# Euler

## Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **May 8th, - June 4th, 2021**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY 4
CONTACTS 4
1 EXECUTIVE OVERVIEW 5
1.1 INTRODUCTION 6
1.2 AUDIT SUMMARY 6
1.3 TEST APPROACH & METHODOLOGY 7
RISK METHODOLOGY 8
1.4 SCOPE 10
2 ASSESSMENT SUMMARY & FINDINGS OVERVIEW 11
3 FINDINGS & TECH DETAILS 12
3.1 (HAL-01) FLOATING PRAGMA - LOW 14
Description 14
Code Location 14
Risk Level 18
Recommendations 18
Remediation Plan 18
3.2 (HAL-02) USE OF BLOCK.TIMESTAMP - LOW 19
Description 19
Code Location 19
Risk Level 21
Recommendation 21
Remediation Plan 21
3.3 (HAL-03) NO STORAGE REFUND WHEN EXITING THE MARKET - INFORMATIONAL 22
Description 22

Code Location 23
Risk Level 24
Recommendation 24
Remediation Plan 24
3.4 (HAL-04) INVALID CONSTANT VALUE - INFORMATIONAL 25
Description 25
Code Location 25
Risk Level 26
Recommendation 26
Remediation Plan 26
3.5 (HAL-05) INFINITE ALLOWANCE - INFORMATIONAL 26
Description 26
Code Location 26
Risk Level 27
Recommendations 27
Remediation Plan 27
4 MANUAL TESTING 28
Architecture overview 31
Proxy Calling contract 32
4.1 General issues while manually testing 32
BaseLogic 33
E-token deposit 34
Multiplication on division 34
Unchecked 34
5 AUTOMATED TESTING 37
5.1 STATIC ANALYSIS REPORT 40
Description 40

5.2 AUTOMATED SECURITY SCAN 51

MYTHX 51

Results 51

# DOCUMENT REVISION HISTORY

| VERSION | MODIFICATION | DATE | AUTHOR |
|---|---|---|---|
| 0.1 | Document Creation | 06/02/2021 | Ferran Celades |
| 0.2 | Document Updates | 06/04/2021 | Ferran Celades |
| 0.9 | Document Updates | 06/07/2021 | Gabi Urrutia |
| 1.0 | Final Draft | 06/07/2021 | Ferran Celades |
| 1.1 | Remediation Plan | 06/08/2021 | Ferran Celades |

# 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 |
| Ferran Celades | Halborn | ferran.celades@halborn.com |

# EXECUTIVE OVERVIEW

## 1.1 INTRODUCTION

Euler leverages the power of Uniswap's V3 time-weighted average price oracles to allow users to activate their own money markets. Euler is designed to allow anyone to publish any token that can be borrowed, earned interests over time and pay back the debts. Asset-specific collateral and borrow factors protect the integrity of the protocol and its users by tailoring the borrowing capacity of users to the risk factors associated with their collateral assets and those they wish to borrow. Euler introduces reactive interest rates, backed by control theory, to allow interest rates to rapidly adapt to market conditions in real-time.

Euler engaged Halborn to conduct a security assessment on their Smart contracts beginning on May 8th, 2021 and ending June 4th, 2021. The security assessment was scoped to the smart contract provided in the Github repository Euler Smart Contracts and an audit of the security risk and implications regarding the changes introduced by the development team at Euler prior to its production release shortly following the assessments deadline.

Though this security audit's outcome is satisfactory, only the most essential aspects were tested and verified to achieve objectives and deliverables set in the scope due to time and resource constraints. It is essential to note the use of the best practices for secure smart-contract development.

## 1.2 AUDIT SUMMARY

The team at Halborn was provided one month for the engagement and assigned two full time security engineers to audit the security of the smart contract. The security engineers are blockchain and smart-contract security experts with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this audit to achieve the following:

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

In summary, Halborn identified few security risks, and recommends performing further testing to validate extended safety and correctness in context to the whole set of contracts. External threats, such as economic attacks, oracle attacks, and inter-contract functions and calls should be validated for expected logic and state.

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

- Research into architecture and purpose.
- Smart Contract manual code review and walkthrough.
- Graphing out functionality and contract logic/connectivity/functions (solgraph)
- Manual Assessment of use and safety for the critical Solidity variables and functions in scope to identify any arithmetic related vulnerability classes (brownie console and manual deployments on Ganache)
- Manual testing by custom Python scripts.
- Scanning of solidity files for vulnerabilities, security hotspots or bugs. (MythX)
- Static Analysis of security for scoped contract, and imported functions. (Slither)
- Testnet deployment (Ganache, Remix IDE)

## RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident, and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. It's quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that was used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

**RISK SCALE - LIKELIHOOD**

5 - Almost certain an incident will occur.
4 - High probability of an incident occurring.
3 - Potential of a security incident in the long term.
2 - Low probability of an incident occurring.
1 - Very unlikely issue will cause an incident.

**RISK SCALE - IMPACT**

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.
3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.
1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

| CRITICAL | HIGH | MEDIUM | LOW | INFORMATIONAL |
|---|---|---|---|---|

**10** - CRITICAL
**9 - 8** - HIGH
**7 - 6** - MEDIUM
**5 - 4** - LOW

**3 - 1** - VERY LOW AND INFORMATIONAL

## 1.4 SCOPE

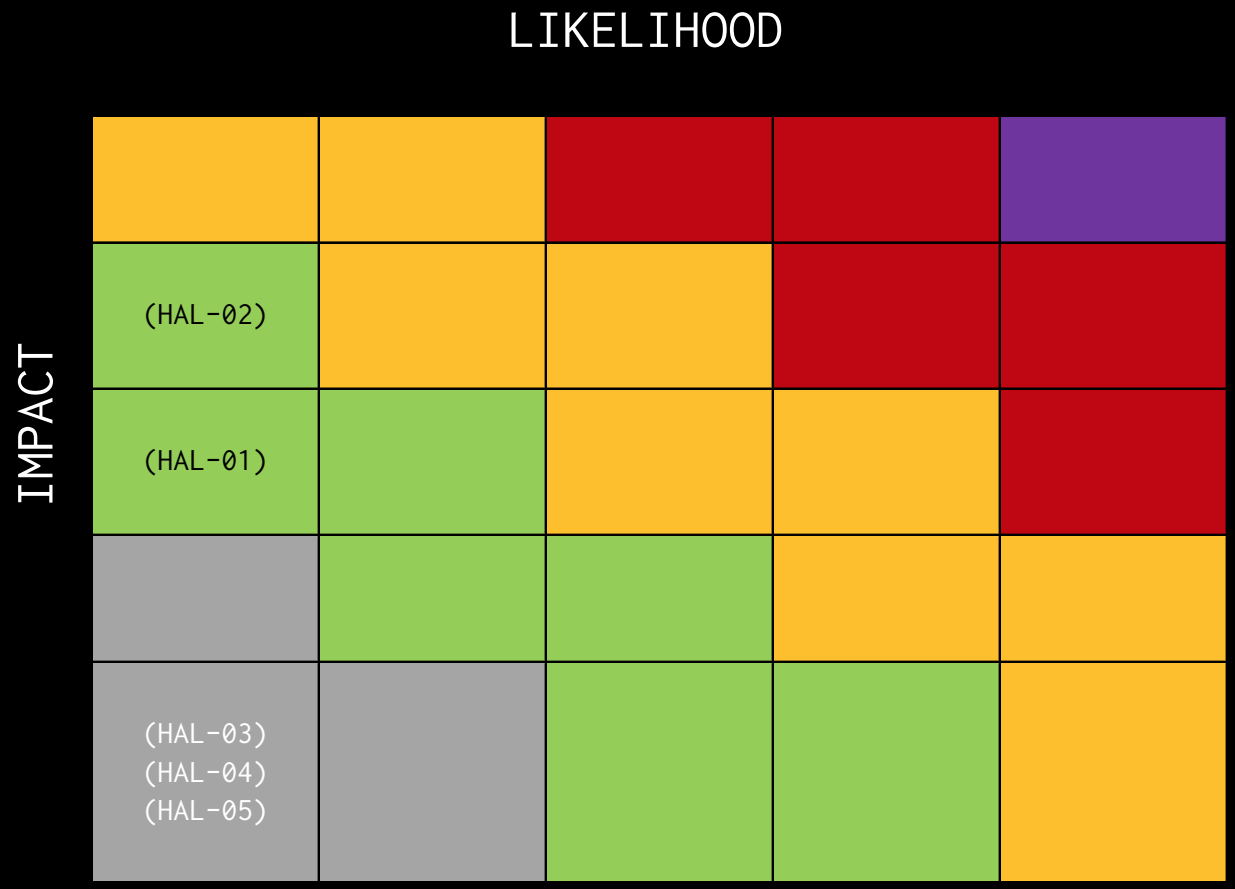
IN-SCOPE:
The security assessment was scoped to the smart contracts:
- Base.sol
- BaseIRM.sol
- BaseLogic.sol
- BaseModule.sol
- Constants.sol
- Euler.sol
- Events.sol
- Interfaces.sol
- Proxy.sol
- Storage.sol
- modules/DToken.sol
- modules/EToken.sol
- modules/Exec.sol
- modules/Governance.sol
- modules/Installer.sol
- modules/Liquidation.sol
- modules/Markets.sol
- modules/RiskManager.sol
- modules/interest-rate-models/IRMDefault.sol
- modules/interest-rate-models/IRMFixed.sol
- modules/interest-rate-models/IRMLinear.sol
- modules/interest-rate-models/IRMLinearRecursive.sol
- modules/interest-rate-models/IRMZero.sol

Commit ID: bd4153487e3c5a66cbf291260c9819a6bcb565d0

OUT-OF-SCOPE:
Economics attacks, external libraries and services.

# 2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

| CRITICAL | HIGH | MEDIUM | LOW | INFORMATIONAL |
|---|---|---|---|---|
| 0 | 0 | 0 | 2 | 3 |



## EXECUTIVE OVERVIEW

| SECURITY ANALYSIS | RISK LEVEL | REMEDIATION DATE |
|---|---|---|
| USE OF BLOCK.TIMESTAMP | Low | RISK ACCEPTED |
| FLOATING PRAGMA | Low | RISK ACCEPTED |
| NO STORAGE REFUND WHEN EXITING THE MARKET | Informational | ACKNOWLEDGED |
| INVALID CONSTANT VALUE | Informational | ACKNOWLEDGED |
| INFINITE ALLOWANCE | Informational | RISK ACCEPTED |

# FINDINGS & TECH DETAILS

## 3.1 (HAL-01) FLOATING PRAGMA - LOW

Description:

Some of the contracts use a floating pragma, such as ^0.8.0. Deploy contracts with the same compiler version and flags used during development and testing. Locking the pragma helps ensure that contracts do not accidentally get deployed using a different compiler specification. For example, an outdated compiler version might introduce bugs, or a new version that is not extensively tested may introduce security vulnerabilities.

Code Location:

**Listing 1: BaseLogic.sol (Lines 3)**

```
1 // SPDX-License-Identifier: UNLICENSED
2
3 pragma solidity ^0.8.0;
```

**Listing 2: BaseIRM.sol (Lines 3)**

```
1 // SPDX-License-Identifier: UNLICENSED
2
3 pragma solidity ^0.8.0;
```

**Listing 3: Constants.sol (Lines 3)**

```
1 // SPDX-License-Identifier: UNLICENSED
2
3 pragma solidity ^0.8.0;
```

**Listing 4: Interfaces.sol (Lines 3)**

```
1 // SPDX-License-Identifier: UNLICENSED
2
3 pragma solidity ^0.8.0;
```

**Listing 5: Euler.sol (Lines 3)**

```
1 // SPDX-License-Identifier: UNLICENSED
2
3 pragma solidity ^0.8.0;
```

**Listing 6: Proxy.sol (Lines 3)**

```
1 // SPDX-License-Identifier: UNLICENSED
2
3 pragma solidity ^0.8.0;
```

**Listing 7: Storage.sol (Lines 3)**

```
1 // SPDX-License-Identifier: UNLICENSED
2
3 pragma solidity ^0.8.0;
```

**Listing 8: BaseModule.sol (Lines 3)**

```
1 // SPDX-License-Identifier: UNLICENSED
2
3 pragma solidity ^0.8.0;
```

**Listing 9: Events.sol (Lines 3)**

```
1 // SPDX-License-Identifier: UNLICENSED
2
3 pragma solidity ^0.8.0;
```

**Listing 10: Base.sol (Lines 3)**

```
1 // SPDX-License-Identifier: UNLICENSED
2
3 pragma solidity ^0.8.0;
```

**Listing 11: views/EulerGeneralView.sol (Lines 3)**

```
1 // SPDX-License-Identifier: UNLICENSED
2
3 pragma solidity ^0.8.0;
```

**Listing 12: vendor/TickMath.sol (Lines 9)**

```
5 // Updated to Solidity 0.8 by Euler:
6 //   * Cast MAX_TICK to int256 before casting to uint
7 //   * Wrapped function bodies with "unchecked {}" so as to not
      add any extra gas costs
8
9 pragma solidity ^0.8.0;
```

**Listing 13: modules/Exec.sol (Lines 3)**

```
1 // SPDX-License-Identifier: UNLICENSED
2
3 pragma solidity ^0.8.0;
```

**Listing 14: modules/Installer.sol (Lines 3)**

```
1 // SPDX-License-Identifier: UNLICENSED
2
3 pragma solidity ^0.8.0;
```

**Listing 15: modules/Markets.sol (Lines 3)**

```
1 // SPDX-License-Identifier: UNLICENSED
2
3 pragma solidity ^0.8.0;
```

**Listing 16: modules/EToken.sol (Lines 3)**

```
1 // SPDX-License-Identifier: UNLICENSED
2
3 pragma solidity ^0.8.0;
```

**Listing 17: modules/RiskManager.sol (Lines 3)**

```
1 // SPDX-License-Identifier: UNLICENSED
2
3 pragma solidity ^0.8.0;
```

**Listing 18: modules/DToken.sol (Lines 3)**

```
1 // SPDX-License-Identifier: UNLICENSED
2
3 pragma solidity ^0.8.0;
```

**Listing 19: modules/Governance.sol (Lines 3)**

```
1 // SPDX-License-Identifier: UNLICENSED
2
3 pragma solidity ^0.8.0;
```

**Listing 20: modules/Liquidation.sol (Lines 3)**

```
1 // SPDX-License-Identifier: UNLICENSED
2
3 pragma solidity ^0.8.0;
```

**Listing 21: modules/interest-rate-models/IRMLinearRecursive.sol (Lines 3)**

```
1 // SPDX-License-Identifier: UNLICENSED
2
3 pragma solidity ^0.8.0;
```

**Listing 22: modules/interest-rate-models/IRMZero.sol (Lines 3)**

```
1 // SPDX-License-Identifier: UNLICENSED
2
3 pragma solidity ^0.8.0;
```

**Listing 23: modules/interest-rate-models/IRMLinear.sol (Lines 3)**

```
1 // SPDX-License-Identifier: UNLICENSED
2
3 pragma solidity ^0.8.0;
```

**Listing 24: modules/interest-rate-models/IRMFixed.sol (Lines 3)**

```
1 // SPDX-License-Identifier: UNLICENSED
2
3 pragma solidity ^0.8.0;
```

**Listing 25: modules/interest-rate-models/IRMDefault.sol (Lines 1)**

```
1 pragma solidity ^0.8.0;
```

Risk Level:

**Likelihood - 1**
**Impact - 3**

Recommendations:

Lock the pragma version whenever possible and avoid using a floating pragma in the final deployment. The pragma can be locked in the code by removing the caret (^) and by specifying the exact version in the Truffle configuration file truffle-config.js or hardhat.config.js if using the HardHat framework.

