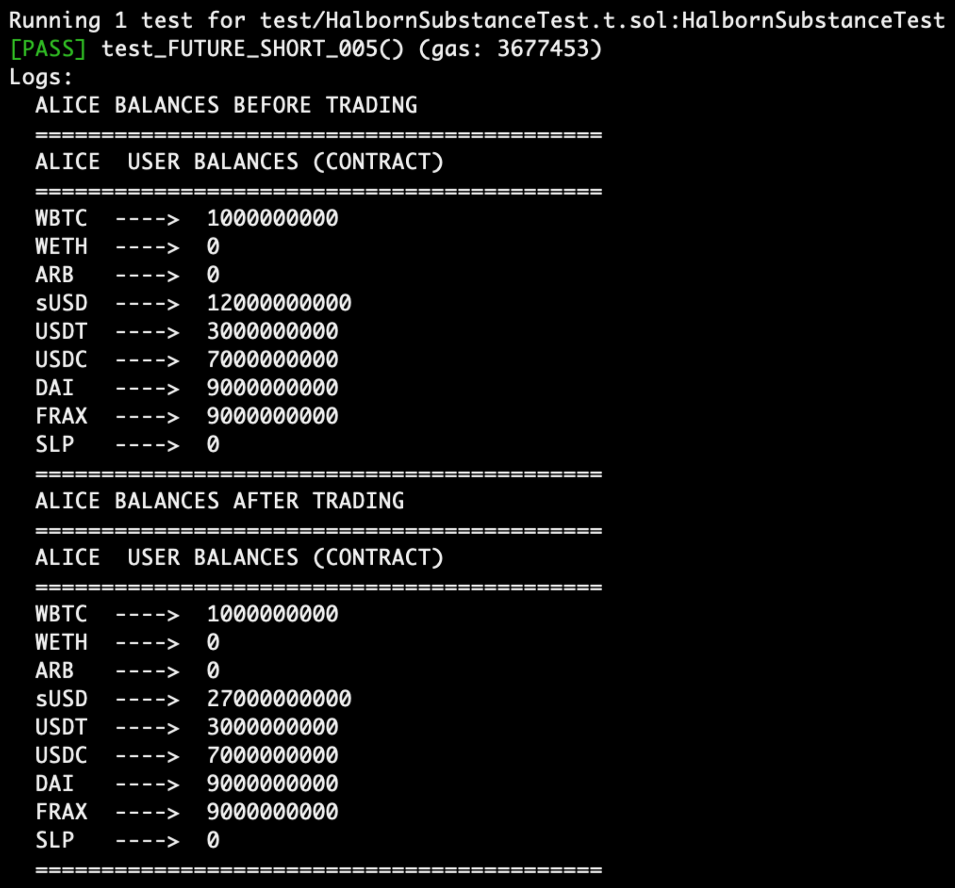
```
Running 1 test for test/HalbornSubstanceTest.t.sol:HalbornSubstanceTest
[PASS] test_FUTURE_SHORT_005() (gas: 3677453)
Logs:
  ALICE BALANCES BEFORE TRADING
  ===========================================
  ALICE  USER BALANCES (CONTRACT)
  ===========================================
  WBTC  ---->  1000000000
  WETH  ---->  0
  ARB   ---->  0
  sUSD  ---->  12000000000
  USDT  ---->  3000000000
  USDC  ---->  7000000000
  DAI   ---->  9000000000
  FRAX  ---->  9000000000
  SLP   ---->  0
  ===========================================
  ALICE BALANCES AFTER TRADING
  ===========================================
  ALICE  USER BALANCES (CONTRACT)
  ===========================================
  WBTC  ---->  1000000000
  WETH  ---->  0
  ARB   ---->  0
  sUSD  ---->  27000000000
  USDT  ---->  3000000000
  USDC  ---->  7000000000
  DAI   ---->  9000000000
  FRAX  ---->  9000000000
  SLP   ---->  0
  ===========================================
```

Recommendation:

The variables tokenSize, openCost, costGlobal, sizeGlobal should be updated before the liquidation check and not in the else statement.

Remediation Plan:

SOLVED: The Substance Exchange team found this issue and solved it by updating it before the liquidation check and outside the else statement.



```
Running 1 test for test/HalbornSubstanceTest.t.sol:HalbornSubstanceTest
[PASS] test_FUTURE_SHORT_005() (gas: 3677453)
Logs:
  ALICE BALANCES BEFORE TRADING
  ==========================================
  ALICE  USER BALANCES (CONTRACT)
  ==========================================
  WBTC  ---->  1000000000
  WETH  ---->  0
  ARB   ---->  0
  sUSD  ---->  12000000000
  USDT  ---->  3000000000
  USDC  ---->  7000000000
  DAI   ---->  9000000000
  FRAX  ---->  9000000000
  SLP   ---->  0
  ==========================================
  ALICE BALANCES AFTER TRADING
  ==========================================
  ALICE  USER BALANCES (CONTRACT)
  ==========================================
  WBTC  ---->  1000000000
  WETH  ---->  0
  ARB   ---->  0
  sUSD  ---->  27000000000
  USDT  ---->  3000000000
  USDC  ---->  7000000000
  DAI   ---->  9000000000
  FRAX  ---->  9000000000
  SLP   ---->  0
  ==========================================
```

FutureLongV2.sol - L40 // COMMIT ID: 1785e82f2b7229c9f4d650cee5f3848a2b55482c

MANUAL TESTING

The manual testing phase included isolated testing and integration testing to assure the correct functionality of the whole protocol. Whether it is an isolated test or an integration test, all of them are focused on checking a particular component, feature or functionality is working as expected. They can be summarized and categorized as follows:

