MythX

REPORT 64A2C4466239860019FAF0AF

Created	Mon Jul 03 2023 12:51:18 GMT+0000 (Coordinated Universal Time)
Number of analyses	10
User	64a2b476f4bf587c5b592746

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
85ff7dbc-b398-4bd7-9c86-ede94d1387d1	contracts/OraclePoS.sol	0
cfb05e29-d267-43d8-b866-87e14c26089e	contracts/SilicaV2_1.sol	1
c061c5b6-9dc8-40a9-b942-f3362290a5f6	contracts/SilicaEthStaking.sol	1
0511bf8f-e8b5-47b7-bb4c-7bdbb183e867	contracts/Oracle.sol	0
0cdce906-2803-4348-b5fb-9154621caf91	contracts/OracleRegistry.sol	0
78593334-b6e3-4bb8-86ad-dcdee6b03c4a	contracts/OracleEthStaking.sol	1
885e72ef-8217-47a4-96cc-6d8fe49934dc	contracts/SwapProxy.sol	1
d4a30329-bdd9-4702-bfa5-11fbcc94ef06	contracts/AbstractSilicaV2_1.sol	1
5791bde5-5a6a-4a58-bafd-a4968ae0df9b	contracts/RewardsProxy.sol	2
753e80a6-8899-4583-bf26-e5c963a4b7f4	contracts/SilicaFactory.sol	1

Analysis 85ff7dbc-b398-4bd7-9c86-ede94d1387d1

MythX

Started	Mon Jul 03 2023 12:55:24 GMT+0000 (Coordinated Universal Time)
Finished	Mon Jul 03 2023 12:57:34 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.7.3
Main Source File	`Contracts/OraclePoS.Sol`

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	0

ISSUES

Analysis `cfb05e29-d267-43d8-b866-87e14c26089e`

MythX

Started	Mon Jul 03 2023 12:55:34 GMT+0000 (Coordinated Universal Time)
Finished	Mon Jul 03 2023 12:57:47 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.7.3
Main Source File	Contracts/SilicaV2_1.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	1

ISSUES

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.6"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

contracts/SilicaV2_1.sol

Locations

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.6;

import {AbstractSilicaV2_1} from "./AbstractSilicaV2_1.sol";
```

Analysis c061c5b6-9dc8-40a9-b942-f3362290a5f6 MythX

Started	Mon Jul 03 2023 12:55:34 GMT+0000 (Coordinated Universal Time)
Finished	Mon Jul 03 2023 12:57:51 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.7.3
Main Source File	Contracts/SilicaEthStaking.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	1

ISSUES

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.6"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

contracts/SilicaEthStaking.sol

Locations

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.6;

import {AbstractSilicaV2_1} from "./AbstractSilicaV2_1.sol";
```

Analysis `0511bf8f-e8b5-47b7-bb4c-7bdbb183e867` MythX

Started	Mon Jul 03 2023 12:55:34 GMT+0000 (Coordinated Universal Time)
Finished	Mon Jul 03 2023 12:57:44 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.7.3
Main Source File	`Contracts/Oracle.Sol`

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	0

ISSUES

Analysis `0cdce906-2803-4348-b5fb-9154621caf91` MythX

Started	Mon Jul 03 2023 12:55:34 GMT+0000 (Coordinated Universal Time)
Finished	Mon Jul 03 2023 12:57:40 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.7.3
Main Source File	`Contracts/OracleRegistry.Sol`

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	0

ISSUES

Analysis 78593334-b6e3-4bb8-86ad-dcdee6b03c4a

MythX

Started	Mon Jul 03 2023 12:55:45 GMT+0000 (Coordinated Universal Time)
Finished	Mon Jul 03 2023 12:57:54 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.7.3
Main Source File	`Contracts/OracleEthStaking.Sol`

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	1

ISSUES

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.6"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

`contracts/OracleEthStaking.sol`

Locations

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.6;

import "@openzeppelin/contracts/access/AccessControl.sol";
```

Analysis 885e72ef-8217-47a4-96cc-6d8fe49934dc MythX

Started	Mon Jul 03 2023 12:55:45 GMT+0000 (Coordinated Universal Time)
Finished	Mon Jul 03 2023 12:57:49 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.7.3
Main Source File	Contracts/SwapProxy.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	1

ISSUES

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.6"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

contracts/SwapProxy.sol

Locations

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.6;

import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
```