Remediation Plan:

RISK ACCEPTED: Euler team accepted this risk. During development it's easier to upgrade the compiler if it's floating, and post-deployment it obviously can't be changed anyway, so they will fixate the compiler version when deploying.

## 3.2 (HAL-02) USE OF BLOCK.TIMESTAMP - LOW

Description:

During a manual static review, the tester noticed the use of block.timestamp in BaseLogic, RiskManager, Markets and Liquidation contract. The global variable block.timestamp does not necessarily hold the current time, and may not be accurate. Miners can influence the value of block.timestamp to perform Maximal Extractable Value (MEV) attacks. There is no guarantee that the value is correct, only that it is higher than the previous block's timestamp.

Code Location:

**Listing 26: Liquidation.sol (Lines 191)**

```
189         if (bonus > 1e18) bonus = 1e18;
190
191         bonus = bonus * (block.timestamp - lastActivity) /
                BONUS_REFRESH_PERIOD;
192         if (bonus > 1e18) bonus = 1e18;
```

**Listing 27: BaseLogic.sol (Lines 27)**

```
25      function updateLastActivity(address account) internal {
26          uint lastActivity = accountLookup[account].lastActivity;
27          if (lastActivity != 0 && lastActivity != block.timestamp)
                accountLookup[account].lastActivity = uint40(block.
                timestamp);
28      }
```

**Listing 28: BaseLogic.sol (Lines 163,166)**

```
161     // Update interest accumulator and reserves
162
163     if (block.timestamp != assetCache.
            lastInterestAccumulatorUpdate) {
```

```
164         dirty = true;
165
166         uint deltaT = block.timestamp - assetCache.
                lastInterestAccumulatorUpdate;
167
168         // Compute new values
```

**Listing 29: BaseLogic.sol (Lines 193)**

```
191     assetCache.totalBorrows = encodeDebtAmount(newTotalBorrows);
192     assetCache.interestAccumulator = newInterestAccumulator;
193     assetCache.lastInterestAccumulatorUpdate = uint40(block.
            timestamp);
194
195     if (newTotalBalances != assetCache.totalBalances) {
```

**Listing 30: RiskManager.sol (Lines 146)**

```
144     // Call observe() again to get the oldest available
145
146     ago = block.timestamp - oldestAvailableAge;
147     secondsAgos[0] = uint32(ago);
148
```

**Listing 31: Markets.sol (Lines 57)**

```
55     assetStorage.dTokenAddress = childDToken;
56
57     assetStorage.lastInterestAccumulatorUpdate = uint40(block.
           timestamp);
58     assetStorage.underlyingDecimals = decimals;
59     assetStorage.interestRateModel = uint32(MODULEID__IRM_DEFAULT)
           ;
```

**Listing 32: Markets.sol (Lines 162)**

```
160     address msgSender = unpackTrailingParamMsgSender();
161     address account = getSubAccount(msgSender, subAccountId);
162     accountLookup[account].lastActivity = uint40(block.timestamp);
163 }
```

Risk Level:

**Likelihood - 1**
**Impact - 4**

Recommendation:

Use `block.number` instead of `block.timestamp` to reduce the risk of MEV attacks. If possible, use an oracle.

Remediation Plan:

RISK ACCEPTED: This is part of our system's design. For instance see the "Compounding Behaviour" in the arch doc. I don't believe miners can extract any value by manipulating the timestamp within the network's allowable range.

## 3.3 (HAL-03) NO STORAGE REFUND WHEN EXITING THE MARKET - INFORMATIONAL

Description:

During a manual static review, the tester noticed that the doExitMarket function on BaseLogic does not zero out the storage when numMarketsEntered is set to 1. For the showcase, the code shown in Listing **??** was written. The **??** shows that the value was not zeroed out when exiting the last market.

**Listing 33: Code used to showcase memory not zeroed out**

```
1 function halborn_market(uint subAccountId, uint256 index) external
      returns (address){
2     return halborn_marketbase(subAccountId, index);
3 }
4
5 function halborn_marketbase(uint subAccountId, uint256 index)
     public returns (address){
6     address msgSender = unpackTrailingParamMsgSender();
7     address account = getSubAccount(msgSender, subAccountId);
8     address[MAX_POSSIBLE_ENTERED_MARKETS] storage markets =
          marketsEntered[account];
9     return _getEnteredMarketIndex(account, markets, index);
10 }
```

```
<Transaction '0x8a906e07469d7f90875d65a096c897d39abaac7ba796c284a4f119577a3bfd95'>
>>> markets.halborn_market(0, 0, {'from':deployer}).return_value
Transaction sent: 0x2acd1e75f8858bee44282e456a9ce855a7928b42aee5b7ae3ce46685e76fcc5d
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 17
  Markets.halborn_market confirmed - Block: 18   Gas used: 25892 (0.22%)

'0x3194cBDC3dbcd3E11a07892e7bA5c3394048Cc87'
>>> markets.halborn_market(0, 1, {'from':deployer}).return_value
Transaction sent: 0x71e23474b8d66c287dc459c75d91e767f5dc0a8e8f422a1e1ff391f37b40fd89
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 18
  Markets.halborn_market confirmed - Block: 19   Gas used: 25830 (0.22%)

'0x0000000000000000000000000000000000000000'
>>> markets.exitMarket(0, hal_token, {'from':deployer})
Transaction sent: 0x3c1950f0ee56e18ac1db4e34194ea5943e2f975e29e221f963b29dbcdc225ea5
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 19
  Markets.exitMarket confirmed - Block: 20   Gas used: 38981 (0.32%)

<Transaction '0x3c1950f0ee56e18ac1db4e34194ea5943e2f975e29e221f963b29dbcdc225ea5'>
>>> markets.halborn_market(0, 0, {'from':deployer}).return_value
Transaction sent: 0x8f387a38d4e63af13bf103a12c9983fc32acfe502acd3bf168441ac56b18aa07
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 20
  Markets.halborn_market confirmed - Block: 21   Gas used: 25892 (0.22%)

'0x3194cBDC3dbcd3E11a07892e7bA5c3394048Cc87'
>>>
```

Code Location:

**Listing 34: BaseLogic.sol (Lines 90,94)**

```
76 function doExitMarket(address account, address underlying)
       internal {
77     uint32 numMarketsEntered = accountLookup[account].
           numMarketsEntered;
78     address[MAX_POSSIBLE_ENTERED_MARKETS] storage markets =
           marketsEntered[account];
79     uint searchIndex = type(uint).max;
80
81     for (uint i = 0; i < numMarketsEntered; i++) {
82         if (_getEnteredMarketIndex(account, markets, i) ==
               underlying) {
83             searchIndex = i;
84             break;
85         }
86     }
87
88     if (searchIndex == type(uint).max) return; // already exited
89
90     uint lastMarketIndex = numMarketsEntered - 1;
91     if (searchIndex != lastMarketIndex) _setEnteredMarketIndex(
```

```
            account, markets, searchIndex, _getEnteredMarketIndex(
            account, markets, lastMarketIndex));
92      accountLookup[account].numMarketsEntered--;
93
94      if (lastMarketIndex != 0) _setEnteredMarketIndex(account,
            markets, lastMarketIndex, address(0)); // zero out for
            storage refund
95  }
```

Risk Level:

**Likelihood - 1**
**Impact - 1**

Recommendation:

It is recommended to be consistent in the code and make sure that `0` indices are correctly checked and always kept in mind. Not being consistent on the usage of the indices could lead to out-of-bounds access or unexpected behaviours.

Remediation Plan:

RISK ACCEPTED: This is intentional. That's why there is the explicit if condition for this on line 94 of Listing 34. The reason is that the first market entered is packed into a slot with other fields. If any of them are non-zero then there will be no refund issued. Secondly, by leaving it non-zero, it will be cheaper to enter a market in the future. Since exit market operations can be done in a transaction that uses a very small amount of gas, it is possible that the storage refund will not be fully claimable (refunds are only claimable up to half a transaction's gas usage), so in these cases exiting and entering a market actually nets out to be cheaper if left set. Avoiding un-claimable refunds is the same reason that the re-entrancy guard uses the values 1 and 2, rather than 0 and 1.

## 3.4 (HAL-04) INVALID CONSTANT VALUE - INFORMATIONAL

Description:

During a manual static review, the tester noticed that the constant value defined in all the interest-rate-models contracts are returning a different value than the one described on the comments. The comment states that the value is the result of the 10% APR = 1e27 * 0.1 / (86400*365) operation resulting in 3170979198376458650 on the code. However, the result of the previous stated operation is 3170979198376458752 a difference of 102 units.

Code Location:

**Listing 35: IRMDefault.sol**

```
9 uint internal constant MAX_IR = 3170979198376458650; // 10% APR =
      1e27 * 0.1 / (86400*365)
```

**Listing 36: IRMLinear.sol**

```
13 uint internal constant MAX_IR = 3170979198376458650; // 10% APR =
      1e27 * 0.1 / (86400*365)
```

**Listing 37: IRMFixed.sol**

```
14 return 3170979198376458650; // 10% APR = 1e27 * 0.1 / (86400*365)
```

**Listing 38: IRMLinearRecursive.sol**

```
12 int internal constant MAX_IR = 3170979198376458650; // 10% APR = 1
      e27 * 0.1 / (86400*365)
```

Risk Level:

**Likelihood - 1**
**Impact - 1**

Recommendation:

It is recommended to either fix the comment with the correct operation or change the constant value to reflect the described operation.

Remediation Plan:

RISK ACCEPTED: The comment and constant values are correct. The python code that demonstrates this is experiencing floating point rounding error. Double precision floating point has a precision of approximately 16 decimal places, which is why you observe it as off by about 2 decimal places.

## 3.5 (HAL-05) INFINITE ALLOWANCE - INFORMATIONAL

Description:

Setting the allowance value to `-1` or `MAX UINT 256` on the `EToken` and `DToken` contract does allow the spender to keep performing transfers until a new approval is set on the spender.

Code Location:

**Listing 39: EToken.sol (Lines 186)**

```
186     if (!isSubAccountOf(msgSender, from) && assetStorage.
            eTokenAllowance[from][msgSender] != type(uint).max) {
187         require(assetStorage.eTokenAllowance[from][msgSender] >=
                amount, "e/insufficient-allowance");
```

```
188         unchecked { assetStorage.eTokenAllowance[from][msgSender]
                -= amount; }
189     }
```

**Listing 40: DToken.sol (Lines 165)**

```
165     if (!isSubAccountOf(msgSender, to) && assetStorage.
            dTokenAllowance[to][msgSender] != type(uint).max) {
166         require(assetStorage.dTokenAllowance[to][msgSender] >=
                amount, "e/insufficient-allowance");
167         unchecked { assetStorage.dTokenAllowance[to][msgSender] -=
                amount; }
168     }
```

Risk Level:

**Likelihood - 1**
**Impact - 1**

Recommendations:

Follow the standard ERC20 practices and allow the user to increase and decrease the approval amount (see increaseApproval and decreaseApproval of StandardToken.sol#L63-L98).

If this is not possible, ensure users are aware of this extra functionality and encourage them to use it when appropriate. Furthermore, it is preferable to periodically increase the allowance rather than disabling the allowance feature by using a max value.

Remediation Plan:

RISK ACCEPTED: This is intentional. For better or for worse, the increase/decreaseAllowance methods are almost never used in real-life, and users almost always authorise infinite allowance to contracts (probably because by convention it is cheaper to interact with tokens in this state since they don't bother decrementing the allowance).

# MANUAL TESTING

During the manual testing multiple questions where considered while evaluation each of the defined functions:

- Can it be re-called changing admin/roles and permissions?
- Can somehow an external controlled contract call again the function during the execution of it? (Re-entrancy)
- Can it be called twice in the same block and cause issues?
- Do we control sensitive or vulnerable parameters?
- Does the function check for boundaries on the parameters and internal values? Bigger than zero or equal? Argument count, array sizes, integer truncation . . .
- Are the function parameters and variables controlled by external contracts?
- Can extended contracts cause issues on the extender contract?
- Can we bypass Proxy restrictions and interact with the contract directly?

During the initial steps, coverage information was taken from the hardhat test cases. This information can be seen in 41. The uncovered lines were manually tested and kept in mind for logic errors and possible side-effects during the execution of the contracts.