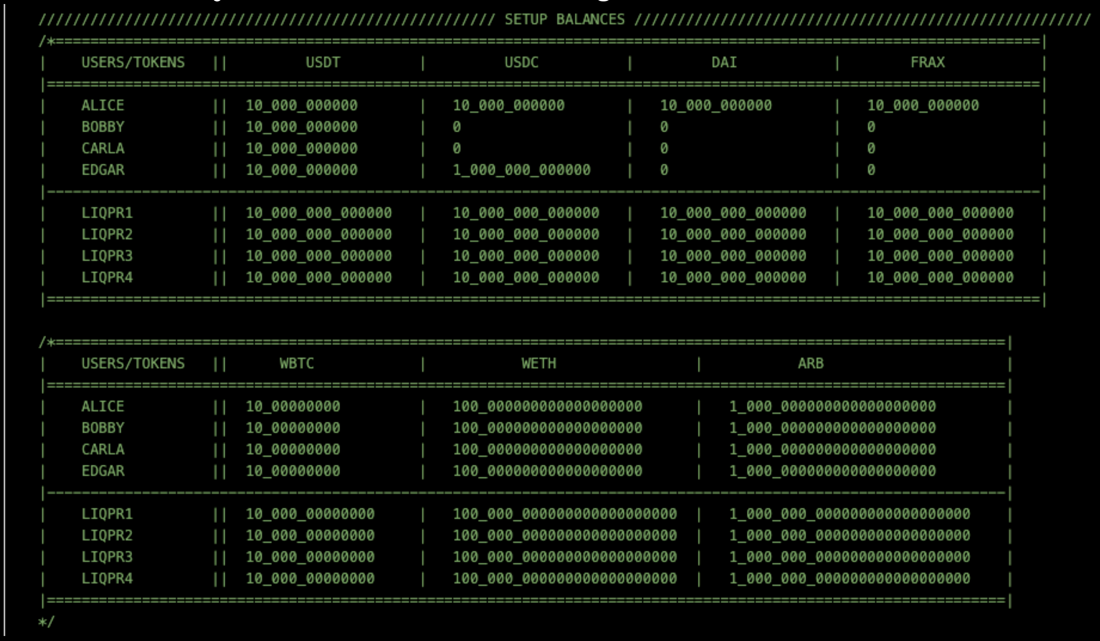
- Tokens used in the protocol:
  - Substance USD (sUSD): That users can use safely buy() and sell() returning the correct value from the oracles.
  - Substance Liquidity Pool (SLP): That SLP is correctly minted and burned depending on the actions in the liquidity pool and it is correctly managed.
  - Substance Exchange (SEX): That the governance is working properly while vesting and staking.
- Core contracts:
  - It was checked that the UserBalance contract is properly handling deposits and users balances.
  - It was checked that the DelegationHub contract is properly making the delegateCalls handling the delegations and keeps the entire protocol stable.
  - The LiquidityPool contract was tested to make sure the liquidity providers have their liquidity stored securely, and that users cannot act maliciously against any of the stakeholders. Furthermore, that the user’s balances are correctly calculated by the protocol.
- Futures and Options:
  - That all the future trades are handled correctly there in the correct epoch and cannot be doubled or double spent.
  - That the yield obtained from a successful trade are within the expected ranges
  - That the fees cannot be bypassed
  - That the incentives scheme is correctly implemented to incentivise the liquidations
  - That slippage tolerance is correctly handled
- Swaps:

- That swaps are working correctly with stable prices
- That there are no problems with slippage
- That the slippage of the price from the time the swap order is made until when it’s executed is correctly handled

6.1 SETUP

For the tests 3 types of users were created:
- Owner / Deployer (who has the ownership of and permissions for the protocol)
- Liquidity Providers: The users interested on providing liquidity and obtaining yield passively (liqPr1, liqPr2, liqPr3, and liqPr4 in the protocol)
- Traders: The users interested interacting with options and futures to obtain earnings actively (alice, bobby, carla, and edgar in the protocol)

The tests always start at the following state:



```
//////////////////////////////////////////////////// SETUP BALANCES ////////////////////////////////////////////////////
/*==========================================================================================================|
|    USERS/TOKENS   ||         USDT          |          USDC          |          DAI           |          FRAX          |
|==========================================================================================================|
|    ALICE          ||  10_000_000000        |  10_000_000000         |  10_000_000000         |  10_000_000000         |
|    BOBBY          ||  10_000_000000        |  0                     |  0                     |  0                      |
|    CARLA          ||  10_000_000000        |  0                     |  0                     |  0                      |
|    EDGAR          ||  10_000_000000        |  1_000_000_000000      |  0                     |  0                      |
|----------------------------------------------------------------------------------------------------------|
|    LIQPR1         ||  10_000_000_000000    |  10_000_000_000000     |  10_000_000_000000     |  10_000_000_000000     |
|    LIQPR2         ||  10_000_000_000000    |  10_000_000_000000     |  10_000_000_000000     |  10_000_000_000000     |
|    LIQPR3         ||  10_000_000_000000    |  10_000_000_000000     |  10_000_000_000000     |  10_000_000_000000     |
|    LIQPR4         ||  10_000_000_000000    |  10_000_000_000000     |  10_000_000_000000     |  10_000_000_000000     |
|==========================================================================================================|

/*======================================================================================================|
|    USERS/TOKENS   ||       WBTC            |              WETH               |              ARB                |
|======================================================================================================|
|    ALICE          ||  10_00000000          |  100_000000000000000000         |  1_000_000000000000000000       |
|    BOBBY          ||  10_00000000          |  100_000000000000000000         |  1_000_000000000000000000       |
|    CARLA          ||  10_00000000          |  100_000000000000000000         |  1_000_000000000000000000       |
|    EDGAR          ||  10_00000000          |  100_000000000000000000         |  1_000_000000000000000000       |
|------------------------------------------------------------------------------------------------------|
|    LIQPR1         ||  10_000_00000000      |  100_000_000000000000000000     |  1_000_000_000000000000000000   |
|    LIQPR2         ||  10_000_00000000      |  100_000_000000000000000000     |  1_000_000_000000000000000000   |
|    LIQPR3         ||  10_000_00000000      |  100_000_000000000000000000     |  1_000_000_000000000000000000   |
|    LIQPR4         ||  10_000_00000000      |  100_000_000000000000000000     |  1_000_000_000000000000000000   |
|======================================================================================================|
*/
```

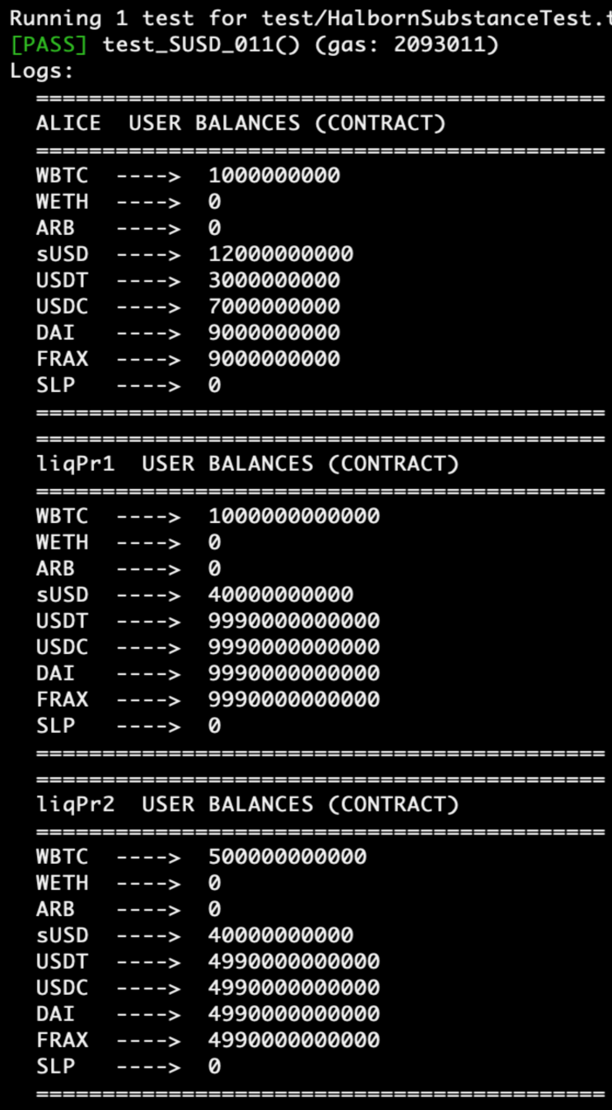
Below are given some examples of the tests performed:

6.2 BUY SCENARIO

- All the users are approving, depositing in the contract and buying sUSD
- liqPr1 is buying USD with all their stablecoins.
- liqPr2 is buying USD with half of their stablecoins.
- alice deposits all her balance in the UserBalance contract but is

only buying USD with 7000 USDT, 3000 USDC, 1000 DAI and 1000 FRAX

RESULTS:



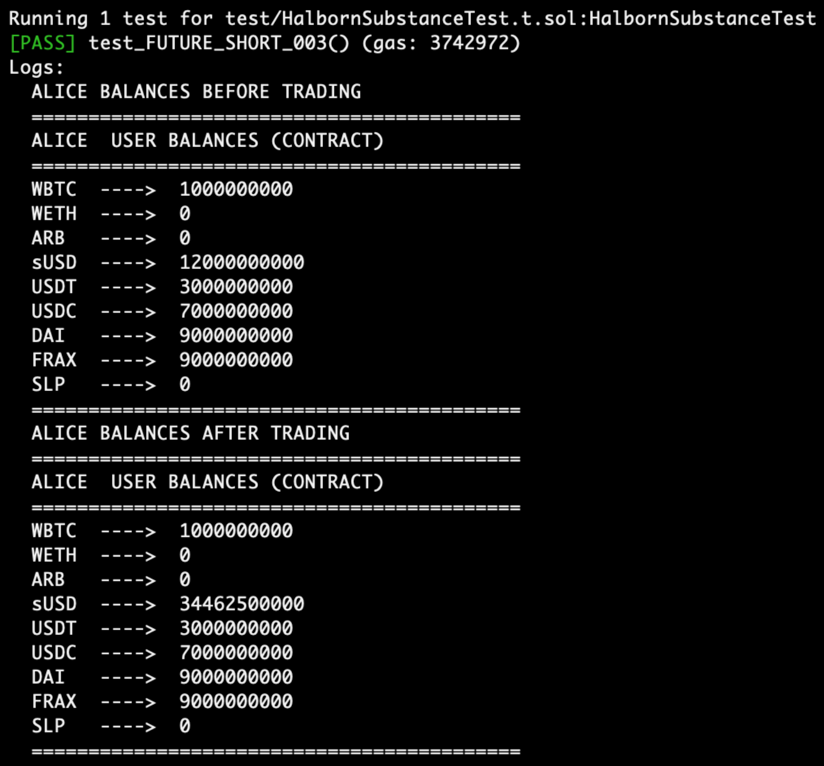
```
Running 1 test for test/HalbornSubstanceTest.t.sol:HalbornSubstanceTest
[PASS] test_SUSD_011() (gas: 2093011)
Logs:
  ============================================
  ALICE  USER BALANCES (CONTRACT)
  ============================================
  WBTC  ---->  1000000000
  WETH  ---->  0
  ARB   ---->  0
  sUSD  ---->  12000000000
  USDT  ---->  3000000000
  USDC  ---->  7000000000
  DAI   ---->  9000000000
  FRAX  ---->  9000000000
  SLP   ---->  0
  ============================================
  ============================================
  liqPr1  USER BALANCES (CONTRACT)
  ============================================
  WBTC  ---->  1000000000000
  WETH  ---->  0
  ARB   ---->  0
  sUSD  ---->  40000000000
  USDT  ---->  9990000000000
  USDC  ---->  9990000000000
  DAI   ---->  9990000000000
  FRAX  ---->  9990000000000
  SLP   ---->  0
  ============================================
  ============================================
  liqPr2  USER BALANCES (CONTRACT)
  ============================================
  WBTC  ---->  500000000000
  WETH  ---->  0
  ARB   ---->  0
  sUSD  ---->  40000000000
  USDT  ---->  4990000000000
  USDC  ---->  4990000000000
  DAI   ---->  4990000000000
  FRAX  ---->  4990000000000
  SLP   ---->  0
  ============================================
```

6.3 SHORT SCENARIO

STRATEGY:

- Alice has $12000 USD as initial balance
- Opens a x10 Short to BTC with $3000 USD Collateral
- BTC decreases the value for a 75%, with a value of $7500USD
- Alice closes the position, having to return around 0.9WBTC and making a $22.5k profit
- Expected output: $12000 + $22500 = 34500$
- Result: $34462.5

RESULTS:

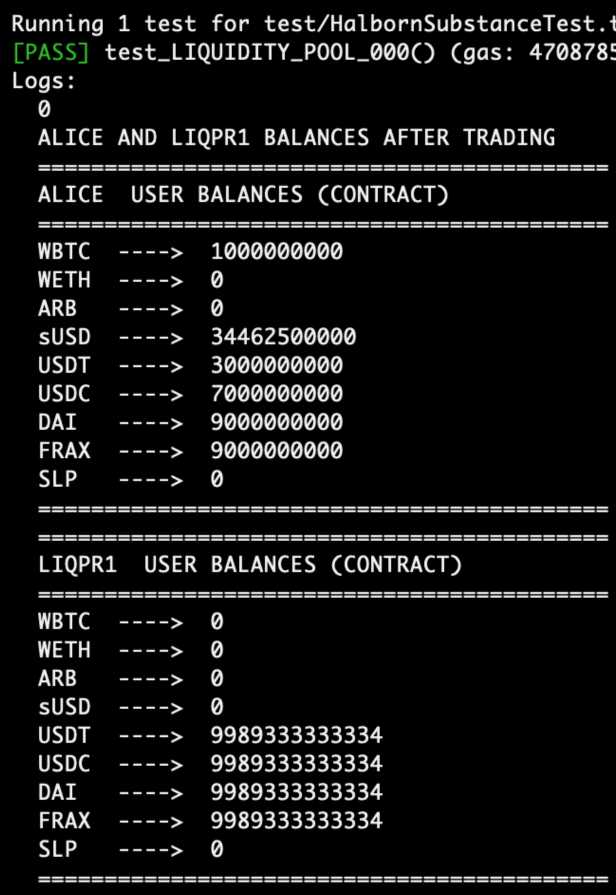


```
Running 1 test for test/HalbornSubstanceTest.t.sol:HalbornSubstanceTest
[PASS] test_FUTURE_SHORT_003() (gas: 3742972)
Logs:
  ALICE BALANCES BEFORE TRADING
  ==========================================
  ALICE  USER BALANCES (CONTRACT)
  ==========================================
  WBTC  ---->  1000000000
  WETH  ---->  0
  ARB   ---->  0
  sUSD  ---->  12000000000
  USDT  ---->  3000000000
  USDC  ---->  7000000000
  DAI   ---->  9000000000
  FRAX  ---->  9000000000
  SLP   ---->  0
  ==========================================
  ALICE BALANCES AFTER TRADING
  ==========================================
  ALICE  USER BALANCES (CONTRACT)
  ==========================================
  WBTC  ---->  1000000000
  WETH  ---->  0
  ARB   ---->  0
  sUSD  ---->  34462500000
  USDT  ---->  3000000000
  USDC  ---->  7000000000
  DAI   ---->  9000000000
  FRAX  ---->  9000000000
  SLP   ---->  0
  ==========================================
```