Analysis d4a30329-bdd9-4702-bfa5-11fbcc94ef06 MythX

Started	Mon Jul 03 2023 12:55:55 GMT+0000 (Coordinated Universal Time)
Finished	Mon Jul 03 2023 12:58:05 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.7.3
Main Source File	Contracts/AbstractSilicaV2_1.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	1

ISSUES

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.6"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

contracts/AbstractSilicaV2_1.sol

Locations

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.6;

import {ISilicaV2_1} from "./interfaces/silica/ISilicaV2_1.sol";
```

Analysis 5791bde5-5a6a-4a58-bafd-a4968ae0df9b

MythX

Started	Mon Jul 03 2023 12:55:55 GMT+0000 (Coordinated Universal Time)
Finished	Mon Jul 03 2023 12:56:02 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.7.3
Main Source File	Contracts/RewardsProxy.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	2

ISSUES

UNKNOWN Arithmetic operation "++" discovered

SWC-101

This plugin produces issues to support false positive discovery within MythX.

Source file

contracts/RewardsProxy.sol

Locations

```
function streamRewards(StreamRequest[] calldata streamRequests) external override {
uint256 numRequest = streamRequests.length;
for (uint256 i = 0; i < numRequest; ++i) {
streamReward(streamRequests[i]);
}
```

UNKNOWN Arithmetic operation "+" discovered

SWC-101

This plugin produces issues to support false positive discovery within MythX.

Source file

node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol

Locations

```
function safeIncreaseAllowance(IERC20 token, address spender, uint256 value) internal {
uint256 oldAllowance = token.allowance(address(this), spender);
_callOptionalReturn(token, abi.encodeWithSelector(token.approve.selector, spender, oldAllowance + value));
}
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

`node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol`

Locations

```
uint256 oldAllowance = token.allowance(address(this), spender);
require(oldAllowance >= value, "SafeERC20: decreased allowance below zero");
_callOptionalReturn(token, abi.encodeWithSelector(token.approve.selector, spender, oldAllowance - value));
}
}
```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

`node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol`

Locations

```
token.permit(owner, spender, value, deadline, v, r, s);
uint256 nonceAfter = token.nonces(owner);
require(nonceAfter == nonceBefore + 1, "SafeERC20: permit did not succeed");
}
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.6"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

`contracts/RewardsProxy.sol`

Locations

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.6;

import "./interfaces/rewardsProxy/IRewardsProxy.sol";
```

LOW

SWC-108

State variable visibility is not set.

It is best practice to set the visibility of state variables explicitly. The default visibility for "oracleRegistry" is internal. Other possible visibility settings are public and private.

Source file

`contracts/RewardsProxy.sol`

Locations

```
*/
contract RewardsProxy is IRewardsProxy {
IOracleRegistry immutable oracleRegistry;

constructor(address _oracleRegistry) {
```

UNKNOWN

SWC-110

Out of bounds array access

The index access expression can cause an exception in case of use of invalid array index value.

Source file

`contracts/RewardsProxy.sol`

Locations

```
uint256 numRequest = streamRequests.length;
for (uint256 i = 0; i < numRequest; ++i) {
streamReward(streamRequests[i]);
}
emit RewardsStreamed(streamRequests);
```

Analysis 753e80a6-8899-4583-bf26-e5c963a4b7f4

MythX

Started	Mon Jul 03 2023 12:55:55 GMT+0000 (Coordinated Universal Time)
Finished	Mon Jul 03 2023 12:56:15 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Cli-0.7.3
Main Source File	Contracts/SilicaFactory.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	1

ISSUES

UNKNOWN
SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file
contracts/SilicaFactory.sol
Locations

```
uint256 numDeposits = getNumDeposits(oracleData.lastIndexedDay, lastDueDay);
return (hashrate * oracleData.networkReward * numDeposits) / (oracleData.networkHashrate * 10);
}
```

UNKNOWN
SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file
contracts/SilicaFactory.sol
Locations

```
uint256 numDeposits = getNumDeposits(oracleData.lastIndexedDay, lastDueDay);
return (hashrate * oracleData.networkReward * numDeposits) / (oracleData.networkHashrate * 10);
}
```