**Listing 41**

```
------------------------------|----------|----------------|
File                          |  % Lines |Uncovered Lines |
------------------------------|----------|----------------|
 contracts/                   |    97.93 |                |
  Base.sol                    |    95.83 |             92 |
  BaseIRM.sol                 |      100 |                |
  BaseLogic.sol               |    98.43 |    150,433,434 |
  BaseModule.sol              |      100 |                |
  Constants.sol               |      100 |                |
  Euler.sol                   |    92.86 |             21 |
  Events.sol                  |      100 |                |
  Interfaces.sol              |      100 |                |
  Proxy.sol                   |      100 |                |
  Storage.sol                 |      100 |                |
  DToken.sol                  |    97.22 |        160,167 |
  EToken.sol                  |      100 |                |
```

```
  Exec.sol                      |    80.88 |... 84,90,92,94 |
  Governance.sol                |      100 |                |
  Installer.sol                 |    83.33 |          33,34 |
  Liquidation.sol               |    95.29 |123,125,150,151 |
  Markets.sol                   |    92.16 | 77,144,145,146 |
  RiskManager.sol               |    96.19 |  71,73,171,196 |
  IRMFixed.sol                  |      100 |                |
  IRMLinear.sol                 |      100 |                |
  IRMLinearRecursive.sol        |        0 |          15,16 |
  IRMZero.sol                   |      100 |                |
  InvariantChecker.sol          |        0 |... 84,87,90,93 |
  JunkETokenUpgrade.sol         |      100 |                |
  JunkMarketsUpgrade.sol        |      100 |                |
  LiquidationTest.sol           |    93.75 |             55 |
  MockUniswapV3Factory.sol      |      100 |                |
  MockUniswapV3Pool.sol         |    89.47 |          40,44 |
  SimpleUniswapPeriphery.sol    |    58.82 |... 60,68,81,84 |
  TestERC20.sol                 |       80 | 39,40,56,65,73 |
  RPow.sol                      |      100 |                |
  TickMath.sol                  |      100 |                |
  EulerGeneralView.sol          |    90.48 |   95,96,98,100 |
------------------------------|----------|----------------|
All files                       |    88.93 |                |
------------------------------|----------|----------------|
```

Some logical ideas were manually tested which include but not limited to:

- `unpackTrailingParams` does some validations, but can it be crafted to cause errors?
- Can module and proxy dependency be bypassed.
- Can I activate an `Etoken` and `Dtoken` token on the market causing underlying miscalculations.
- Decimal wrapping issues on `Etokens` if the underlying has different decimal places.
- Bad balance interpretation of token on `callBalanceOf`.
- What if the token contract is destroyed?
- Use of `unchecked` bypassing `SafeMath` integration could lead to issues.

Architecture overview:

Except for a small amount of dispatching logic (see Euler.sol), the contracts are organised into modules, which live in contracts/modules/.

- Each token must have its own address to conform to ERC-20, even though all storage lives inside the Euler contract

The modules are divided in 3 categories:

- **Single-proxy modules**: These are modules that are only accessible by a single address. For example, market activation is done by invoking a function on the single proxy for the Markets module. The IDs range is from 1 to 499,999.
- **Multi-proxy modules**: These are modules that have many addresses. For example, each EToken gets an address, but any calls to them are dispatched to the single EToken module instance. The IDs range is from 500,000 to 999,999.
- **Internal modules**: These are modules that are called internally by the Euler system and don't have any public proxies. These are only useful for their upgrade functionality, and the ability to stub in non-production code during testing/development. Examples are the RiskManager and interest rate model (IRM) modules. The IDs range is from 1,000,000 onwards.

Since modules are invoked by delegatecall, they should not have any storage-related initialisation in their constructors (Otherwise collision storage issues could happen). The only thing that should be done in their constructors is to initialise immutable variables, since these are embedded into the contract's bytecode, not storage. Modules also should not define any storage variables. In the rare cases they need private storage (ie interest rate model state), they should use unstructured storage.

Proxy Calling contract:

If the caller of the proxy is the deployer, the proxy copies the calldata on address 0x1f causing the first byte to be misaligned using calldatacopy (31, 0, calldatasize()). This first byte is then read and used on a switch-case statement for logging purposes. If the caller is not the deployer, the dispatch function is then called with the original calldata used.

# 4.1 General issues while manually testing

- trustedSenders for the installer module has the moduleImpl set to 0 on the Euler contract.
  - This happens because the proxy is created on the Euler contract creation with _createProxy which sets the struct moduleImpl to 0. The only time that the moduleImpl variable is set is during the call to installModules on Installer.sol. This means that all installed modules will have the moduleImpl variable set but not the Installer module as seen in Figure 1 and Figure 2. An expected code should be similar to the Listing 42.

**Listing 42: Expected code for the installer module added trustedSenders on the Euler constructor**

```
1     address installerProxy = _createProxy(MODULEID__INSTALLER);
2     trustedSenders[installerProxy].moduleImpl = installerModule;
```

```
function halborn_test1(address sender) external view returns (TrustedSenderInfo memory info){
    return trustedSenders[sender];
}
```

Figure 1: Code used to access the trustedSenders mapping

- All Proxy creator should be set to Euler address.

```
>>> Euler[0].moduleIdToProxy(1)
'0x3897810a334833184Ef7D6B419ba4d78EC2bBF80'
>>> euler.halborn_test1('0x3897810a334833184Ef7D6B419ba4d78EC2bBF80')
(1, "0x0000000000000000000000000000000000000000")
>>>
```

Figure 2: Screenshot showing that the Installer trustedSenders implementation is 0

It was observed that when creating the proxies for the activateMarket the trustedSenders mapping moduleImpl for the EToken and DToken module was not set. It was deduced that it should be at 0 since multiple implementations for them exists, aka multi-proxy.

```
>>> hal_dtoken
<DToken Contract '0x45CADE9Be96679283DB3835be8f351886eCBdC0A'>
>>> euler.halborn_test1(hal_dtoken)
(500001, "0x0000000000000000000000000000000000000000")
>>>
```

Figure 3: Screenshot showing that the implmentation of the Dtoken is set to 0

Successfully entering the market:

```
>>> markets.enterMarket(0, hal_token, {'from':deployer})
Transaction sent: 0x0d86c8162a9d152b8b81096dca4519e7b2c575d41b2e00c910eee6d8ad6d1080
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 17
  Markets.enterMarket confirmed - Block: 18   Gas used: 53237 (0.44%)

<Transaction '0x0d86c8162a9d152b8b81096dca4519e7b2c575d41b2e00c910eee6d8ad6d1080'>
>>> markets.getEnteredMarkets(deployer)
("0x3194cBDC3dbcd3E11a07892e7bA5c3394048Cc87")
>>>
```

Figure 4: Successful transaction showing a enter market call

BaseLogic:

If numMarketsEntered is 1, the lastMarketIndex will be 0 causing the _setEnteredMarketIndex to not happen. This issue is shown as informative on the report.

- ETokens and DTokens cannot be added as an “active market” since they

are checked using trustedSenders.

- IRMFixed.sol and IRMLinear 3170979198376458650; // 10% APR = 1e27 * 0.1 / (86400*365) should be, this is reflected in an informative issue on the report.

**Listing 43**

```
>>> int(1e27 * 0.1 / (86400*365))
3170979198376458752
```

E-token deposit:

Directly calling the implementation without passing though the proxy is causing reentrancy:

```
>>> t = EToken[0].deposit(0, 100, {'from': deployer})
Transaction sent: 0xa901cd918e7d7b2e801352f249c991531403a2bd8a4199939a685c45a868d8ee
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 38
  EToken.deposit confirmed (e/reentrancy) - Block: 45   Gas used: 22602 (0.19%)

>>> t = EToken[1].deposit(0, 100, {'from': deployer})
Transaction sent: 0xe3fac8d533bbb4f3c151cc2e772ebd26c6c0454b1fc0bbf34e034ac0f84a2d93
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 39
  EToken.deposit confirmed ŷ (ERC20: transfer amount exceeds allowance) - Block: 46   Gas used: 73
589 (0.61%)

>>>
```

Figure 5: manual/Untitled%205.png

Multiplication on division:

During automated testing multiple multiplication on division issues were found. Those issues were manually analyzed and were identified as not problematic since the truncation is done on purpose to round up internal representations. The function was compiled and bytecode analyzed in order to illustrate the opcode order as shown in Figure 6.

Unchecked:

The solidity version used 0.8.0 has SafeMath integration which performs arithmetic checks on each of the typed operations. A new keyword, named

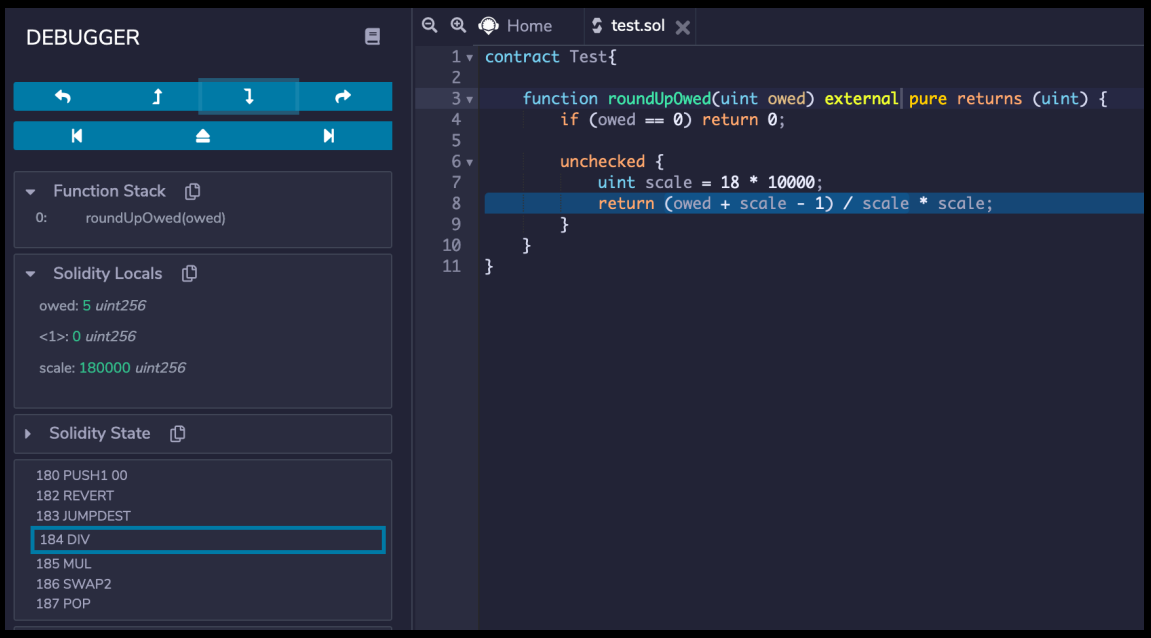



Figure 6: Showing that the multiplication opcode is done after the division

unchecked was added in order to remove those security checks causing gas costs to decrease. All the unchecked usages are displayed in Listing 44.

**Listing 44: Unchecked keyword usage**

```
BaseLogic.sol
143:         unchecked { assetCache.underlyingDecimalsScaler =
    10**(18 - underlyingDecimals); }
148:             unchecked { assetCache.poolSize = poolSize *
    assetCache.underlyingDecimalsScaler; }
175:         unchecked { scaledAmount = externalAmount * assetCache
    .underlyingDecimalsScaler; }
248:         unchecked { newFromBalance = origFromBalance - amount;
     }
287:     // unchecked is OK here since owed is always loaded from
    storage, so we know it fits into a uint144 (pre-interest
    accural)
293:         unchecked {
400:         unchecked { owedRemaining = owedRoundedUp - amount; }
429:         unchecked { newFromBorrow = origFromBorrow - amount; }
469:         unchecked { amountTransferred = poolSizeAfter -
    poolSizeBefore; }
```

```
11 479:         unchecked { amountTransferred = poolSizeBefore -
       poolSizeAfter; }
12
13 vendor/TickMath.sol
14 7://   * Wrapped function bodies with "unchecked {}" so as to not
       add any extra gas costs
15 31:       unchecked {
16 71:       unchecked {
17
18 halborn/ERC20Example.sol
19 124: * Using this library instead of the unchecked operations
       eliminates an entire
20
21 modules/EToken.sol
22 91:         unchecked {
23 178:             unchecked { assetStorage.eTokenAllowance[from][
       msgSender] -= amount; }
24
25 modules/DToken.sol
26 167:             unchecked { assetStorage.dTokenAllowance[to][
       msgSender] -= amount; }
27
28 modules/RiskManager.sol
29 105:         unchecked {
```

The code does use int96 very nicely, no issue with max values being overflowed or underflowed by truncating uint to int or the other way around.

Could potentially overflow if owed is to large:

**Listing 45: Possible overflow (Lines 295)**

```
290 function roundUpOwed(AssetCache memory assetCache, uint owed)
       internal pure returns (uint) {
291     if (owed == 0) return 0;
292
293     unchecked {
294         // max scale -> 0x33b2e3c9fd0803ce8000000 --->>> 1e9 * (10
              ** 18)
295         uint scale = INTERNAL_DEBT_PRECISION * assetCache.
              underlyingDecimalsScaler;
296         return (owed + scale - 1) / scale * scale;
```

```
297     }
298 }
```

As an example if owed is the following value:

**Listing 46**

```
1 MAX_UINT256 - scale + 1
2
3 max scale -> 0x33b2e3c9fd0803ce8000000 --->>> 1e9 * (10 ** 18)
4
5 (2 ** 256) - 0x33b2e3c9fd0803ce8000000 + 1
```

The roundUpOwed will be 0. Causing a division by zero error. To summarize if owed is larger than MAX_UINT256 - scale + 1 the operation will overflow and wrap around.

As an example, by using the code shown in Listing 47 and by exceeding the cache maxExternalAmount value of the underlying asset the code was able to detect the overflow as shown in Figure 7.