6.4 LIQUIDITY POOL SCENARIO

- Alice is shorting BTC with $3000 of collateral
- LiqPr1 is depositing $40k USD in stablecoins to buy 4$0k sUSD and then provide it to the liquidity pool.
- BTC decreases the value for a 75%, with a value of $7500USD
- Alice closes the position, winning the trade and obtaining rewards
- LiqPr1 is withdrawing the liquidity again to their stablecoins
- Result: LiqPr1 is losing

RESULTS:



```
Running 1 test for test/HalbornSubstanceTest.t.sol:HalbornSubstanceTest
[PASS] test_LIQUIDITY_POOL_000() (gas: 4708785)
Logs:
  0
  ALICE AND LIQPR1 BALANCES AFTER TRADING
  ===========================================
  ALICE  USER BALANCES (CONTRACT)
  ===========================================
  WBTC  ---->  1000000000
  WETH  ---->  0
  ARB   ---->  0
  sUSD  ---->  34462500000
  USDT  ---->  3000000000
  USDC  ---->  7000000000
  DAI   ---->  9000000000
  FRAX  ---->  9000000000
  SLP   ---->  0
  ===========================================
  ===========================================
  LIQPR1  USER BALANCES (CONTRACT)
  ===========================================
  WBTC  ---->  0
  WETH  ---->  0
  ARB   ---->  0
  sUSD  ---->  0
  USDT  ---->  9989333333334
  USDC  ---->  9989333333334
  DAI   ---->  9989333333334
  FRAX  ---->  9989333333334
  SLP   ---->  0
  ===========================================
```

AUTOMATED TESTING

7.1 STATIC ANALYSIS REPORT

Description:

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

The security team assessed all findings identified by the Slither software, however, findings with severity Information and Optimization are not included in the below results for the sake of report readability.

Results:

Slither results for Delegatable.sol	
Finding	**Impact**
Delegatable._checkOperator() (src/core/Delegatable.sol#25-30) is never used and should be removed	Low
End of table for Delegatable.sol	

Slither results for DelegationHub.sol	
Finding	**Impact**
Reentrancy in DelegationHub._aggregate(address,DelegationHub.Call[]) (src/core/DELEGATIONHUB.sol#109-148): External calls: (success,retData) = calli.target.callvalue: val(calli.payload) (src/core/DELEGATIONHUB.sol#127) State variables written after the call(s): senderOverride = address(0) (src/core/DELEGATIONHUB.sol#147) DelegationHub.senderOverride (src/core/DELEGATIONHUB.sol#21) can be used in cross function reentrancies: DelegationHub._aggregate(address,DelegationHub.Call[]) (src/core/DELEGATIONHUB.sol#109-148) DelegationHub.msgSender() (src/core/DELEGATIONHUB.sol#80-86) DelegationHub.receiveEthFromUserBalance() (src/core/DELEGATIONHUB.sol#49-54)	High
DelegationHub (src/core/DELEGATIONHUB.sol#16-151) is an upgradeable contract that does not protect its initialize functions: DelegationHub.initialize() (src/core/DELEGATIONHUB.sol#41-43). Anyone can delete the contract with: UUPSUpgradeable.upgradeTo(address) (lib/openzeppelin-contracts/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol#74-77) UUPSUpgradeable.upgradeToAndCall(address,bytes) (lib/openzeppelin-contracts/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol#89-92)	High
DelegationHub.setOperator(address[],bool[]).i (src/core/DELEGATIONHUB.sol#61) is a local variable never initialized ERC1967UpgradeUpgradeable._upgradeToAndCallUUPS(address,bytes,bool).slot (lib/openzeppelin-contracts/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#84) is a local variable never initialized	Medium

Finding	Impact
DelegationHub.setOperator(address[],bool[]).i (src/core/DELEGATIONHUB.sol#61) is a local variable never initialized ERC1967UpgradeUpgradeable._upgradeToAndCallUUPS(address,bytes,bool).slot (lib/openzeppelin-contracts/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#84) is a local variable never initialized	Low
End of table for DelegationHub.sol	

Slither results for ExchangeManager.sol	
Finding	**Impact**
ExchangeManager (src/core/ExchangeManager.sol#16-100) is an upgradeable contract that does not protect its initialize functions: ExchangeManager.initialize(UserBalance, LiquidityPool, OptionManager, FutureManager, SwapManager) (src/core/ExchangeManager.sol#30-44).	High
ExchangeManager.userClaimWithdrawLiquidity(uint256).i (src/core/ExchangeManager.sol#89) is a local variable never initialized	Medium
ExchangeManager.setHub(address).hub (src/core/ExchangeManager.sol#46) shadows: Delegatable.hub (src/core/Delegatable.sol#10) (state variable)	Low
End of table for ExchangeManager.sol	

Slither results for LiquidityPool.sol	
Finding	**Impact**
LiquidityPool.adminWithdrawBurnFees(address) (src/core/LiquidityPool.sol#183-190) ignores return value by IERC20(token).transfer(_collection,feesAvailable[token]) (src/core/LiquidityPool.sol#187)	High
LiquidityPool.globalWithdrawToken(int256) (src/core/LiquidityPool.sol#315-384) ignores return value by IERC20(token_scope_1).transfer(userBalance,withdrawAmount) (src/core/LiquidityPool.sol#377)	High
LiquidityPool.transfer(address,address,uint256) (src/core/LiquidityPool.sol#510-513) ignores return value by IERC20(_token).transfer(_dist,_amount) (src/core/LiquidityPool.sol#512)	High

Finding	Impact
LiquidityPool.getSLPDiscountRatio(uint256,uint256,uint256) (src/core/LiquidityPool.sol#574-594) performs a multiplication on the result of a division: averageDiff = (initDiff + nextDiff) / 2 (src/core/LiquidityPool.sol#583) taxBps = (taxBasisPoints * averageDiff) / targetRatio (src/core/LiquidityPool.sol#589)	Medium
Reentrancy in LiquidityPool.adminWithdrawBurnFees(address) (src/core/LiquidityPool.sol#183-190): External calls: IERC20(token).transfer(_collection,feesAvailable[token]) (src/core/LiquidityPool.sol#187) State variables written after the call(s): feesAvailable[token] = 0 (src/core/LiquidityPool.sol#188) LiquidityPool.feesAvailable (src/core/LiquidityPool.sol#61) can be used in cross function reentrancies: LiquidityPool.adminWithdrawBurnFees(address) (src/core/LiquidityPool.sol#183-190) LiquidityPool.feesAvailable (src/core/LiquidityPool.sol#61) LiquidityPool.globalWithdrawToken(int256) (src/core/LiquidityPool.sol#315-384)	Medium
LiquidityPool.claimCurrentEpochLiquidityTokenPrices(uint256[]).i (src/core/LiquidityPool.sol#250) is a local variable never initialized	Medium
LiquidityPool.globalWithdrawToken(int256).i (src/core/LiquidityPool.sol#331) is a local variable never initialized	Medium
LiquidityPool._addValidLiquidityToken(address,uint8) (src/core/LiquidityPool.sol#161-169) has external calls inside a loop: IERC20(_tokenAddress).totalSupply() (src/core/LiquidityPool.sol#168)	Low
End of table for LiquidityPool.sol	