UNKNOWN Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/SilicaFactory.sol

Locations

```

uint256 numDeposits = getNumDeposits(oracleData.lastIndexedDay, lastDueDay);
return (hashrate * oracleData.networkReward * numDeposits) / (oracleData.networkHashrate * 10);
}
```

UNKNOWN Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/SilicaFactory.sol

Locations

```

uint256 numDeposits = getNumDeposits(oracleData.lastIndexedDay, lastDueDay);
return (hashrate * oracleData.networkReward * numDeposits) / (oracleData.networkHashrate * 10);
}
```

UNKNOWN Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/SilicaFactory.sol

Locations

```
OracleEthStakingData memory oracleData = getOracleEthStakingData(rewardTokenAddress);
uint256 numDeposits = getNumDeposits(oracleData.lastIndexedDay, lastDueDay);
return (oracleData.baseRewardPerIncrementPerDay * stakedAmount * numDeposits) / (10**(decimals + 1));
}
```

UNKNOWN Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/SilicaFactory.sol

Locations

```
OracleEthStakingData memory oracleData = getOracleEthStakingData(rewardTokenAddress);
uint256 numDeposits = getNumDeposits(oracleData.lastIndexedDay, lastDueDay);
return (oracleData.baseRewardPerIncrementPerDay * stakedAmount * numDeposits) / (10**(decimals + 1));
}
```

UNKNOWN SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

contracts/SilicaFactory.sol

Locations

```
OracleEthStakingData memory oracleData = getOracleEthStakingData(rewardTokenAddress);
uint256 numDeposits = getNumDeposits(oracleData.lastIndexedDay, lastDueDay);
return (oracleData.baseRewardPerIncrementPerDay * stakedAmount * numDeposits) / (10**(decimals + 1));
}
```

UNKNOWN SWC-101

Arithmetic operation "**" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

contracts/SilicaFactory.sol

Locations

```
OracleEthStakingData memory oracleData = getOracleEthStakingData(rewardTokenAddress);
uint256 numDeposits = getNumDeposits(oracleData.lastIndexedDay, lastDueDay);
return (oracleData.baseRewardPerIncrementPerDay * stakedAmount * numDeposits) / (10**(decimals + 1));
}
```

UNKNOWN SWC-101

Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

contracts/SilicaFactory.sol

Locations

```
OracleEthStakingData memory oracleData = getOracleEthStakingData(rewardTokenAddress);
uint256 numDeposits = getNumDeposits(oracleData.lastIndexedDay, lastDueDay);
return (oracleData.baseRewardPerIncrementPerDay * stakedAmount * numDeposits) / (10**(decimals + 1));
}
```

UNKNOWN SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

contracts/SilicaFactory.sol

Locations

```
) internal pure returns (uint256) {
uint256 numDeposits = getNumDeposits(oracleData.lastIndexedDay, lastDueDay);
return (hashrate * oracleData.networkReward * numDeposits) / (oracleData.networkHashrate * 10);
}
```

UNKNOWN Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/SilicaFactory.sol

Locations

```
) internal pure returns (uint256) {
uint256 numDeposits = getNumDeposits(oracleData.lastIndexedDay, lastDueDay);
return (hashrate * oracleData.networkReward * numDeposits) / (oracleData.networkHashrate * 10);
}
```

UNKNOWN Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/SilicaFactory.sol

Locations

```
) internal pure returns (uint256) {
uint256 numDeposits = getNumDeposits(oracleData.lastIndexedDay, lastDueDay);
return (hashrate * oracleData.networkReward * numDeposits) / (oracleData.networkHashrate * 10);
}
```

UNKNOWN Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/SilicaFactory.sol

Locations

```
) internal pure returns (uint256) {
uint256 numDeposits = getNumDeposits(oracleData.lastIndexedDay, lastDueDay);
return (hashrate * oracleData.networkReward * numDeposits) / (oracleData.networkHashrate * 10);
}
```

UNKNOWN Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/SilicaFactory.sol

Locations

```
) internal pure returns (uint256) {
uint256 numDeposits = getNumDeposits(oracleData.lastIndexedDay, lastDueDay);
return (oracleData.baseRewardPerIncrementPerDay * stakedAmount * numDeposits) / (10**(decimals + 1));
}
```