**Listing 47: Function used to load the asset cache values**

```
1     function halborn_cache() external returns (AssetCache memory
          assetCache) {
2         (address underlying, AssetStorage storage assetStorage,
              address proxyAddr, address msgSender) = CALLER();
3
4         AssetCache memory assetCache = loadAssetCache(underlying,
              assetStorage);
5
6         return assetCache;
7     }
```

```
>>> hal_etoken.deposit(0, 0xffffffffffffffffffffffffff5, {'from':deployer})
Transaction sent: 0x7997fb203344634ca42648166aee7e32d4e6e1f2cc4d256b870aada5cf4c0310
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 20
  EToken.deposit confirmed - Block: 21   Gas used: 58807 (0.49%)

<Transaction '0x7997fb203344634ca42648166aee7e32d4e6e1f2cc4d256b870aada5cf4c0310'>
>>> hal_etoken.halborn_cache({'from':deployer}).return_value
Transaction sent: 0x48c7eb56ef87bfb3ec31662adf0f092ef8ff939d0bdddcd29096f7c64032a79f
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 21
  EToken.halborn_cache confirmed - Block: 22   Gas used: 34070 (0.28%)

("0x3194cBDC3dbcd3E11a07892e7bA5c3394048Cc87", 0, 0, 0, 0, 18, 2000000, 0, 0, 1, 3000, 519229685853482762853049632922009 5, 1, 519229685853
482762853049632922009 5, 0)
>>> hal_etoken.deposit(0, 0x1, {'from':deployer})
Transaction sent: 0xb35644f423198f6a25c6dc4c791d09c9cc51b777d27bac5af1d76971b018a654
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 22
  EToken.deposit confirmed (e/amount-too-large) - Block: 23   Gas used: 60073 (0.50%)

<Transaction '0xb35644f423198f6a25c6dc4c791d09c9cc51b777d27bac5af1d76971b018a654'>
>>>
```

Figure 7: Example transactions that lead to the detection of an overflow on the maxExternalAmount cache

# AUTOMATED TESTING

# 5.1 STATIC ANALYSIS REPORT

Description:

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

BaseIRM.sol

```
INFO:Detectors:
Storage.moduleLookup (contracts/Storage.sol#15) is never initialized. It is used in:
        - Base.callInternalModule(uint256,bytes) (contracts/Base.sol#34-38)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables
INFO:Detectors:
Base.revertBytes(bytes) (contracts/Base.sol#85-93) uses assembly
        - INLINE ASM (contracts/Base.sol#87-89)
BaseModule.unpackTrailingParamMsgSender() (contracts/BaseModule.sol#21-28) uses assembly
        - INLINE ASM (contracts/BaseModule.sol#22-27)
BaseModule.unpackTrailingParams() (contracts/BaseModule.sol#30-40) uses assembly
        - INLINE ASM (contracts/BaseModule.sol#31-39)
Proxy.fallback() (contracts/Proxy.sol#17-49) uses assembly
        - INLINE ASM (contracts/Proxy.sol#21-34)
        - INLINE ASM (contracts/Proxy.sol#36-47)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Base._createProxy(uint256) (contracts/Base.sol#13-32) is never used and should be removed
Base.callInternalModule(uint256,bytes) (contracts/Base.sol#34-38) is never used and should be removed
Base.revertBytes(bytes) (contracts/Base.sol#85-93) is never used and should be removed
BaseModule.emitViaProxy_Approval(address,address,address,uint256) (contracts/BaseModule.sol#56-65) is never used and should be removed
BaseModule.emitViaProxy_Transfer(address,address,address,uint256) (contracts/BaseModule.sol#45-54) is never used and should be removed
BaseModule.unpackTrailingParamMsgSender() (contracts/BaseModule.sol#21-28) is never used and should be removed
BaseModule.unpackTrailingParams() (contracts/BaseModule.sol#30-40) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (contracts/Base.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/BaseIRM.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/BaseModule.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Constants.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Events.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Interfaces.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Proxy.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Storage.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Base.callInternalModule(uint256,bytes) (contracts/Base.sol#34-38):
        - (success,result) = moduleLookup[moduleId].delegatecall(input) (contracts/Base.sol#35)
Low level call in BaseModule.emitViaProxy_Transfer(address,address,address,uint256) (contracts/BaseModule.sol#45-54):
        - (success) = proxyAddr.call(abi.encodePacked(uint8(3),keccak256(bytes)(bytes(Transfer(address,address,uint256))),bytes32(uint256(uint160(from))),bytes32(uint256(uint160(to))),value)
) (contracts/BaseModule.sol#46-52)
Low level call in BaseModule.emitViaProxy_Approval(address,address,address,uint256) (contracts/BaseModule.sol#56-65):
        - (success) = proxyAddr.call(abi.encodePacked(uint8(3),keccak256(bytes)(bytes(Approval(address,address,uint256))),bytes32(uint256(uint160(owner))),bytes32(uint256(uint160(spender))),
value)) (contracts/BaseModule.sol#57-63)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
BaseIRM (contracts/BaseIRM.sol#7-13) should inherit from IIRM (contracts/Interfaces.sol#69-72)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-inheritance
INFO:Detectors:
Modifier Base.FREEMEM() (contracts/Base.sol#62-79) is not in mixedCase
Function BaseModule.emitViaProxy_Transfer(address,address,address,uint256) (contracts/BaseModule.sol#45-54) is not in mixedCase
Function BaseModule.emitViaProxy_Approval(address,address,address,uint256) (contracts/BaseModule.sol#56-65) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
```

## BaseLogic.sol

```
INFO:Detectors:
Storage.moduleLookup (contracts/Storage.sol#15) is never initialized. It is used in:
        - Base.callInternalModule(uint256,bytes) (contracts/Base.sol#34-38)
Storage.marketsEntered (contracts/Storage.sol#39) is never initialized. It is used in:
        - BaseLogic.getEnteredMarketsArray(address) (contracts/BaseLogic.sol#43-58)
        - BaseLogic.doEnterMarket(address,address) (contracts/BaseLogic.sol#60-72)
        - BaseLogic.doExitMarket(address,address) (contracts/BaseLogic.sol#76-95)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables
INFO:Detectors:
BaseLogic.roundUpOwed(BaseLogic.AssetCache,uint256) (contracts/BaseLogic.sol#290-297) performs a multiplication on the result of a division:
        -(owed + scale - 1) / scale * scale (contracts/BaseLogic.sol#295)
BaseLogic.updateInterestRate(BaseLogic.AssetCache) (contracts/BaseLogic.sol#336-354) performs a multiplication on the result of a division:
        -totalBorrows = assetCache.totalBorrows / INTERNAL_DEBT_PRECISION (contracts/BaseLogic.sol#340)
        -newUtilisation = uint32(totalBorrows * (uint256(type()(uint32).max) * 1e18) / total / 1e18) (contracts/BaseLogic.sol#343)
RPow.rpow(uint256,uint256,uint256) (contracts/vendor/RPow.sol#23-45) performs a multiplication on the result of a division:
        -x = xxRound_rpow_asm_0 / base (contracts/vendor/RPow.sol#34)
        -zx_rpow_asm_0 = z * x (contracts/vendor/RPow.sol#36)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
BaseLogic.computeUpdatedInterestAccumulator(BaseLogic.AssetCache) (contracts/BaseLogic.sol#204-210) uses a dangerous strict equality:
        - deltaT == 0 (contracts/BaseLogic.sol#208)
BaseLogic.roundUpOwed(BaseLogic.AssetCache,uint256) (contracts/BaseLogic.sol#290-297) uses a dangerous strict equality:
        - owed == 0 (contracts/BaseLogic.sol#291)
BaseLogic.updateInterestAccumulator(Storage.AssetStorage,BaseLogic.AssetCache) (contracts/BaseLogic.sol#316-331) uses a dangerous strict equality:
        - block.timestamp == assetCache.lastInterestAccumulatorUpdate (contracts/BaseLogic.sol#317)
BaseLogic.updateInterestRate(BaseLogic.AssetCache) (contracts/BaseLogic.sol#336-354) uses a dangerous strict equality:
        - total == 0 (contracts/BaseLogic.sol#342)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in BaseLogic.increaseBorrow(Storage.AssetStorage,BaseLogic.AssetCache,address,address,uint256) (contracts/BaseLogic.sol#369-388):
        External calls:
        - updateInterestRate(assetCache) (contracts/BaseLogic.sol#383)
                - (success,result) = moduleLookup[moduleId].delegatecall(input) (contracts/Base.sol#35)
        - emitViaProxy_Transfer(dTokenAddress,address(0),account,origAmount) (contracts/BaseLogic.sol#386)
                - (success) = proxyAddr.call(abi.encodePacked(uint8(3),keccak256(bytes)(bytes(Transfer(address,address,uint256))),bytes32(uint256(uint160(from))),bytes32(uint256(uint160(to))
),value)) (contracts/BaseModule.sol#46-52)
        State variables written after the call(s):
        - updateLastActivity(account) (contracts/BaseLogic.sol#387)
                - accountLookup[account].lastActivity = uint40(block.timestamp) (contracts/BaseLogic.sol#27)
Reentrancy in BaseLogic.transferBorrow(Storage.AssetStorage,BaseLogic.AssetCache,address,address,address,uint256) (contracts/BaseLogic.sol#416-443):
        External calls:
        - emitViaProxy_Transfer(dTokenAddress,from,to,origAmount) (contracts/BaseLogic.sol#440)
                - (success) = proxyAddr.call(abi.encodePacked(uint8(3),keccak256(bytes)(bytes(Transfer(address,address,uint256))),bytes32(uint256(uint160(from))),bytes32(uint256(uint160(to))
),value)) (contracts/BaseModule.sol#46-52)
        State variables written after the call(s):
        - updateLastActivity(from) (contracts/BaseLogic.sol#441)
                - accountLookup[account].lastActivity = uint40(block.timestamp) (contracts/BaseLogic.sol#27)
        - updateLastActivity(to) (contracts/BaseLogic.sol#442)
                - accountLookup[account].lastActivity = uint40(block.timestamp) (contracts/BaseLogic.sol#27)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
Reentrancy in BaseLogic.decreaseBalance(Storage.AssetStorage,BaseLogic.AssetCache,address,address,uint256) (contracts/BaseLogic.sol#229-242):
        External calls:
        - updateInterestRate(assetCache) (contracts/BaseLogic.sol#237)
```

## BaseModule.sol

```
INFO:Detectors:
Storage.moduleLookup (contracts/Storage.sol#15) is never initialized. It is used in:
        - Base.callInternalModule(uint256,bytes) (contracts/Base.sol#34-38)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables
INFO:Detectors:
Base.revertBytes(bytes) (contracts/Base.sol#85-93) uses assembly
        - INLINE ASM (contracts/Base.sol#87-89)
BaseModule.unpackTrailingParamMsgSender() (contracts/BaseModule.sol#21-28) uses assembly
        - INLINE ASM (contracts/BaseModule.sol#22-27)
BaseModule.unpackTrailingParams() (contracts/BaseModule.sol#30-40) uses assembly
        - INLINE ASM (contracts/BaseModule.sol#31-39)
Proxy.fallback() (contracts/Proxy.sol#17-49) uses assembly
        - INLINE ASM (contracts/Proxy.sol#21-34)
        - INLINE ASM (contracts/Proxy.sol#36-47)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Base._createProxy(uint256) (contracts/Base.sol#13-32) is never used and should be removed
Base.callInternalModule(uint256,bytes) (contracts/Base.sol#34-38) is never used and should be removed
Base.revertBytes(bytes) (contracts/Base.sol#85-93) is never used and should be removed
BaseModule.emitViaProxy_Approval(address,address,address,uint256) (contracts/BaseModule.sol#56-65) is never used and should be removed
BaseModule.emitViaProxy_Transfer(address,address,address,uint256) (contracts/BaseModule.sol#45-54) is never used and should be removed
BaseModule.unpackTrailingParamMsgSender() (contracts/BaseModule.sol#21-28) is never used and should be removed
BaseModule.unpackTrailingParams() (contracts/BaseModule.sol#30-40) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (contracts/Base.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/BaseModule.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Constants.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Events.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Interfaces.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Proxy.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Storage.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Base.callInternalModule(uint256,bytes) (contracts/Base.sol#34-38):
        - (success,result) = moduleLookup[moduleId].delegatecall(input) (contracts/Base.sol#35)
Low level call in BaseModule.emitViaProxy_Transfer(address,address,address,uint256) (contracts/BaseModule.sol#45-54):
        - (success) = proxyAddr.call(abi.encodePacked(uint8(3),keccak256(bytes)(bytes(Transfer(address,address,uint256))),bytes32(uint256(uint160(from))),bytes32(uint256(uint160(to))),value)
) (contracts/BaseModule.sol#46-52)
Low level call in BaseModule.emitViaProxy_Approval(address,address,address,uint256) (contracts/BaseModule.sol#56-65):
        - (success) = proxyAddr.call(abi.encodePacked(uint8(3),keccak256(bytes)(bytes(Approval(address,address,uint256))),bytes32(uint256(uint160(owner))),bytes32(uint256(uint160(spender))),
value)) (contracts/BaseModule.sol#57-63)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Modifier Base.FREEMEM() (contracts/Base.sol#62-79) is not in mixedCase
Function BaseModule.emitViaProxy_Transfer(address,address,address,uint256) (contracts/BaseModule.sol#45-54) is not in mixedCase
Function BaseModule.emitViaProxy_Approval(address,address,address,uint256) (contracts/BaseModule.sol#56-65) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable Constants.MODULEID__DTOKEN (contracts/Constants.sol#42) is too similar to Constants.MODULEID__ETOKEN (contracts/Constants.sol#41)
Variable Storage.dTokenLookup (contracts/Storage.sol#87) is too similar to Storage.eTokenLookup (contracts/Storage.sol#86)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
```

## Base.sol

```
INFO:Detectors:
Storage.moduleLookup (contracts/Storage.sol#15) is never initialized. It is used in:
        - Base.callInternalModule(uint256,bytes) (contracts/Base.sol#34-38)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables
INFO:Detectors:
Base.revertBytes(bytes) (contracts/Base.sol#85-93) uses assembly
        - INLINE ASM (contracts/Base.sol#87-89)
Proxy.fallback() (contracts/Proxy.sol#17-49) uses assembly
        - INLINE ASM (contracts/Proxy.sol#21-34)
        - INLINE ASM (contracts/Proxy.sol#36-47)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Base._createProxy(uint256) (contracts/Base.sol#13-32) is never used and should be removed
Base.callInternalModule(uint256,bytes) (contracts/Base.sol#34-38) is never used and should be removed
Base.revertBytes(bytes) (contracts/Base.sol#85-93) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (contracts/Base.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Constants.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Events.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Interfaces.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Proxy.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Storage.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Base.callInternalModule(uint256,bytes) (contracts/Base.sol#34-38):
        - (success,result) = moduleLookup[moduleId].delegatecall(input) (contracts/Base.sol#35)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Modifier Base.FREEMEM() (contracts/Base.sol#62-79) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable Constants.MODULEID__DTOKEN (contracts/Constants.sol#42) is too similar to Constants.MODULEID__ETOKEN (contracts/Constants.sol#41)
Variable Storage.dTokenLookup (contracts/Storage.sol#87) is too similar to Storage.eTokenLookup (contracts/Storage.sol#86)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
Proxy.fallback() (contracts/Proxy.sol#17-49) uses literals with too many digits:
        - mstore(uint256,uint256)(0,0xe9c4a3ac0000000000000000000000000000000000000000000000000000000000000000) (contracts/Proxy.sol#37)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
Constants.MAX_SANE_AMOUNT (contracts/Constants.sol#8) is never used in Base (contracts/Base.sol#10-94)
Constants.MAX_SANE_DEBT_AMOUNT (contracts/Constants.sol#9) is never used in Base (contracts/Base.sol#10-94)
Constants.INTERNAL_DEBT_PRECISION (contracts/Constants.sol#10) is never used in Base (contracts/Base.sol#10-94)
Constants.MAX_ENTERED_MARKETS (contracts/Constants.sol#11) is never used in Base (contracts/Base.sol#10-94)
Constants.MAX_POSSIBLE_ENTERED_MARKETS (contracts/Constants.sol#12) is never used in Base (contracts/Base.sol#10-94)
Constants.CONFIG_FACTOR_SCALE (contracts/Constants.sol#13) is never used in Base (contracts/Base.sol#10-94)
Constants.INITIAL_INTEREST_ACCUMULATOR (contracts/Constants.sol#14) is never used in Base (contracts/Base.sol#10-94)
Constants.PRICINGTYPE__PEGGED (contracts/Constants.sol#25) is never used in Base (contracts/Base.sol#10-94)
Constants.PRICINGTYPE__UNISWAP3_TWAP (contracts/Constants.sol#26) is never used in Base (contracts/Base.sol#10-94)
Constants.MODULEID__INSTALLER (contracts/Constants.sol#32) is never used in Base (contracts/Base.sol#10-94)
Constants.MODULEID__MARKETS (contracts/Constants.sol#33) is never used in Base (contracts/Base.sol#10-94)
```