Slither results for SLPToken.sol	
Finding	**Impact**
SLPToken.setPool(address)._pool (src/core/SLPToken.sol#15) lacks a zero-check on : - pool = _pool (src/core/SLPToken.sol#16)	Low
End of table for SLPToken.sol	

Slither results for SubstanceProxy.sol

Finding	Impact
ERC1967Upgrade._upgradeToAndCallUUPS(address,bytes,bool).slot (lib/openzeppelin-contracts/contracts/proxy/ERC1967/ERC1967Upgrade.sol#78) is a local variable never initialized	Medium
ERC1967Upgrade._upgradeToAndCall(address,bytes,bool) (lib/openzeppelin-contracts/contracts/proxy/ERC1967/ERC1967Upgrade.sol#59-64) ignores return value by Address.functionDelegateCall(newImplementation,data) (lib/openzeppelin-contracts/contracts/proxy/ERC1967/ERC1967Upgrade.sol#62)	Medium
ERC1967Upgrade._upgradeToAndCallUUPS(address,bytes,bool) (lib/openzeppelin-contracts/contracts/proxy/ERC1967/ERC1967Upgrade.sol#71-85) ignores return value by IERC1822Proxiable(newImplementation).proxiableUUID() (lib/openzeppelin-contracts/contracts/proxy/ERC1967/ERC1967Upgrade.sol#78-82)	Low

End of table for SubstanceProxy.sol

Slither results for SubstanceUSD.sol

Finding	Impact
SubstanceUSD (src/core/SubstanceUSD.sol#23-151) is an upgradeable contract that does not protect its initialize functions: SubstanceUSD.initialize(IUserBalance,uint256,uint256,address) (src/core/SubstanceUSD.sol#44-52).	High
Math.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts/utils/math/Math.sol#55-134) performs a multiplication on the result of a division: - denominator = denominator / twos (lib/openzeppelin-contracts/contracts/utils/math/Math.sol#101)	Medium
SubstanceUSD.setHub(address).hub (src/core/SubstanceUSD.sol#78) shadows: - Delegatable.hub (src/core/Delegatable.sol#10) (state variable)	Low

End of table for SubstanceUSD.sol

<table>
<tr><th colspan="2">Slither results for UserBalance.sol</th></tr>
<tr><th>Finding</th><th>Impact</th></tr>
<tr><td>UserBalance.transfer(address,address,address,uint256) (src/core/UserBalance.sol#148-156) ignores return value by IERC20(_token).transfer(_to,_amount) (src/core/UserBalance.sol#155)</td><td>High</td></tr>
<tr><td colspan="2">End of table for UserBalance.sol</td></tr>
</table>

Results summary:

The findings obtained as a result of the Slither scan were reviewed. The majority of Slither findings were determined false-positives.

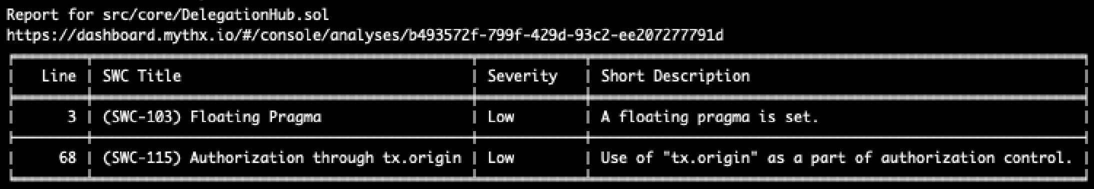
7.2 AUTOMATED SECURITY SCAN

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 the smart contracts and sent the compiled results to the analyzers in order to locate any vulnerabilities.

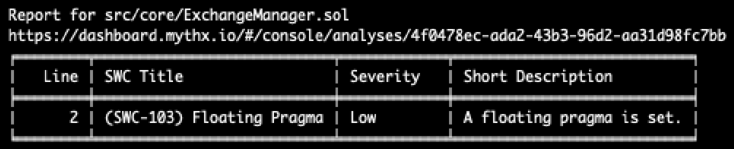
Results:

src/core/DelegationHub.sol



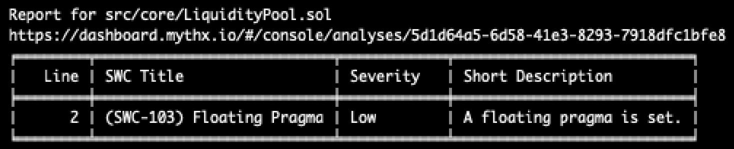
```
Report for src/core/DelegationHub.sol
https://dashboard.mythx.io/#/console/analyses/b493572f-799f-429d-93c2-ee207277791d
╒════════╤════════════════════════════════════════╤════════════╤══════════════════════════════════════════════════╕
│   Line │ SWC Title                              │ Severity   │ Short Description                                │
╞════════╪════════════════════════════════════════╪════════════╪══════════════════════════════════════════════════╡
│      3 │ (SWC-103) Floating Pragma              │ Low        │ A floating pragma is set.                        │
├────────┼────────────────────────────────────────┼────────────┼──────────────────────────────────────────────────┤
│     68 │ (SWC-115) Authorization through tx.origin │ Low     │ Use of "tx.origin" as a part of authorization control. │
╘════════╧════════════════════════════════════════╧════════════╧══════════════════════════════════════════════════╛
```

src/core/ExchangeManager.sol



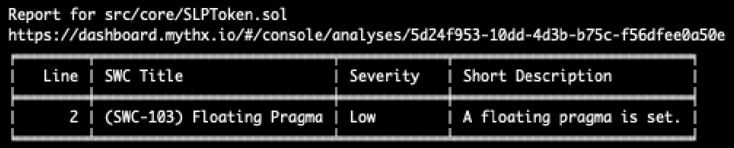
```
Report for src/core/ExchangeManager.sol
https://dashboard.mythx.io/#/console/analyses/4f0478ec-ada2-43b3-96d2-aa31d98fc7bb
╒════════╤═══════════════════════════╤════════════╤═══════════════════════════╕
│   Line │ SWC Title                 │ Severity   │ Short Description         │
╞════════╪═══════════════════════════╪════════════╪═══════════════════════════╡
│      2 │ (SWC-103) Floating Pragma │ Low        │ A floating pragma is set. │
╘════════╧═══════════════════════════╧════════════╧═══════════════════════════╛
```