UNKNOWN SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

contracts/SilicaFactory.sol

Locations

```
) internal pure returns (uint256) {
uint256 numDeposits = getNumDeposits(oracleData.lastIndexedDay, lastDueDay);
return (oracleData.baseRewardPerIncrementPerDay * stakedAmount * numDeposits) / (10**(decimals + 1));
}
```

UNKNOWN SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

contracts/SilicaFactory.sol

Locations

```
) internal pure returns (uint256) {
uint256 numDeposits = getNumDeposits(oracleData.lastIndexedDay, lastDueDay);
return (oracleData.baseRewardPerIncrementPerDay * stakedAmount * numDeposits) / (10**(decimals + 1));
}
```

UNKNOWN SWC-101

Arithmetic operation "**" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

contracts/SilicaFactory.sol

Locations

```
) internal pure returns (uint256) {
uint256 numDeposits = getNumDeposits(oracleData.lastIndexedDay, lastDueDay);
return (oracleData.baseRewardPerIncrementPerDay * stakedAmount * numDeposits) / (10**(decimals + 1));
}
```

UNKNOWN SWC-101

Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

contracts/SilicaFactory.sol

Locations

```
) internal pure returns (uint256) {
uint256 numDeposits = getNumDeposits(oracleData.lastIndexedDay, lastDueDay);
return (oracleData.baseRewardPerIncrementPerDay * stakedAmount * numDeposits) / (10**(decimals + 1));
}
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

contracts/SilicaFactory.sol

Locations

```
/// @dev lastDueDay is always greater than lastIndexedDay
function getNumDeposits(uint256 lastIndexedDay, uint256 lastDueDay) internal pure returns (uint256) {
return lastDueDay - lastIndexedDay - 1;
} // Is this the same for day 0?
//underscore internal functions
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

contracts/SilicaFactory.sol

Locations

```
/// @dev lastDueDay is always greater than lastIndexedDay
function getNumDeposits(uint256 lastIndexedDay, uint256 lastDueDay) internal pure returns (uint256) {
return lastDueDay - lastIndexedDay - 1;
} // Is this the same for day 0?
//underscore internal functions
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

contracts/SilicaFactory.sol

Locations

```

OracleData memory oracleData = _getOracleData(_rewardTokenAddress);
uint256 collateralAmount = getMiningSwapCollateralRequirement(
_lastDueDay,
_resourceAmount,
oracleData
) * (10 + _additionalCollateralPercent) / 10; // Check this maths

ISilicaV2_1.InitializeData memory initializeData;
```

UNKNOWN

SWC-101

Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

contracts/SilicaFactory.sol

Locations

```

OracleData memory oracleData = _getOracleData(_rewardTokenAddress);
uint256 collateralAmount = getMiningSwapCollateralRequirement(
_lastDueDay,
_resourceAmount,
oracleData
) * (10 + _additionalCollateralPercent) / 10; // Check this maths

ISilicaV2_1.InitializeData memory initializeData;
```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

contracts/SilicaFactory.sol

Locations

```
_resourceAmount,
oracleData
) * (10 + _additionalCollateralPercent) / 10; // Check this maths

ISilicaV2_1.InitializeData memory initializeData;
```

UNKNOWN

SWC-101

Arithmetic operation "/" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

contracts/SilicaFactory.sol

Locations

```
ISilicaV2_1 newSilicaV2 = ISilicaV2_1(newContractAddress);
OracleEthStakingData memory oracleData = getOracleEthStakingData(_rewardTokenAddress);
uint256 collateralAmount = getEthStakingCollateralRequirement(
_lastDueDay,
_resourceAmount,
oracleData,
newSilicaV2.getDecimals()
) * (10 + _additionalCollateralPercent) / 10;

ISilicaV2_1.InitializeData memory initializeData;
```

UNKNOWN Arithmetic operation "*" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/SilicaFactory.sol

Locations