## Euler.sol

```
INFO:Detectors:
Base.callInternalModule(uint256,bytes) (contracts/Base.sol#34-38) uses delegatecall to a input-controlled function id
        - (success,result) = moduleLookup[moduleId].delegatecall(input) (contracts/Base.sol#35)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#controlled-delegatecall
INFO:Detectors:
Euler.constructor(address,address).admin (contracts/Euler.sol#9) lacks a zero-check on :
              - upgradeAdmin = admin (contracts/Euler.sol#11)
              - governorAdmin = admin (contracts/Euler.sol#12)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Base.revertBytes(bytes) (contracts/Base.sol#85-93) uses assembly
        - INLINE ASM (contracts/Base.sol#87-89)
Euler.dispatch() (contracts/Euler.sol#28-52) uses assembly
        - INLINE ASM (contracts/Euler.sol#39-51)
Proxy.fallback() (contracts/Proxy.sol#17-49) uses assembly
        - INLINE ASM (contracts/Proxy.sol#21-34)
        - INLINE ASM (contracts/Proxy.sol#36-47)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Base.callInternalModule(uint256,bytes) (contracts/Base.sol#34-38) is never used and should be removed
Base.revertBytes(bytes) (contracts/Base.sol#85-93) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (contracts/Base.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Constants.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Euler.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Events.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Interfaces.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Proxy.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Storage.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Base.callInternalModule(uint256,bytes) (contracts/Base.sol#34-38):
        - (success,result) = moduleLookup[moduleId].delegatecall(input) (contracts/Base.sol#35)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Euler (contracts/Euler.sol#8-53) should inherit from IEuler (contracts/Interfaces.sol#45-48)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-inheritance
INFO:Detectors:
Modifier Base.FREEMEM() (contracts/Base.sol#62-79) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable Constants.MODULEID__DTOKEN (contracts/Constants.sol#42) is too similar to Constants.MODULEID__ETOKEN (contracts/Constants.sol#41)
Variable Storage.dTokenLookup (contracts/Storage.sol#87) is too similar to Storage.eTokenLookup (contracts/Storage.sol#86)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
Proxy.fallback() (contracts/Proxy.sol#17-49) uses literals with too many digits:
        - mstore(uint256,uint256)(0,0xe9c4a3ac0000000000000000000000000000000000000000000000000000000000000000) (contracts/Proxy.sol#37)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
Constants.MAX_SANE_AMOUNT (contracts/Constants.sol#8) is never used in Euler (contracts/Euler.sol#8-53)
```

## modules/DToken.sol

```
INFO:Detectors:
Storage.moduleLookup (contracts/Storage.sol#15) is never initialized. It is used in:
        - Base.callInternalModule(uint256,bytes) (contracts/Base.sol#34-38)
Storage.marketsEntered (contracts/Storage.sol#39) is never initialized. It is used in:
        - BaseLogic.getEnteredMarketsArray(address) (contracts/BaseLogic.sol#43-58)
        - BaseLogic.doEnterMarket(address,address) (contracts/BaseLogic.sol#60-72)
        - BaseLogic.doExitMarket(address,address) (contracts/BaseLogic.sol#76-95)
Storage.eTokenLookup (contracts/Storage.sol#86) is never initialized. It is used in:
        - DToken.CALLER() (contracts/modules/DToken.sol#11-17)
Storage.dTokenLookup (contracts/Storage.sol#87) is never initialized. It is used in:
        - DToken.CALLER() (contracts/modules/DToken.sol#11-17)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables
INFO:Detectors:
BaseLogic.roundUpOwed(BaseLogic.AssetCache,uint256) (contracts/BaseLogic.sol#290-297) performs a multiplication on the result of a division:
        -(owed + scale - 1) / scale * scale (contracts/BaseLogic.sol#295)
BaseLogic.updateInterestRate(BaseLogic.AssetCache) (contracts/BaseLogic.sol#336-354) performs a multiplication on the result of a division:
        -totalBorrows = assetCache.totalBorrows / INTERNAL_DEBT_PRECISION (contracts/BaseLogic.sol#340)
        -newUtilisation = uint32(totalBorrows * (uint256(type()(uint32).max) * 1e18) / total / 1e18) (contracts/BaseLogic.sol#343)
RPow.rpow(uint256,uint256,uint256) (contracts/vendor/RPow.sol#23-45) performs a multiplication on the result of a division:
        -x = xxRound_rpow_asm_0 / base (contracts/vendor/RPow.sol#34)
        -zx_rpow_asm_0 = z * x (contracts/vendor/RPow.sol#36)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
BaseLogic.computeUpdatedInterestAccumulator(BaseLogic.AssetCache) (contracts/BaseLogic.sol#204-210) uses a dangerous strict equality:
        - deltaT == 0 (contracts/BaseLogic.sol#208)
DToken.repay(uint256,uint256) (contracts/modules/DToken.sol#99-120) uses a dangerous strict equality:
        - owed == 0 (contracts/modules/DToken.sol#111)
BaseLogic.roundUpOwed(BaseLogic.AssetCache,uint256) (contracts/BaseLogic.sol#290-297) uses a dangerous strict equality:
        - owed == 0 (contracts/BaseLogic.sol#291)
BaseLogic.safeTransferFrom(address,address,address,uint256) (contracts/BaseLogic.sol#450-453) uses a dangerous strict equality:
        - require(bool,string)(success && (data.length == 0 || abi.decode(data,(bool))),string(data)) (contracts/BaseLogic.sol#452)
BaseLogic.updateInterestAccumulator(Storage.AssetStorage,BaseLogic.AssetCache) (contracts/BaseLogic.sol#316-331) uses a dangerous strict equality:
        - block.timestamp == assetCache.lastInterestAccumulatorUpdate (contracts/BaseLogic.sol#317)
BaseLogic.updateInterestRate(BaseLogic.AssetCache) (contracts/BaseLogic.sol#336-354) uses a dangerous strict equality:
        - total == 0 (contracts/BaseLogic.sol#342)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in BaseLogic.increaseBorrow(Storage.AssetStorage,BaseLogic.AssetCache,address,address,uint256) (contracts/BaseLogic.sol#369-388):
        External calls:
        - updateInterestRate(assetCache) (contracts/BaseLogic.sol#383)
                - (success,result) = moduleLookup[moduleId].delegatecall(input) (contracts/Base.sol#35)
        - emitViaProxy_Transfer(dTokenAddress,address(0),account,origAmount) (contracts/BaseLogic.sol#386)
                - (success) = proxyAddr.call(abi.encodePacked(uint8(3),keccak256(bytes)(bytes(Transfer(address,address,uint256))),bytes32(uint256(uint160(from))),bytes32(uint256(uint160(to))
),value)) (contracts/BaseModule.sol#46-52)
        State variables written after the call(s):
        - updateLastActivity(account) (contracts/BaseLogic.sol#387)
                - accountLookup[account].lastActivity = uint40(block.timestamp) (contracts/BaseLogic.sol#27)
Reentrancy in BaseLogic.transferBorrow(Storage.AssetStorage,BaseLogic.AssetCache,address,address,uint256) (contracts/BaseLogic.sol#416-443):
        External calls:
        - emitViaProxy_Transfer(dTokenAddress,from,to,origAmount) (contracts/BaseLogic.sol#440)
                - (success) = proxyAddr.call(abi.encodePacked(uint8(3),keccak256(bytes)(bytes(Transfer(address,address,uint256))),bytes32(uint256(uint160(from))),bytes32(uint256(uint160(to))
),value)) (contracts/BaseModule.sol#46-52)
        State variables written after the call(s):
        - updateLastActivity(from) (contracts/BaseLogic.sol#441)
```

## modules/EToken.sol

```
INFO:Detectors:
Storage.moduleLookup (contracts/Storage.sol#15) is never initialized. It is used in:
        - Base.callInternalModule(uint256,bytes) (contracts/Base.sol#34-38)
Storage.marketsEntered (contracts/Storage.sol#39) is never initialized. It is used in:
        - BaseLogic.getEnteredMarketsArray(address) (contracts/BaseLogic.sol#43-58)
        - BaseLogic.doEnterMarket(address,address) (contracts/BaseLogic.sol#60-72)
        - BaseLogic.doExitMarket(address,address) (contracts/BaseLogic.sol#76-95)
Storage.eTokenLookup (contracts/Storage.sol#86) is never initialized. It is used in:
        - EToken.CALLER() (contracts/modules/EToken.sol#11-16)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables
INFO:Detectors:
BaseLogic.roundUpOwed(BaseLogic.AssetCache,uint256) (contracts/BaseLogic.sol#290-297) performs a multiplication on the result of a division:
        -(owed + scale - 1) / scale * scale (contracts/BaseLogic.sol#295)
BaseLogic.updateInterestRate(BaseLogic.AssetCache) (contracts/BaseLogic.sol#336-354) performs a multiplication on the result of a division:
        -totalBorrows = assetCache.totalBorrows / INTERNAL_DEBT_PRECISION (contracts/BaseLogic.sol#340)
        -newUtilisation = uint32(totalBorrows * (uint256(type()(uint32).max) * 1e18) / total / 1e18) (contracts/BaseLogic.sol#343)
RPow.rpow(uint256,uint256,uint256) (contracts/vendor/RPow.sol#23-45) performs a multiplication on the result of a division:
        -x = xxRound_rpow_asm_0 / base (contracts/vendor/RPow.sol#34)
        -zx_rpow_asm_0 = z * x (contracts/vendor/RPow.sol#36)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
BaseLogic.computeUpdatedInterestAccumulator(BaseLogic.AssetCache) (contracts/BaseLogic.sol#204-210) uses a dangerous strict equality:
        - deltaT == 0 (contracts/BaseLogic.sol#208)
BaseLogic.roundUpOwed(BaseLogic.AssetCache,uint256) (contracts/BaseLogic.sol#290-297) uses a dangerous strict equality:
        - owed == 0 (contracts/BaseLogic.sol#291)
BaseLogic.updateInterestAccumulator(Storage.AssetStorage,BaseLogic.AssetCache) (contracts/BaseLogic.sol#316-331) uses a dangerous strict equality:
        - block.timestamp == assetCache.lastInterestAccumulatorUpdate (contracts/BaseLogic.sol#317)
BaseLogic.updateInterestRate(BaseLogic.AssetCache) (contracts/BaseLogic.sol#336-354) uses a dangerous strict equality:
        - total == 0 (contracts/BaseLogic.sol#342)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in BaseLogic.increaseBorrow(Storage.AssetStorage,BaseLogic.AssetCache,address,address,uint256) (contracts/BaseLogic.sol#369-388):
        External calls:
        - updateInterestRate(assetCache) (contracts/BaseLogic.sol#383)
                - (success,result) = moduleLookup[moduleId].delegatecall(input) (contracts/Base.sol#35)
        - emitViaProxy_Transfer(dTokenAddress,address(0),account,origAmount) (contracts/BaseLogic.sol#386)
                - (success) = proxyAddr.call(abi.encodePacked(uint8(3),keccak256(bytes)(bytes(Transfer(address,address,uint256))),bytes32(uint256(uint160(from))),bytes32(uint256(uint160(to))
),value)) (contracts/BaseModule.sol#46-52)
        State variables written after the call(s):
        - updateLastActivity(account) (contracts/BaseLogic.sol#387)
                - accountLookup[account].lastActivity = uint40(block.timestamp) (contracts/BaseLogic.sol#27)
Reentrancy in BaseLogic.transferBorrow(Storage.AssetStorage,BaseLogic.AssetCache,address,address,address,uint256) (contracts/BaseLogic.sol#416-443):
        External calls:
        - emitViaProxy_Transfer(dTokenAddress,from,to,origAmount) (contracts/BaseLogic.sol#440)
                - (success) = proxyAddr.call(abi.encodePacked(uint8(3),keccak256(bytes)(bytes(Transfer(address,address,uint256))),bytes32(uint256(uint160(from))),bytes32(uint256(uint160(to))
),value)) (contracts/BaseModule.sol#46-52)
        State variables written after the call(s):
        - updateLastActivity(from) (contracts/BaseLogic.sol#441)
                - accountLookup[account].lastActivity = uint40(block.timestamp) (contracts/BaseLogic.sol#27)
        - updateLastActivity(to) (contracts/BaseLogic.sol#442)
                - accountLookup[account].lastActivity = uint40(block.timestamp) (contracts/BaseLogic.sol#27)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
Reentrancy in BaseLogic.decreaseBalance(Storage.AssetStorage,BaseLogic.AssetCache,address,address,uint256) (contracts/BaseLogic.sol#229-242):
```