src/core/LiquidityPool.sol



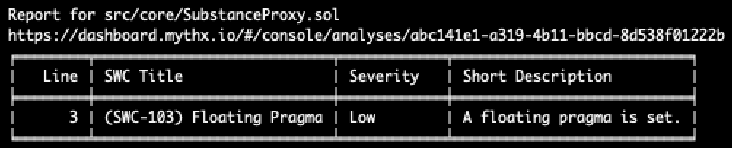
```
Report for src/core/LiquidityPool.sol
https://dashboard.mythx.io/#/console/analyses/5d1d64a5-6d58-41e3-8293-7918dfc1bfe8
╒════════╤═══════════════════════════╤════════════╤═══════════════════════════╕
│   Line │ SWC Title                 │ Severity   │ Short Description         │
╞════════╪═══════════════════════════╪════════════╪═══════════════════════════╡
│      2 │ (SWC-103) Floating Pragma │ Low        │ A floating pragma is set. │
╘════════╧═══════════════════════════╧════════════╧═══════════════════════════╛
```

src/core/SLPToken.sol



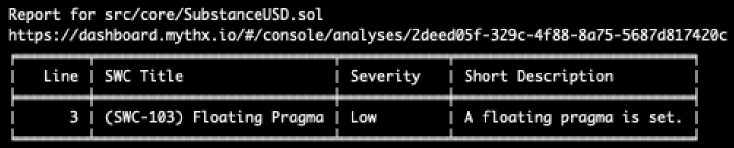
```
Report for src/core/SLPToken.sol
https://dashboard.mythx.io/#/console/analyses/5d24f953-10dd-4d3b-b75c-f56dfee0a50e
╒════════╤═══════════════════════════╤════════════╤═══════════════════════════╕
│   Line │ SWC Title                 │ Severity   │ Short Description         │
╞════════╪═══════════════════════════╪════════════╪═══════════════════════════╡
│      2 │ (SWC-103) Floating Pragma │ Low        │ A floating pragma is set. │
╘════════╧═══════════════════════════╧════════════╧═══════════════════════════╛
```

src/core/SubstanceProxy.sol



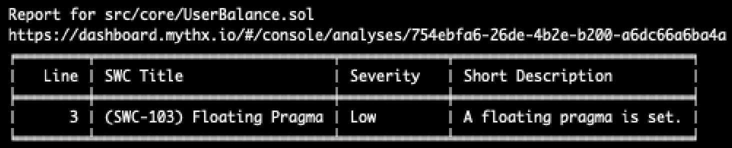
```
Report for src/core/SubstanceProxy.sol
https://dashboard.mythx.io/#/console/analyses/abc141e1-a319-4b11-bbcd-8d538f01222b
╒════════╤═══════════════════════════╤════════════╤═══════════════════════════╕
│   Line │ SWC Title                 │ Severity   │ Short Description         │
╞════════╪═══════════════════════════╪════════════╪═══════════════════════════╡
│      3 │ (SWC-103) Floating Pragma │ Low        │ A floating pragma is set. │
╘════════╧═══════════════════════════╧════════════╧═══════════════════════════╛
```

src/core/SubstanceUSD.sol



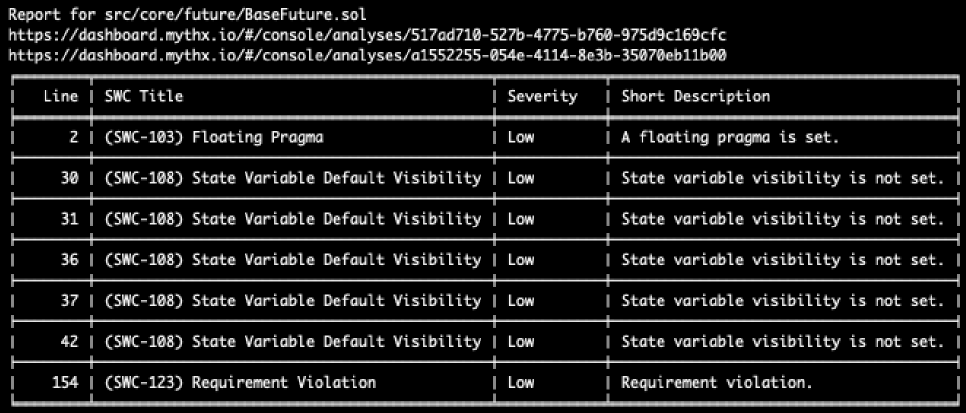
```
Report for src/core/SubstanceUSD.sol
https://dashboard.mythx.io/#/console/analyses/2deed05f-329c-4f88-8a75-5687d817420c
╒════════╤═══════════════════════════╤════════════╤═══════════════════════════╕
│   Line │ SWC Title                 │ Severity   │ Short Description         │
╞════════╪═══════════════════════════╪════════════╪═══════════════════════════╡
│      3 │ (SWC-103) Floating Pragma │ Low        │ A floating pragma is set. │
╘════════╧═══════════════════════════╧════════════╧═══════════════════════════╛
```

src/core/UserBalance.sol



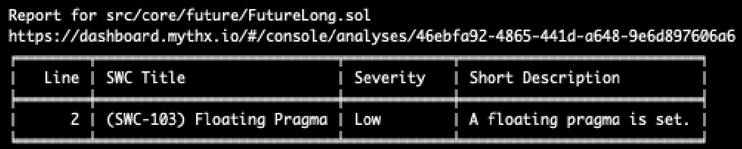
```
Report for src/core/UserBalance.sol
https://dashboard.mythx.io/#/console/analyses/754ebfa6-26de-4b2e-b200-a6dc66a6ba4a
╒════════╤═══════════════════════════╤════════════╤═══════════════════════════╕
│   Line │ SWC Title                 │ Severity   │ Short Description         │
╞════════╪═══════════════════════════╪════════════╪═══════════════════════════╡
│      3 │ (SWC-103) Floating Pragma │ Low        │ A floating pragma is set. │
╘════════╧═══════════════════════════╧════════════╧═══════════════════════════╛
```