```
ISilicaV2_1 newSilicaV2 = ISilicaV2_1(newContractAddress);
OracleEthStakingData memory oracleData = getOracleEthStakingData(_rewardTokenAddress);
uint256 collateralAmount = getEthStakingCollateralRequirement(
_lastDueDay,
_resourceAmount,
oracleData,
newSilicaV2.getDecimals()
) * (10 + _additionalCollateralPercent) / 10;

ISilicaV2_1.InitializeData memory initializeData;
```

UNKNOWN Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/SilicaFactory.sol

Locations

```
oracleData,
newSilicaV2.getDecimals()
) * (10 + _additionalCollateralPercent) / 10;

ISilicaV2_1.InitializeData memory initializeData;
```

UNKNOWN Compiler-rewritable "<uint> - 1" discovered

This plugin produces issues to support false positive discovery within MythX.

SWC-101

Source file

contracts/SilicaFactory.sol

Locations

```
/// @dev lastDueDay is always greater than lastIndexedDay
function getNumDeposits(uint256 lastIndexedDay, uint256 lastDueDay) internal pure returns (uint256) {
return lastDueDay - lastIndexedDay - 1;
} // Is this the same for day 0?
//underscore internal functions
```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

`node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol`

Locations

```
function increaseAllowance(address spender, uint256 addedValue) public virtual returns (bool) {
address owner = _msgSender();
_approve(owner, spender, allowance(owner, spender) + addedValue);
return true;
}
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

`node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol`

Locations

```
require(currentAllowance >= subtractedValue, "ERC20: decreased allowance below zero");
unchecked {
_approve(owner, spender, currentAllowance - subtractedValue);
}
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

`node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol`

Locations

```
require(fromBalance >= amount, "ERC20: transfer amount exceeds balance");
unchecked {
_balances[from] = fromBalance - amount;
// Overflow not possible: the sum of all balances is capped by totalSupply, and the sum is preserved by
// decrementing then incrementing.
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

`node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol`

Locations

```
// Overflow not possible: the sum of all balances is capped by totalSupply, and the sum is preserved by
// decrementing then incrementing.
_balances[to] += amount;
}
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
_beforeTokenTransfer(address(0), account, amount);

_totalSupply += amount;
unchecked {
// Overflow not possible: balance + amount is at most totalSupply + amount, which is checked above.
```

UNKNOWN

SWC-101

Arithmetic operation "+=" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
unchecked {
// Overflow not possible: balance + amount is at most totalSupply + amount, which is checked above.
_balances[account] += amount;
}
emit Transfer(address(0), account, amount);
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
require(accountBalance >= amount, "ERC20: burn amount exceeds balance");
unchecked {
_balances[account] = accountBalance - amount;
// Overflow not possible: amount <= accountBalance <= totalSupply.
_totalSupply -= amount;
```

UNKNOWN

SWC-101

Arithmetic operation "-=" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
_balances[account] = accountBalance - amount;
// Overflow not possible: amount <= accountBalance <= totalSupply.
_totalSupply -= amount;
}
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol

Locations

```
require(currentAllowance >= amount, "ERC20: insufficient allowance");
unchecked {
_approve(owner, spender, currentAllowance - amount);
}
}
```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol

Locations

```
function safeIncreaseAllowance(IERC20 token, address spender, uint256 value) internal {
uint256 oldAllowance = token.allowance(address(this), spender);
_callOptionalReturn(token, abi.encodeWithSelector(token.approve.selector, spender, oldAllowance + value));
}
```

UNKNOWN

SWC-101

Arithmetic operation "-" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol

Locations

```
uint256 oldAllowance = token.allowance(address(this), spender);
require(oldAllowance >= value, "SafeERC20: decreased allowance below zero");
_callOptionalReturn(token, abi.encodeWithSelector(token.approve.selector, spender, oldAllowance - value));
}
}
```

UNKNOWN

SWC-101

Arithmetic operation "+" discovered

This plugin produces issues to support false positive discovery within MythX.

Source file

node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol

Locations

```
token.permit(owner, spender, value, deadline, v, r, s);
uint256 nonceAfter = token.nonces(owner);
require(nonceAfter == nonceBefore + 1, "SafeERC20: permit did not succeed");
}
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.6"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

contracts/SilicaFactory.sol

Locations

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.6;

import "@openzeppelin/contracts/proxy/Clones.sol";
```