## modules/Exec.sol

```
ERROR:ContractSolcParsing:Missing function 'name'
INFO:Detectors:
BaseLogic.roundUpOwed(BaseLogic.AssetCache,uint256) (contracts/BaseLogic.sol#290-297) performs a multiplication on the result of a division:
        -(owed + scale - 1) / scale * scale (contracts/BaseLogic.sol#295)
BaseLogic.updateInterestRate(BaseLogic.AssetCache) (contracts/BaseLogic.sol#336-354) performs a multiplication on the result of a division:
        -totalBorrows = assetCache.totalBorrows / INTERNAL_DEBT_PRECISION (contracts/BaseLogic.sol#340)
        -newUtilisation = uint32(totalBorrows * (uint256(type()(uint32).max) * 1e18) / total / 1e18) (contracts/BaseLogic.sol#343)
RPow.rpow(uint256,uint256,uint256) (contracts/vendor/RPow.sol#23-45) performs a multiplication on the result of a division:
        -x = xxRound_rpow_asm_0 / base (contracts/vendor/RPow.sol#34)
        -zx_rpow_asm_0 = z * x (contracts/vendor/RPow.sol#36)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
Reentrancy in Exec.deferLiquidityCheck(address,bytes) (contracts/modules/Exec.sol#41-52):
        External calls:
        - IDeferredLiquidityCheck(msgSender).onDeferredLiquidityCheck(data) (contracts/modules/Exec.sol#47)
        State variables written after the call(s):
        - accountLookup[account].liquidityCheckInProgress = false (contracts/modules/Exec.sol#49)
Reentrancy in BaseLogic.increaseBorrow(Storage.AssetStorage,BaseLogic.AssetCache,address,address,uint256) (contracts/BaseLogic.sol#369-388):
        External calls:
        - updateInterestRate(assetCache) (contracts/BaseLogic.sol#383)
                - (success,result) = moduleLookup[moduleId].delegatecall(input) (contracts/Base.sol#35)
        - emitViaProxy_Transfer(dTokenAddress,address(0),account,origAmount) (contracts/BaseLogic.sol#386)
                - (success) = proxyAddr.call(abi.encodePacked(uint8(3),keccak256(bytes)(bytes(Transfer(address,address,uint256))),bytes32(uint256(uint160(from))),bytes32(uint256(uint160(to))
),value)) (contracts/BaseModule.sol#46-52)
        State variables written after the call(s):
        - updateLastActivity(account) (contracts/BaseLogic.sol#387)
                - accountLookup[account].lastActivity = uint40(block.timestamp) (contracts/BaseLogic.sol#27)
Reentrancy in BaseLogic.transferBorrow(Storage.AssetStorage,BaseLogic.AssetCache,address,address,address,uint256) (contracts/BaseLogic.sol#416-443):
        External calls:
        - emitViaProxy_Transfer(dTokenAddress,from,to,origAmount) (contracts/BaseLogic.sol#440)
                - (success) = proxyAddr.call(abi.encodePacked(uint8(3),keccak256(bytes)(bytes(Transfer(address,address,uint256))),bytes32(uint256(uint160(from))),bytes32(uint256(uint160(to))
),value)) (contracts/BaseModule.sol#46-52)
        State variables written after the call(s):
        - updateLastActivity(from) (contracts/BaseLogic.sol#441)
                - accountLookup[account].lastActivity = uint40(block.timestamp) (contracts/BaseLogic.sol#27)
        - updateLastActivity(to) (contracts/BaseLogic.sol#442)
                - accountLookup[account].lastActivity = uint40(block.timestamp) (contracts/BaseLogic.sol#27)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
Exec.batchDispatch(EulerBatchItem[],address[]).moduleId (contracts/modules/Exec.sol#68) shadows:
        - BaseModule.moduleId (contracts/BaseModule.sol#12) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Reentrancy in BaseLogic.decreaseBalance(Storage.AssetStorage,BaseLogic.AssetCache,address,address,uint256) (contracts/BaseLogic.sol#229-242):
        External calls:
        - updateInterestRate(assetCache) (contracts/BaseLogic.sol#237)
                - (success,result) = moduleLookup[moduleId].delegatecall(input) (contracts/Base.sol#35)
        - emitViaProxy_Transfer(eTokenAddress,account,address(0),amount) (contracts/BaseLogic.sol#240)
                - (success) = proxyAddr.call(abi.encodePacked(uint8(3),keccak256(bytes)(bytes(Transfer(address,address,uint256))),bytes32(uint256(uint160(from))),bytes32(uint256(uint160(to))
),value)) (contracts/BaseModule.sol#46-52)
        State variables written after the call(s):
        - updateLastActivity(account) (contracts/BaseLogic.sol#241)
                - accountLookup[account].lastActivity = uint40(block.timestamp) (contracts/BaseLogic.sol#27)
Reentrancy in BaseLogic.decreaseBorrow(Storage.AssetStorage,BaseLogic.AssetCache,address,address,uint256) (contracts/BaseLogic.sol#390-414):
        External calls:
```

## modules/Governance.sol

```
INFO:Detectors:
Storage.governorAdmin (contracts/Storage.sol#13) is never initialized. It is used in:
        - Governance.getGovernorAdmin() (contracts/modules/Governance.sol#49-51)
Storage.moduleLookup (contracts/Storage.sol#15) is never initialized. It is used in:
        - Base.callInternalModule(uint256,bytes) (contracts/Base.sol#34-38)
Storage.marketsEntered (contracts/Storage.sol#39) is never initialized. It is used in:
        - BaseLogic.getEnteredMarketsArray(address) (contracts/BaseLogic.sol#43-58)
        - BaseLogic.doEnterMarket(address,address) (contracts/BaseLogic.sol#60-72)
        - BaseLogic.doExitMarket(address,address) (contracts/BaseLogic.sol#76-95)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables
INFO:Detectors:
BaseLogic.roundUpOwed(BaseLogic.AssetCache,uint256) (contracts/BaseLogic.sol#290-297) performs a multiplication on the result of a division:
        -(owed + scale - 1) / scale * scale (contracts/BaseLogic.sol#295)
BaseLogic.updateInterestRate(BaseLogic.AssetCache) (contracts/BaseLogic.sol#336-354) performs a multiplication on the result of a division:
        -totalBorrows = assetCache.totalBorrows / INTERNAL_DEBT_PRECISION (contracts/BaseLogic.sol#340)
        -newUtilisation = uint32(totalBorrows * (uint256(type()(uint32).max) * 1e18) / total / 1e18) (contracts/BaseLogic.sol#343)
RPow.rpow(uint256,uint256,uint256) (contracts/vendor/RPow.sol#23-45) performs a multiplication on the result of a division:
        -x = xxRound_rpow_asm_0 / base (contracts/vendor/RPow.sol#34)
        -zx_rpow_asm_0 = z * x (contracts/vendor/RPow.sol#36)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
BaseLogic.computeUpdatedInterestAccumulator(BaseLogic.AssetCache) (contracts/BaseLogic.sol#204-210) uses a dangerous strict equality:
        - deltaT == 0 (contracts/BaseLogic.sol#208)
BaseLogic.roundUpOwed(BaseLogic.AssetCache,uint256) (contracts/BaseLogic.sol#290-297) uses a dangerous strict equality:
        - owed == 0 (contracts/BaseLogic.sol#291)
BaseLogic.updateInterestAccumulator(Storage.AssetStorage,BaseLogic.AssetCache) (contracts/BaseLogic.sol#316-331) uses a dangerous strict equality:
        - block.timestamp == assetCache.lastInterestAccumulatorUpdate (contracts/BaseLogic.sol#317)
BaseLogic.updateInterestRate(BaseLogic.AssetCache) (contracts/BaseLogic.sol#336-354) uses a dangerous strict equality:
        - total == 0 (contracts/BaseLogic.sol#342)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in BaseLogic.increaseBorrow(Storage.AssetStorage,BaseLogic.AssetCache,address,address,uint256) (contracts/BaseLogic.sol#369-388):
        External calls:
        - updateInterestRate(assetCache) (contracts/BaseLogic.sol#383)
                - (success,result) = moduleLookup[moduleId].delegatecall(input) (contracts/Base.sol#35)
        - emitViaProxy_Transfer(dTokenAddress,address(0),account,origAmount) (contracts/BaseLogic.sol#386)
                - (success) = proxyAddr.call(abi.encodePacked(uint8(3),keccak256(bytes)(bytes(Transfer(address,address,uint256))),bytes32(uint256(uint160(from))),bytes32(uint256(uint160(to))
),value)) (contracts/BaseModule.sol#46-52)
        State variables written after the call(s):
        - updateLastActivity(account) (contracts/BaseLogic.sol#387)
                - accountLookup[account].lastActivity = uint40(block.timestamp) (contracts/BaseLogic.sol#27)
Reentrancy in Governance.setIRM(address,uint256,bytes) (contracts/modules/Governance.sol#28-43):
        External calls:
        - assetCache = loadAssetCache(underlying,assetStorage) (contracts/modules/Governance.sol#33)
                - (success,data) = assetCache.underlying.staticcall{gas: 20000}(abi.encodeWithSelector(IERC20.balanceOf.selector,account)) (contracts/BaseLogic.sol#162)
        - callInternalModule(interestRateModel,abi.encodeWithSelector(IIRM.reset.selector,assetCache.underlying,resetParams)) (contracts/modules/Governance.sol#35)
                - (success,result) = moduleLookup[moduleId].delegatecall(input) (contracts/Base.sol#35)
        State variables written after the call(s):
        - assetStorage.interestRateModel = assetCache.interestRateModel = uint32(interestRateModel) (contracts/modules/Governance.sol#38)
Reentrancy in BaseLogic.transferBorrow(Storage.AssetStorage,BaseLogic.AssetCache,address,address,address,uint256) (contracts/BaseLogic.sol#416-443):
        External calls:
        - emitViaProxy_Transfer(dTokenAddress,from,to,origAmount) (contracts/BaseLogic.sol#440)
                - (success) = proxyAddr.call(abi.encodePacked(uint8(3),keccak256(bytes)(bytes(Transfer(address,address,uint256))),bytes32(uint256(uint160(from))),bytes32(uint256(uint160(to))
),value)) (contracts/BaseModule.sol#46-52)
```

## modules/Installer.sol

```
INFO:Detectors:
Base.callInternalModule(uint256,bytes) (contracts/Base.sol#34-38) uses delegatecall to a input-controlled function id
        - (success,result) = moduleLookup[moduleId].delegatecall(input) (contracts/Base.sol#35)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#controlled-delegatecall
INFO:Detectors:
Storage.upgradeAdmin (contracts/Storage.sol#12) is never initialized. It is used in:
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables
INFO:Detectors:
Installer.installModules(address[]).moduleId (contracts/modules/Installer.sol#21) shadows:
        - BaseModule.moduleId (contracts/BaseModule.sol#12) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Installer.installModules(address[]) (contracts/modules/Installer.sol#18-30) has external calls inside a loop: moduleId = IModule(moduleAddr).moduleId() (contracts/modules/Installer.sol#21)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
Base.revertBytes(bytes) (contracts/Base.sol#85-93) uses assembly
        - INLINE ASM (contracts/Base.sol#87-89)
BaseModule.unpackTrailingParamMsgSender() (contracts/BaseModule.sol#21-28) uses assembly
        - INLINE ASM (contracts/BaseModule.sol#22-27)
BaseModule.unpackTrailingParams() (contracts/BaseModule.sol#30-40) uses assembly
        - INLINE ASM (contracts/BaseModule.sol#31-39)
Proxy.fallback() (contracts/Proxy.sol#17-49) uses assembly
        - INLINE ASM (contracts/Proxy.sol#21-34)
        - INLINE ASM (contracts/Proxy.sol#36-47)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Base.callInternalModule(uint256,bytes) (contracts/Base.sol#34-38) is never used and should be removed
Base.revertBytes(bytes) (contracts/Base.sol#85-93) is never used and should be removed
BaseModule.emitViaProxy_Approval(address,address,address,uint256) (contracts/BaseModule.sol#56-65) is never used and should be removed
BaseModule.emitViaProxy_Transfer(address,address,address,uint256) (contracts/BaseModule.sol#45-54) is never used and should be removed
BaseModule.unpackTrailingParams() (contracts/BaseModule.sol#30-40) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (contracts/Base.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/BaseModule.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Constants.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Events.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Interfaces.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Proxy.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Storage.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/modules/Installer.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Base.callInternalModule(uint256,bytes) (contracts/Base.sol#34-38):
        - (success,result) = moduleLookup[moduleId].delegatecall(input) (contracts/Base.sol#35)
Low level call in BaseModule.emitViaProxy_Transfer(address,address,address,uint256) (contracts/BaseModule.sol#45-54):
        - (success) = proxyAddr.call(abi.encodePacked(uint8(3),keccak256(bytes)(bytes(Transfer(address,address,uint256))),bytes32(uint256(uint160(from))),bytes32(uint256(uint160(to))),value)
) (contracts/BaseModule.sol#46-52)
Low level call in BaseModule.emitViaProxy_Approval(address,address,address,uint256) (contracts/BaseModule.sol#56-65):
        - (success) = proxyAddr.call(abi.encodePacked(uint8(3),keccak256(bytes)(bytes(Approval(address,address,uint256))),bytes32(uint256(uint160(owner))),bytes32(uint256(uint160(spender))),
value)) (contracts/BaseModule.sol#57-63)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
```