src/core/future/BaseFuture.sol



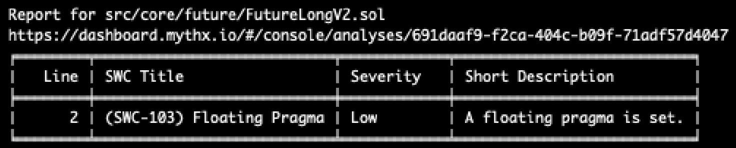
```
Report for src/core/future/BaseFuture.sol
https://dashboard.mythx.io/#/console/analyses/517ad710-527b-4775-b760-975d9c169cfc
https://dashboard.mythx.io/#/console/analyses/a1552255-054e-4114-8e3b-35070eb11b00
╒════════╤═════════════════════════════════════════════╤════════════╤═══════════════════════════════════════╕
│   Line │ SWC Title                                   │ Severity   │ Short Description                     │
╞════════╪═════════════════════════════════════════════╪════════════╪═══════════════════════════════════════╡
│      2 │ (SWC-103) Floating Pragma                   │ Low        │ A floating pragma is set.             │
├────────┼─────────────────────────────────────────────┼────────────┼───────────────────────────────────────┤
│     30 │ (SWC-108) State Variable Default Visibility │ Low        │ State variable visibility is not set. │
├────────┼─────────────────────────────────────────────┼────────────┼───────────────────────────────────────┤
│     31 │ (SWC-108) State Variable Default Visibility │ Low        │ State variable visibility is not set. │
├────────┼─────────────────────────────────────────────┼────────────┼───────────────────────────────────────┤
│     36 │ (SWC-108) State Variable Default Visibility │ Low        │ State variable visibility is not set. │
├────────┼─────────────────────────────────────────────┼────────────┼───────────────────────────────────────┤
│     37 │ (SWC-108) State Variable Default Visibility │ Low        │ State variable visibility is not set. │
├────────┼─────────────────────────────────────────────┼────────────┼───────────────────────────────────────┤
│     42 │ (SWC-108) State Variable Default Visibility │ Low        │ State variable visibility is not set. │
├────────┼─────────────────────────────────────────────┼────────────┼───────────────────────────────────────┤
│    154 │ (SWC-123) Requirement Violation             │ Low        │ Requirement violation.                │
╘════════╧═════════════════════════════════════════════╧════════════╧═══════════════════════════════════════╛
```

src/core/future/FutureLong.sol



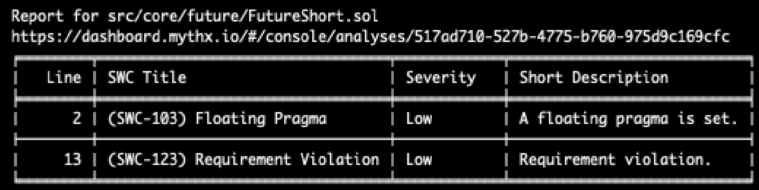
```
Report for src/core/future/FutureLong.sol
https://dashboard.mythx.io/#/console/analyses/46ebfa92-4865-441d-a648-9e6d897606a6
╒════════╤═══════════════════════════╤════════════╤═══════════════════════════╕
│   Line │ SWC Title                 │ Severity   │ Short Description         │
╞════════╪═══════════════════════════╪════════════╪═══════════════════════════╡
│      2 │ (SWC-103) Floating Pragma │ Low        │ A floating pragma is set. │
╘════════╧═══════════════════════════╧════════════╧═══════════════════════════╛
```

src/core/future/FutureLongV2.sol



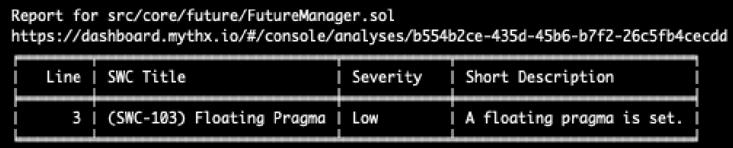
```
Report for src/core/future/FutureLongV2.sol
https://dashboard.mythx.io/#/console/analyses/691daaf9-f2ca-404c-b09f-71adf57d4047
╒════════╤═══════════════════════════╤════════════╤═══════════════════════════╕
│   Line │ SWC Title                 │ Severity   │ Short Description         │
╞════════╪═══════════════════════════╪════════════╪═══════════════════════════╡
│      2 │ (SWC-103) Floating Pragma │ Low        │ A floating pragma is set. │
╘════════╧═══════════════════════════╧════════════╧═══════════════════════════╛
```

src/core/future/FutureShort.sol



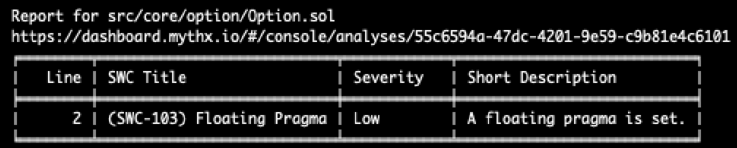
```
Report for src/core/future/FutureShort.sol
https://dashboard.mythx.io/#/console/analyses/517ad710-527b-4775-b760-975d9c169cfc

┌──────┬───────────────────────────────┬──────────┬───────────────────────────┐
│ Line │ SWC Title                     │ Severity │ Short Description         │
├──────┼───────────────────────────────┼──────────┼───────────────────────────┤
│    2 │ (SWC-103) Floating Pragma     │ Low      │ A floating pragma is set. │
├──────┼───────────────────────────────┼──────────┼───────────────────────────┤
│   13 │ (SWC-123) Requirement Violation │ Low    │ Requirement violation.    │
└──────┴───────────────────────────────┴──────────┴───────────────────────────┘
```

src/core/future/FutureManager.sol



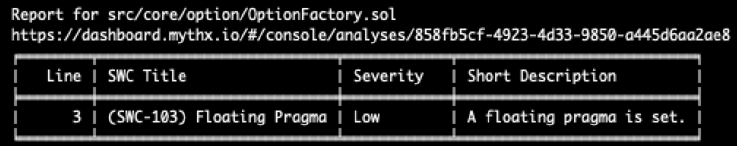
```
Report for src/core/future/FutureManager.sol
https://dashboard.mythx.io/#/console/analyses/b554b2ce-435d-45b6-b7f2-26c5fb4cecdd

┌──────┬───────────────────────────┬──────────┬───────────────────────────┐
│ Line │ SWC Title                 │ Severity │ Short Description         │
├──────┼───────────────────────────┼──────────┼───────────────────────────┤
│    3 │ (SWC-103) Floating Pragma │ Low      │ A floating pragma is set. │
└──────┴───────────────────────────┴──────────┴───────────────────────────┘
```

src/core/option/Option.sol



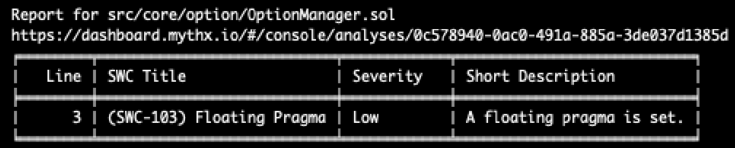
```
Report for src/core/option/Option.sol
https://dashboard.mythx.io/#/console/analyses/55c6594a-47dc-4201-9e59-c9b81e4c6101

┌──────┬───────────────────────────┬──────────┬───────────────────────────┐
│ Line │ SWC Title                 │ Severity │ Short Description         │
├──────┼───────────────────────────┼──────────┼───────────────────────────┤
│    2 │ (SWC-103) Floating Pragma │ Low      │ A floating pragma is set. │
└──────┴───────────────────────────┴──────────┴───────────────────────────┘
```

