MythX

REPORT 600BD6A003C6340011685134

Created	Sat Jan 23 2021 07:56:16 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	bitedama@gmail.com

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
09c2594e-d22f-4f90-9213-a01c3f0a663a	/contracts/treasury.sol	9

Analysis 09c2594e-d22f-4f90-9213-a01c3f0a663a

MythX

Started	Sat Jan 23 2021 07:56:18 GMT+0000 (Coordinated Universal Time)
Finished	Sat Jan 23 2021 08:11:33 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Client Tool	Mythx-Vscode-Extension
Main Source File	/Contracts/Treasury.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	7	2

ISSUES

MEDIUM SWC-000

Incorrect function "_getCashPrice" state mutability

Function "_getCashPrice" state mutability is considered "view" by compiler, but should be set to non-payable (default).

Source file
/contracts/treasury.sol
Locations

```
return price;
} catch {
revert('Treasury: failed to consult cash price from the oracle');
}
}


function estimatedCashPrice() public view returns (uint256) {
try IOracle(seigniorageOracle).consultNow(cash, 1e18) returns (uint256 price) {
return price;
} catch {
revert('Treasury: failed to consult cash price from the oracle');
}
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "getReserve" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

/contracts/treasury.sol

Locations

```
// oracle
function getBondOraclePrice() public view returns (uint256) {
return _getCashPrice(bondOracle);
}

function getSeigniorageOraclePrice() public view returns (uint256) {
return _getCashPrice(seigniorageOracle);
}
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "getBondOraclePrice" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

/contracts/treasury.sol

Locations

```

function getSeigniorageOraclePrice() public view returns (uint256) {
return _getCashPrice(seigniorageOracle);
}

function _getCashPrice(address oracle) internal view returns (uint256) {
try IOracle(oracle).consult(cash, 1e18) returns (uint256 price) {
return price;
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "getSeigniorageOraclePrice" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

/contracts/treasury.sol

Locations

```
}

function _getCashPrice(address oracle) internal view returns (uint256) {
try IOracle(oracle).consult(cash, 1e18) returns (uint256 price) {
return price;
} catch {
revert('Treasury: failed to consult cash price from the oracle');
}
}
```

MEDIUM

SWC-000

Incorrect function "estimatedCashPrice" state mutability

Function "estimatedCashPrice" state mutability is considered "view" by compiler, but should be set to non-payable (default).

Source file

`/contracts/treasury.sol`

Locations

```
return price;
} catch {
revert('Treasury: failed to consult cash price from the oracle');
}
}



/* ========== GOVERNANCE ========== */

function initialize() public checkOperator {
require(!initialized, 'Treasury: initialized');

// burn all of it's balance
IBasisAsset(cash).burn(IERC20(cash).balanceOf(address(this)));

// set accumulatedSeigniorage to it's balance
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "initialize" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

`/contracts/treasury.sol`

Locations

```

// burn all of it's balance
IBasisAsset(cash).burn(IERC20(cash).balanceOf(address(this)));

// set accumulatedSeigniorage to it's balance
accumulatedSeigniorage = IERC20(cash).balanceOf(address(this));

initialized = true;
emit Initialized(msg.sender, block.number);
}

function migrate(address target) public onlyOperator checkOperator {
require(!migrated, 'Treasury: migrated');

// cash
Operator(cash).transferOperator(target);
Operator(cash).transferOwnership(target);
IERC20(cash).transfer(target, IERC20(cash).balanceOf(address(this)));
```

MEDIUM

SWC-000

Function could be marked as external.

The function definition of "migrate" is marked "public". However, it is never directly called by another function in the same contract or in any of its descendants. Consider to mark it as "external" instead.

Source file

`/contracts/treasury.sol`

Locations

```
// cash
Operator(cash).transferOperator(target);
Operator(cash).transferOwnership(target);
IERC20(cash).transfer(target, IERC20(cash).balanceOf(address(this)));

// bond
Operator(bond).transferOperator(target);
Operator(bond).transferOwnership(target);
IERC20(bond).transfer(target, IERC20(bond).balanceOf(address(this)));

// share
Operator(share).transferOperator(target);
Operator(share).transferOwnership(target);
IERC20(share).transfer(target, IERC20(share).balanceOf(address(this)));

migrated = true;
emit Migration(target);
}

/* ========== MUTABLE FUNCTIONS ========== */

function _updateCashPrice() internal {
try IOracle(bondOracle).update() {} catch {}
try IOracle(seigniorageOracle).update() {} catch {}
}
```

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.6.0"". 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/treasury.sol`

Locations

```
pragma solidity ^0.6.0;


import '@openzeppelin/contracts/math/Math.sol';
```

LOW

SWC-120

Potential use of "block.number" as source of randonmness.

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source file

`/contracts/treasury.sol`

Locations

```

// cash
Operator(cash).transferOperator(target);
Operator(cash).transferOwnership(target);
IERC20(cash).transfer(target, IERC20(cash).balanceOf(address(this)));
```