## modules/Liquidation.sol

```
Storage.moduleLookup (contracts/Storage.sol#15) is never initialized. It is used in:
        - Base.callInternalModule(uint256,bytes) (contracts/Base.sol#34-38)
Storage.marketsEntered (contracts/Storage.sol#39) is never initialized. It is used in:
        - BaseLogic.getEnteredMarketsArray(address) (contracts/BaseLogic.sol#43-58)
        - BaseLogic.doEnterMarket(address,address) (contracts/BaseLogic.sol#60-72)
        - BaseLogic.doExitMarket(address,address) (contracts/BaseLogic.sol#76-95)
Storage.underlyingLookup (contracts/Storage.sol#85) is never initialized. It is used in:
        - Liquidation.liquidate(address,address,address) (contracts/modules/Liquidation.sol#18-69)
        - Liquidation.computeLiqOpp(Storage.AssetStorage,BaseLogic.AssetCache,Storage.AssetStorage,BaseLogic.AssetCache,ILiquidation.LiquidationOpportunity) (contracts/modules/Liquidation.so
l#72-157)
Storage.eTokenLookup (contracts/Storage.sol#86) is never initialized. It is used in:
        - Liquidation.liquidate(address,address,address) (contracts/modules/Liquidation.sol#18-69)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables
INFO:Detectors:
BaseLogic.roundUpOwed(BaseLogic.AssetCache,uint256) (contracts/BaseLogic.sol#290-297) performs a multiplication on the result of a division:
        -(owed + scale - 1) / scale * scale (contracts/BaseLogic.sol#295)
BaseLogic.updateInterestRate(BaseLogic.AssetCache) (contracts/BaseLogic.sol#336-354) performs a multiplication on the result of a division:
        -totalBorrows = assetCache.totalBorrows / INTERNAL_DEBT_PRECISION (contracts/BaseLogic.sol#340)
        -newUtilisation = uint32(totalBorrows * (uint256(type()(uint32).max) * 1e18) / total / 1e18) (contracts/BaseLogic.sol#343)
Liquidation.computeLiqOpp(Storage.AssetStorage,BaseLogic.AssetCache,Storage.AssetStorage,BaseLogic.AssetCache,ILiquidation.LiquidationOpportunity) (contracts/modules/Liquidation.sol#72-157)
performs a multiplication on the result of a division:
        -liqOpp.conversionRate = liqOpp.underlyingPrice * 1e18 / liqOpp.collateralPrice * 1e18 / (1e18 - liqOpp.discount) (contracts/modules/Liquidation.sol#103)
Liquidation.computeLiqOpp(Storage.AssetStorage,BaseLogic.AssetCache,Storage.AssetStorage,BaseLogic.AssetCache,ILiquidation.LiquidationOpportunity) (contracts/modules/Liquidation.sol#72-157)
performs a multiplication on the result of a division:
        -collateralAdj = 1e18 * uint256(collateralConfig.collateralFactor) / CONFIG_FACTOR_SCALE * 1e18 / (1e18 - liqOpp.discount) (contracts/modules/Liquidation.sol#118)
Liquidation.computeLiqOpp(Storage.AssetStorage,BaseLogic.AssetCache,Storage.AssetStorage,BaseLogic.AssetCache,ILiquidation.LiquidationOpportunity) (contracts/modules/Liquidation.sol#72-157)
performs a multiplication on the result of a division:
        -maxRepay = maxRepayInReference * 1e18 / liqOpp.underlyingPrice (contracts/modules/Liquidation.sol#130)
        -maxYield = maxRepay * liqOpp.conversionRate / 1e18 (contracts/modules/Liquidation.sol#144)
Liquidation.computeBonusScale(address,uint256) (contracts/modules/Liquidation.sol#175-197) performs a multiplication on the result of a division:
        -bonus = freeCollateralValue * 1e18 / violatorLiabilityValue (contracts/modules/Liquidation.sol#188)
        -bonus = bonus * (block.timestamp - lastActivity) / BONUS_REFRESH_PERIOD (contracts/modules/Liquidation.sol#191)
Liquidation.computeBonusScale(address,uint256) (contracts/modules/Liquidation.sol#175-197) performs a multiplication on the result of a division:
        -bonus = bonus * (block.timestamp - lastActivity) / BONUS_REFRESH_PERIOD (contracts/modules/Liquidation.sol#191)
        -bonus = bonus * (BONUS_SCALE - 1e18) / 1e18 (contracts/modules/Liquidation.sol#194)
RPow.rpow(uint256,uint256,uint256) (contracts/vendor/RPow.sol#23-45) performs a multiplication on the result of a division:
        -x = xxRound_rpow_asm_0 / base (contracts/vendor/RPow.sol#34)
        -zx_rpow_asm_0 = z * x (contracts/vendor/RPow.sol#36)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
BaseLogic.computeUpdatedInterestAccumulator(BaseLogic.AssetCache) (contracts/BaseLogic.sol#204-210) uses a dangerous strict equality:
        - lastInterestAccumulator == 0 (contracts/BaseLogic.sol#206)
BaseLogic.computeUpdatedInterestAccumulator(BaseLogic.AssetCache) (contracts/BaseLogic.sol#204-210) uses a dangerous strict equality:
        - deltaT == 0 (contracts/BaseLogic.sol#208)
Liquidation.liquidate(address,address,address) (contracts/modules/Liquidation.sol#18-69) uses a dangerous strict equality:
        - repayDesired == 0 (contracts/modules/Liquidation.sol#50)
BaseLogic.roundUpOwed(BaseLogic.AssetCache,uint256) (contracts/BaseLogic.sol#290-297) uses a dangerous strict equality:
        - owed == 0 (contracts/BaseLogic.sol#291)
BaseLogic.updateInterestAccumulator(Storage.AssetStorage,BaseLogic.AssetCache) (contracts/BaseLogic.sol#316-331) uses a dangerous strict equality:
        - block.timestamp == assetCache.lastInterestAccumulatorUpdate (contracts/BaseLogic.sol#317)
BaseLogic.updateInterestAccumulator(Storage.AssetStorage,BaseLogic.AssetCache) (contracts/BaseLogic.sol#316-331) uses a dangerous strict equality:
        - origInterestAccumulator == 0 (contracts/BaseLogic.sol#322)
BaseLogic.updateInterestRate(BaseLogic.AssetCache) (contracts/BaseLogic.sol#336-354) uses a dangerous strict equality:
        - total == 0 (contracts/BaseLogic.sol#342)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
```

## modules/Markets.sol

```
INFO:Detectors:
Storage.moduleLookup (contracts/Storage.sol#15) is never initialized. It is used in:
        - Base.callInternalModule(uint256,bytes) (contracts/Base.sol#34-38)
Storage.marketsEntered (contracts/Storage.sol#39) is never initialized. It is used in:
        - BaseLogic.getEnteredMarketsArray(address) (contracts/BaseLogic.sol#43-58)
        - BaseLogic.doEnterMarket(address,address) (contracts/BaseLogic.sol#60-72)
        - BaseLogic.doExitMarket(address,address) (contracts/BaseLogic.sol#76-95)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables
INFO:Detectors:
BaseLogic.roundUpOwed(BaseLogic.AssetCache,uint256) (contracts/BaseLogic.sol#290-297) performs a multiplication on the result of a division:
        -(owed + scale - 1) / scale * scale (contracts/BaseLogic.sol#295)
BaseLogic.updateInterestRate(BaseLogic.AssetCache) (contracts/BaseLogic.sol#336-354) performs a multiplication on the result of a division:
        -totalBorrows = assetCache.totalBorrows / INTERNAL_DEBT_PRECISION (contracts/BaseLogic.sol#340)
        -newUtilisation = uint32(totalBorrows * (uint256(type()(uint32).max) * 1e18) / total / 1e18) (contracts/BaseLogic.sol#343)
RPow.rpow(uint256,uint256,uint256) (contracts/vendor/RPow.sol#23-45) performs a multiplication on the result of a division:
        -x = xxRound_rpow_asm_0 / base (contracts/vendor/RPow.sol#34)
        -zx_rpow_asm_0 = z * x (contracts/vendor/RPow.sol#36)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
BaseLogic.computeUpdatedInterestAccumulator(BaseLogic.AssetCache) (contracts/BaseLogic.sol#204-210) uses a dangerous strict equality:
        - deltaT == 0 (contracts/BaseLogic.sol#208)
BaseLogic.roundUpOwed(BaseLogic.AssetCache,uint256) (contracts/BaseLogic.sol#290-297) uses a dangerous strict equality:
        - owed == 0 (contracts/BaseLogic.sol#291)
BaseLogic.updateInterestAccumulator(Storage.AssetStorage,BaseLogic.AssetCache) (contracts/BaseLogic.sol#316-331) uses a dangerous strict equality:
        - block.timestamp == assetCache.lastInterestAccumulatorUpdate (contracts/BaseLogic.sol#317)
BaseLogic.updateInterestRate(BaseLogic.AssetCache) (contracts/BaseLogic.sol#336-354) uses a dangerous strict equality:
        - total == 0 (contracts/BaseLogic.sol#342)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in Markets.activateMarket(address) (contracts/modules/Markets.sol#12-63):
        External calls:
        - result = callInternalModule(MODULEID__RISK_MANAGER,abi.encodeWithSelector(IRiskManager.getNewMarketParameters.selector,underlying)) (contracts/modules/Markets.sol#31-32)
                - (success,result) = moduleLookup[moduleId].delegatecall(input) (contracts/Base.sol#35)
        State variables written after the call(s):
        - childEToken = params.config.eTokenAddress = _createProxy(MODULEID__ETOKEN) (contracts/modules/Markets.sol#39)
                - trustedSenders[proxyAddr] = TrustedSenderInfo(uint32(proxyModuleId),address(0)) (contracts/Base.sol#27)
        - childDToken = _createProxy(MODULEID__DTOKEN) (contracts/modules/Markets.sol#40)
                - trustedSenders[proxyAddr] = TrustedSenderInfo(uint32(proxyModuleId),address(0)) (contracts/Base.sol#27)
        - underlyingLookup[underlying] = params.config (contracts/modules/Markets.sol#45)
Reentrancy in BaseLogic.increaseBorrow(Storage.AssetStorage,BaseLogic.AssetCache,address,address,uint256) (contracts/BaseLogic.sol#369-388):
        External calls:
        - updateInterestRate(assetCache) (contracts/BaseLogic.sol#383)
                - (success,result) = moduleLookup[moduleId].delegatecall(input) (contracts/Base.sol#35)
        - emitViaProxy_Transfer(dTokenAddress,address(0),account,origAmount) (contracts/BaseLogic.sol#386)
                - (success) = proxyAddr.call(abi.encodePacked(uint8(3),keccak256(bytes)(bytes(Transfer(address,address,uint256))),bytes32(uint256(uint160(from))),bytes32(uint256(uint160(to))
),value)) (contracts/BaseModule.sol#46-52)
        State variables written after the call(s):
        - updateLastActivity(account) (contracts/BaseLogic.sol#387)
                - accountLookup[account].lastActivity = uint40(block.timestamp) (contracts/BaseLogic.sol#27)
Reentrancy in BaseLogic.transferBorrow(Storage.AssetStorage,BaseLogic.AssetCache,address,address,address,uint256) (contracts/BaseLogic.sol#416-443):
        External calls:
        - emitViaProxy_Transfer(dTokenAddress,from,to,origAmount) (contracts/BaseLogic.sol#440)
                - (success) = proxyAddr.call(abi.encodePacked(uint8(3),keccak256(bytes)(bytes(Transfer(address,address,uint256))),bytes32(uint256(uint160(from))),bytes32(uint256(uint160(to))
),value)) (contracts/BaseModule.sol#46-52)
```

## modules/RiskManager.sol

```
ERROR:ContractSolcParsing:Missing function 'name'
INFO:Detectors:
Storage.underlyingLookup (contracts/Storage.sol#85) is never initialized. It is used in:
        - RiskManager.getPrice(address) (contracts/modules/RiskManager.sol#175-181)
        - RiskManager.getPriceFull(address) (contracts/modules/RiskManager.sol#186-202)
        - RiskManager.computeLiquidityRaw(address,address[]) (contracts/modules/RiskManager.sol#207-251)
Storage.eTokenLookup (contracts/Storage.sol#86) is never initialized. It is used in:
        - RiskManager.getPrice(address) (contracts/modules/RiskManager.sol#175-181)
        - RiskManager.getPriceFull(address) (contracts/modules/RiskManager.sol#186-202)
        - RiskManager.computeLiquidityRaw(address,address[]) (contracts/modules/RiskManager.sol#207-251)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables
INFO:Detectors:
BaseLogic.roundUpOwed(BaseLogic.AssetCache,uint256) (contracts/BaseLogic.sol#290-297) performs a multiplication on the result of a division:
        -(owed + scale - 1) / scale * scale (contracts/BaseLogic.sol#295)
BaseLogic.updateInterestRate(BaseLogic.AssetCache) (contracts/BaseLogic.sol#336-354) performs a multiplication on the result of a division:
        -totalBorrows = assetCache.totalBorrows / INTERNAL_DEBT_PRECISION (contracts/BaseLogic.sol#340)
        -newUtilisation = uint32(totalBorrows * (uint256(type()(uint32).max) * 1e18) / total / 1e18) (contracts/BaseLogic.sol#343)
RiskManager.computeLiquidityRaw(address,address[]) (contracts/modules/RiskManager.sol#207-251) performs a multiplication on the result of a division:
        -assetCollateral = assetCollateral * price / 1e18 (contracts/modules/RiskManager.sol#232)
        -assetCollateral = assetCollateral * config.collateralFactor / CONFIG_FACTOR_SCALE (contracts/modules/RiskManager.sol#233)
RiskManager.computeLiquidityRaw(address,address[]) (contracts/modules/RiskManager.sol#207-251) performs a multiplication on the result of a division:
        -assetLiability = assetLiability * price / 1e18 (contracts/modules/RiskManager.sol#245)
        -assetLiability = assetLiability * CONFIG_FACTOR_SCALE / config.borrowFactor (contracts/modules/RiskManager.sol#246)
RPow.rpow(uint256,uint256,uint256) (contracts/vendor/RPow.sol#23-45) performs a multiplication on the result of a division:
        -x = xxRound_rpow_asm_0 / base (contracts/vendor/RPow.sol#34)
        -zx_rpow_asm_0 = z * x (contracts/vendor/RPow.sol#36)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
Reentrancy in BaseLogic.increaseBorrow(Storage.AssetStorage,BaseLogic.AssetCache,address,address,uint256) (contracts/BaseLogic.sol#369-388):
        External calls:
        - updateInterestRate(assetCache) (contracts/BaseLogic.sol#383)
                - (success,result) = moduleLookup[moduleId].delegatecall(input) (contracts/Base.sol#35)
        - emitViaProxy_Transfer(dTokenAddress,address(0),account,origAmount) (contracts/BaseLogic.sol#386)
                - (success) = proxyAddr.call(abi.encodePacked(uint8(3),keccak256(bytes)(bytes(Transfer(address,address,uint256))),bytes32(uint256(uint160(from))),bytes32(uint256(uint160(to))
),value)) (contracts/BaseModule.sol#46-52)
        State variables written after the call(s):
        - updateLastActivity(account) (contracts/BaseLogic.sol#387)
                - accountLookup[account].lastActivity = uint40(block.timestamp) (contracts/BaseLogic.sol#27)
Reentrancy in BaseLogic.transferBorrow(Storage.AssetStorage,BaseLogic.AssetCache,address,address,address,uint256) (contracts/BaseLogic.sol#416-443):
        External calls:
        - emitViaProxy_Transfer(dTokenAddress,from,to,origAmount) (contracts/BaseLogic.sol#440)
                - (success) = proxyAddr.call(abi.encodePacked(uint8(3),keccak256(bytes)(bytes(Transfer(address,address,uint256))),bytes32(uint256(uint160(from))),bytes32(uint256(uint160(to))
),value)) (contracts/BaseModule.sol#46-52)
        State variables written after the call(s):
        - updateLastActivity(from) (contracts/BaseLogic.sol#441)
                - accountLookup[account].lastActivity = uint40(block.timestamp) (contracts/BaseLogic.sol#27)
        - updateLastActivity(to) (contracts/BaseLogic.sol#442)
                - accountLookup[account].lastActivity = uint40(block.timestamp) (contracts/BaseLogic.sol#27)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
RiskManager.getNewMarketParameters(address).fee (contracts/modules/RiskManager.sol#55) is a local variable never initialized
RiskManager.getNewMarketParameters(address).returnData (contracts/modules/RiskManager.sol#72) is a local variable never initialized
RiskManager.getNewMarketParameters(address).err (contracts/modules/RiskManager.sol#69) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
```