src/core/option/OptionFactory.sol



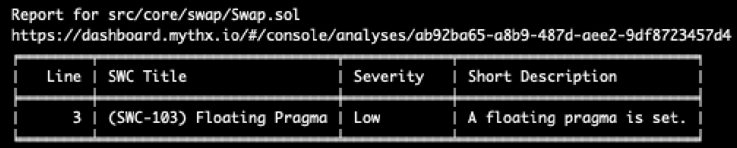
```
Report for src/core/option/OptionFactory.sol
https://dashboard.mythx.io/#/console/analyses/858fb5cf-4923-4d33-9850-a445d6aa2ae8

┌──────┬───────────────────────────┬──────────┬───────────────────────────┐
│ Line │ SWC Title                 │ Severity │ Short Description         │
├──────┼───────────────────────────┼──────────┼───────────────────────────┤
│    3 │ (SWC-103) Floating Pragma │ Low      │ A floating pragma is set. │
└──────┴───────────────────────────┴──────────┴───────────────────────────┘
```

src/core/option/OptionManager.sol



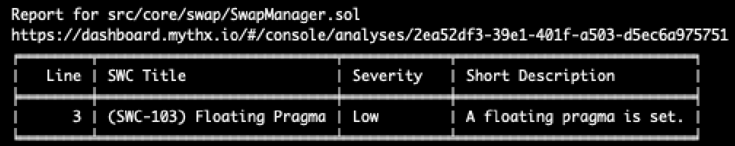
```
Report for src/core/option/OptionManager.sol
https://dashboard.mythx.io/#/console/analyses/0c578940-0ac0-491a-885a-3de037d1385d

┌──────┬───────────────────────────┬──────────┬───────────────────────────┐
│ Line │ SWC Title                 │ Severity │ Short Description         │
├──────┼───────────────────────────┼──────────┼───────────────────────────┤
│    3 │ (SWC-103) Floating Pragma │ Low      │ A floating pragma is set. │
└──────┴───────────────────────────┴──────────┴───────────────────────────┘
```

src/core/swap/Swap.sol



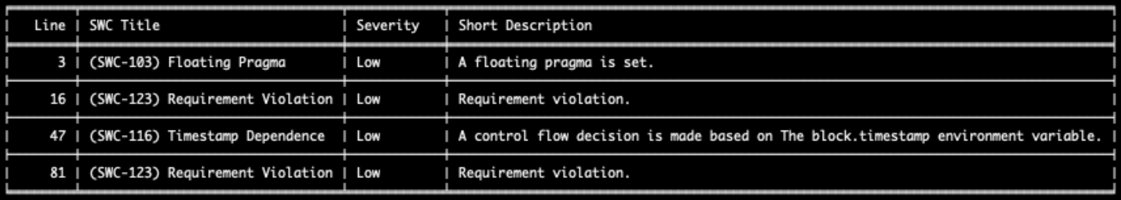
```
Report for src/core/swap/Swap.sol
https://dashboard.mythx.io/#/console/analyses/ab92ba65-a8b9-487d-aee2-9df8723457d4

┌──────┬───────────────────────────┬──────────┬───────────────────────────┐
│ Line │ SWC Title                 │ Severity │ Short Description         │
├──────┼───────────────────────────┼──────────┼───────────────────────────┤
│    3 │ (SWC-103) Floating Pragma │ Low      │ A floating pragma is set. │
└──────┴───────────────────────────┴──────────┴───────────────────────────┘
```

src/core/swap/SwapManager.sol



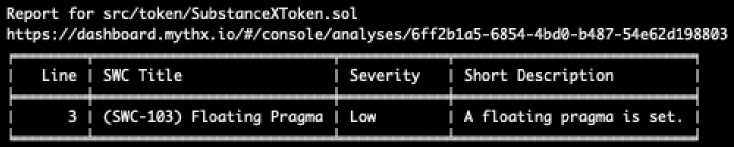
```
Report for src/core/swap/SwapManager.sol
https://dashboard.mythx.io/#/console/analyses/2ea52df3-39e1-401f-a503-d5ec6a975751

┌──────┬───────────────────────────┬──────────┬───────────────────────────┐
│ Line │ SWC Title                 │ Severity │ Short Description         │
├──────┼───────────────────────────┼──────────┼───────────────────────────┤
│    3 │ (SWC-103) Floating Pragma │ Low      │ A floating pragma is set. │
└──────┴───────────────────────────┴──────────┴───────────────────────────┘
```

src/token/GradualVester.sol



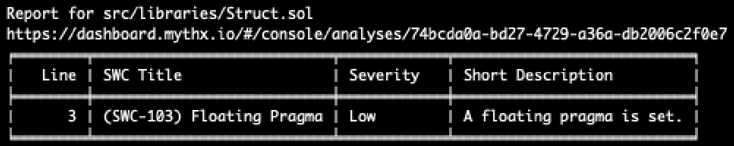
```
| Line | SWC Title                         | Severity | Short Description                                                        |
|------|-----------------------------------|----------|--------------------------------------------------------------------------|
|    3 | (SWC-103) Floating Pragma         | Low      | A floating pragma is set.                                                |
|------|-----------------------------------|----------|--------------------------------------------------------------------------|
|   16 | (SWC-123) Requirement Violation   | Low      | Requirement violation.                                                   |
|------|-----------------------------------|----------|--------------------------------------------------------------------------|
|   47 | (SWC-116) Timestamp Dependence    | Low      | A control flow decision is made based on The block.timestamp environment variable. |
|------|-----------------------------------|----------|--------------------------------------------------------------------------|
|   81 | (SWC-123) Requirement Violation   | Low      | Requirement violation.                                                   |
```

src/token/SubstanceXToken.sol



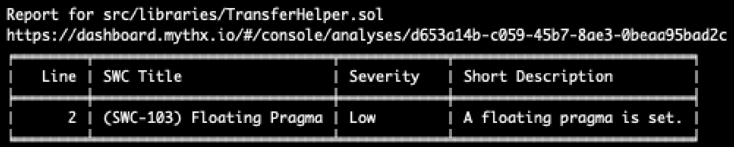
```
Report for src/token/SubstanceXToken.sol
https://dashboard.mythx.io/#/console/analyses/6ff2b1a5-6854-4bd0-b487-54e62d198803

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

src/libraries/Struct.sol



```
Report for src/libraries/Struct.sol
https://dashboard.mythx.io/#/console/analyses/74bcda0a-bd27-4729-a36a-db2006c2f0e7

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

src/libraries/TransferHelper.sol



```
Report for src/libraries/TransferHelper.sol
https://dashboard.mythx.io/#/console/analyses/d653a14b-c059-45b7-8ae3-0beaa95bad2c

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

MythX did not identify any vulnerabilities in the contracts.

The findings obtained as a result of the MythX scan were examined, and they were not included in the report as they were determined false positives.

THANK YOU FOR CHOOSING

HALBORN