## modules/interest-rate-models/IRMDefault.sol

```
Compilation warnings/errors on contracts/modules/interest-rate-models/IRMDefault.sol:
Warning: SPDX license identifier not provided in source file. Before publishing, consider adding a comment containing "SPDX-License-Identifier: <SPDX-License>" to each source file. Use "SPDX
-License-Identifier: UNLICENSED" for non-open-source code. Please see https://spdx.org for more information.
--> contracts/modules/interest-rate-models/IRMDefault.sol


INFO:Detectors:
Storage.moduleLookup (contracts/Storage.sol#15) is never initialized. It is used in:
        - Base.callInternalModule(uint256,bytes) (contracts/Base.sol#34-38)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables
INFO:Detectors:
Base.revertBytes(bytes) (contracts/Base.sol#85-93) uses assembly
        - INLINE ASM (contracts/Base.sol#87-89)
BaseModule.unpackTrailingParamMsgSender() (contracts/BaseModule.sol#21-28) uses assembly
        - INLINE ASM (contracts/BaseModule.sol#22-27)
BaseModule.unpackTrailingParams() (contracts/BaseModule.sol#30-40) uses assembly
        - INLINE ASM (contracts/BaseModule.sol#31-39)
Proxy.fallback() (contracts/Proxy.sol#17-49) uses assembly
        - INLINE ASM (contracts/Proxy.sol#21-34)
        - INLINE ASM (contracts/Proxy.sol#36-47)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Base._createProxy(uint256) (contracts/Base.sol#13-32) is never used and should be removed
Base.callInternalModule(uint256,bytes) (contracts/Base.sol#34-38) is never used and should be removed
Base.revertBytes(bytes) (contracts/Base.sol#85-93) is never used and should be removed
BaseModule.emitViaProxy_Approval(address,address,address,uint256) (contracts/BaseModule.sol#56-65) is never used and should be removed
BaseModule.emitViaProxy_Transfer(address,address,address,uint256) (contracts/BaseModule.sol#45-54) is never used and should be removed
BaseModule.unpackTrailingParamMsgSender() (contracts/BaseModule.sol#21-28) is never used and should be removed
BaseModule.unpackTrailingParams() (contracts/BaseModule.sol#30-40) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (contracts/Base.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/BaseIRM.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/BaseModule.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Constants.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Events.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Interfaces.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Proxy.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Storage.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/modules/interest-rate-models/IRMDefault.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Base.callInternalModule(uint256,bytes) (contracts/Base.sol#34-38):
        - (success,result) = moduleLookup[moduleId].delegatecall(input) (contracts/Base.sol#35)
Low level call in BaseModule.emitViaProxy_Transfer(address,address,address,uint256) (contracts/BaseModule.sol#45-54):
        - (success) = proxyAddr.call(abi.encodePacked(uint8(3),keccak256(bytes)(bytes(Transfer(address,address,uint256))),bytes32(uint256(uint160(from))),bytes32(uint256(uint160(to))),value)
) (contracts/BaseModule.sol#46-52)
Low level call in BaseModule.emitViaProxy_Approval(address,address,address,uint256) (contracts/BaseModule.sol#56-65):
        - (success) = proxyAddr.call(abi.encodePacked(uint8(3),keccak256(bytes)(bytes(Approval(address,address,uint256))),bytes32(uint256(uint160(owner))),bytes32(uint256(uint160(spender))),
value)) (contracts/BaseModule.sol#57-63)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Modifier Base.FREEMEM() (contracts/Base.sol#62-79) is not in mixedCase
Function BaseModule.emitViaProxy_Transfer(address,address,address,uint256) (contracts/BaseModule.sol#45-54) is not in mixedCase
```

## modules/interest-rate-models/IRMFixed.sol

```
INFO:Detectors:
Storage.moduleLookup (contracts/Storage.sol#15) is never initialized. It is used in:
        - Base.callInternalModule(uint256,bytes) (contracts/Base.sol#34-38)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables
INFO:Detectors:
Base.revertBytes(bytes) (contracts/Base.sol#85-93) uses assembly
        - INLINE ASM (contracts/Base.sol#87-89)
BaseModule.unpackTrailingParamMsgSender() (contracts/BaseModule.sol#21-28) uses assembly
        - INLINE ASM (contracts/BaseModule.sol#22-27)
BaseModule.unpackTrailingParams() (contracts/BaseModule.sol#30-40) uses assembly
        - INLINE ASM (contracts/BaseModule.sol#31-39)
Proxy.fallback() (contracts/Proxy.sol#17-49) uses assembly
        - INLINE ASM (contracts/Proxy.sol#21-34)
        - INLINE ASM (contracts/Proxy.sol#36-47)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Base._createProxy(uint256) (contracts/Base.sol#13-32) is never used and should be removed
Base.callInternalModule(uint256,bytes) (contracts/Base.sol#34-38) is never used and should be removed
Base.revertBytes(bytes) (contracts/Base.sol#85-93) is never used and should be removed
BaseModule.emitViaProxy_Approval(address,address,address,uint256) (contracts/BaseModule.sol#56-65) is never used and should be removed
BaseModule.emitViaProxy_Transfer(address,address,address,uint256) (contracts/BaseModule.sol#45-54) is never used and should be removed
BaseModule.unpackTrailingParamMsgSender() (contracts/BaseModule.sol#21-28) is never used and should be removed
BaseModule.unpackTrailingParams() (contracts/BaseModule.sol#30-40) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (contracts/Base.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/BaseIRM.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/BaseModule.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Constants.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Events.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Interfaces.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Proxy.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Storage.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/modules/interest-rate-models/IRMFixed.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Base.callInternalModule(uint256,bytes) (contracts/Base.sol#34-38):
        - (success,result) = moduleLookup[moduleId].delegatecall(input) (contracts/Base.sol#35)
Low level call in BaseModule.emitViaProxy_Transfer(address,address,address,uint256) (contracts/BaseModule.sol#45-54):
        - (success) = proxyAddr.call(abi.encodePacked(uint8(3),keccak256(bytes)(bytes(Transfer(address,address,uint256))),bytes32(uint256(uint160(from))),bytes32(uint256(uint160(to))),value)
) (contracts/BaseModule.sol#46-52)
Low level call in BaseModule.emitViaProxy_Approval(address,address,address,uint256) (contracts/BaseModule.sol#56-65):
        - (success) = proxyAddr.call(abi.encodePacked(uint8(3),keccak256(bytes)(bytes(Approval(address,address,uint256))),bytes32(uint256(uint160(owner))),bytes32(uint256(uint160(spender))),
value)) (contracts/BaseModule.sol#57-63)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Modifier Base.FREEMEM() (contracts/Base.sol#62-79) is not in mixedCase
Function BaseModule.emitViaProxy_Transfer(address,address,address,uint256) (contracts/BaseModule.sol#45-54) is not in mixedCase
Function BaseModule.emitViaProxy_Approval(address,address,address,uint256) (contracts/BaseModule.sol#56-65) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable Constants.MODULEID__DTOKEN (contracts/Constants.sol#42) is too similar to Constants.MODULEID__ETOKEN (contracts/Constants.sol#41)
Variable Storage.dTokenLookup (contracts/Storage.sol#87) is too similar to Storage.eTokenLookup (contracts/Storage.sol#86)
```

## modules/interest-rate-models/IRMLinearRecursive.sol

```
INFO:Detectors:
Storage.moduleLookup (contracts/Storage.sol#15) is never initialized. It is used in:
        - Base.callInternalModule(uint256,bytes) (contracts/Base.sol#34-38)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables
INFO:Detectors:
Base.revertBytes(bytes) (contracts/Base.sol#85-93) uses assembly
        - INLINE ASM (contracts/Base.sol#87-89)
BaseModule.unpackTrailingParamMsgSender() (contracts/BaseModule.sol#21-28) uses assembly
        - INLINE ASM (contracts/BaseModule.sol#22-27)
BaseModule.unpackTrailingParams() (contracts/BaseModule.sol#30-40) uses assembly
        - INLINE ASM (contracts/BaseModule.sol#31-39)
Proxy.fallback() (contracts/Proxy.sol#17-49) uses assembly
        - INLINE ASM (contracts/Proxy.sol#21-34)
        - INLINE ASM (contracts/Proxy.sol#36-47)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Base._createProxy(uint256) (contracts/Base.sol#13-32) is never used and should be removed
Base.callInternalModule(uint256,bytes) (contracts/Base.sol#34-38) is never used and should be removed
Base.revertBytes(bytes) (contracts/Base.sol#85-93) is never used and should be removed
BaseModule.emitViaProxy_Approval(address,address,address,uint256) (contracts/BaseModule.sol#56-65) is never used and should be removed
BaseModule.emitViaProxy_Transfer(address,address,address,uint256) (contracts/BaseModule.sol#45-54) is never used and should be removed
BaseModule.unpackTrailingParamMsgSender() (contracts/BaseModule.sol#21-28) is never used and should be removed
BaseModule.unpackTrailingParams() (contracts/BaseModule.sol#30-40) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (contracts/Base.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/BaseIRM.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/BaseModule.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Constants.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Events.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Interfaces.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Proxy.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Storage.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/modules/interest-rate-models/IRMLinearRecursive.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Base.callInternalModule(uint256,bytes) (contracts/Base.sol#34-38):
        - (success,result) = moduleLookup[moduleId].delegatecall(input) (contracts/Base.sol#35)
Low level call in BaseModule.emitViaProxy_Transfer(address,address,address,uint256) (contracts/BaseModule.sol#45-54):
        - (success) = proxyAddr.call(abi.encodePacked(uint8(3),keccak256(bytes)(bytes(Transfer(address,address,uint256))),bytes32(uint256(uint160(from))),bytes32(uint256(uint160(to))),value)
) (contracts/BaseModule.sol#46-52)
Low level call in BaseModule.emitViaProxy_Approval(address,address,address,uint256) (contracts/BaseModule.sol#56-65):
        - (success) = proxyAddr.call(abi.encodePacked(uint8(3),keccak256(bytes)(bytes(Approval(address,address,uint256))),bytes32(uint256(uint160(owner))),bytes32(uint256(uint160(spender))),
value)) (contracts/BaseModule.sol#57-63)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Modifier Base.FREEMEM() (contracts/Base.sol#62-79) is not in mixedCase
Function BaseModule.emitViaProxy_Transfer(address,address,address,uint256) (contracts/BaseModule.sol#45-54) is not in mixedCase
Function BaseModule.emitViaProxy_Approval(address,address,address,uint256) (contracts/BaseModule.sol#56-65) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable Constants.MODULEID__DTOKEN (contracts/Constants.sol#42) is too similar to Constants.MODULEID__ETOKEN (contracts/Constants.sol#41)
Variable Storage.dTokenLookup (contracts/Storage.sol#87) is too similar to Storage.eTokenLookup (contracts/Storage.sol#86)
```

## modules/interest-rate-models/IRMLinear.sol

```
INFO:Detectors:
Storage.moduleLookup (contracts/Storage.sol#15) is never initialized. It is used in:
        - Base.callInternalModule(uint256,bytes) (contracts/Base.sol#34-38)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables
INFO:Detectors:
Base.revertBytes(bytes) (contracts/Base.sol#85-93) uses assembly
        - INLINE ASM (contracts/Base.sol#87-89)
BaseModule.unpackTrailingParamMsgSender() (contracts/BaseModule.sol#21-28) uses assembly
        - INLINE ASM (contracts/BaseModule.sol#22-27)
BaseModule.unpackTrailingParams() (contracts/BaseModule.sol#30-40) uses assembly
        - INLINE ASM (contracts/BaseModule.sol#31-39)
Proxy.fallback() (contracts/Proxy.sol#17-49) uses assembly
        - INLINE ASM (contracts/Proxy.sol#21-34)
        - INLINE ASM (contracts/Proxy.sol#36-47)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Base._createProxy(uint256) (contracts/Base.sol#13-32) is never used and should be removed
Base.callInternalModule(uint256,bytes) (contracts/Base.sol#34-38) is never used and should be removed
Base.revertBytes(bytes) (contracts/Base.sol#85-93) is never used and should be removed
BaseModule.emitViaProxy_Approval(address,address,address,uint256) (contracts/BaseModule.sol#56-65) is never used and should be removed
BaseModule.emitViaProxy_Transfer(address,address,address,uint256) (contracts/BaseModule.sol#45-54) is never used and should be removed
BaseModule.unpackTrailingParamMsgSender() (contracts/BaseModule.sol#21-28) is never used and should be removed
BaseModule.unpackTrailingParams() (contracts/BaseModule.sol#30-40) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (contracts/Base.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/BaseIRM.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/BaseModule.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Constants.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Events.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Interfaces.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Proxy.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Storage.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/modules/interest-rate-models/IRMLinear.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Base.callInternalModule(uint256,bytes) (contracts/Base.sol#34-38):
        - (success,result) = moduleLookup[moduleId].delegatecall(input) (contracts/Base.sol#35)
Low level call in BaseModule.emitViaProxy_Transfer(address,address,address,uint256) (contracts/BaseModule.sol#45-54):
        - (success) = proxyAddr.call(abi.encodePacked(uint8(3),keccak256(bytes)(bytes(Transfer(address,address,uint256))),bytes32(uint256(uint160(from))),bytes32(uint256(uint160(to))),value)
) (contracts/BaseModule.sol#46-52)
Low level call in BaseModule.emitViaProxy_Approval(address,address,address,uint256) (contracts/BaseModule.sol#56-65):
        - (success) = proxyAddr.call(abi.encodePacked(uint8(3),keccak256(bytes)(bytes(Approval(address,address,uint256))),bytes32(uint256(uint160(owner))),bytes32(uint256(uint160(spender))),
value)) (contracts/BaseModule.sol#57-63)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Modifier Base.FREEMEM() (contracts/Base.sol#62-79) is not in mixedCase
Function BaseModule.emitViaProxy_Transfer(address,address,address,uint256) (contracts/BaseModule.sol#45-54) is not in mixedCase
Function BaseModule.emitViaProxy_Approval(address,address,address,uint256) (contracts/BaseModule.sol#56-65) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable Constants.MODULEID__DTOKEN (contracts/Constants.sol#42) is too similar to Constants.MODULEID__ETOKEN (contracts/Constants.sol#41)
Variable Storage.dTokenLookup (contracts/Storage.sol#87) is too similar to Storage.eTokenLookup (contracts/Storage.sol#86)
```

modules/interest-rate-models/IRMZero.sol

```
INFO:Detectors:
Storage.moduleLookup (contracts/Storage.sol#15) is never initialized. It is used in:
        - Base.callInternalModule(uint256,bytes) (contracts/Base.sol#34-38)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables
INFO:Detectors:
Base.revertBytes(bytes) (contracts/Base.sol#85-93) uses assembly
        - INLINE ASM (contracts/Base.sol#87-89)
BaseModule.unpackTrailingParamMsgSender() (contracts/BaseModule.sol#21-28) uses assembly
        - INLINE ASM (contracts/BaseModule.sol#22-27)
BaseModule.unpackTrailingParams() (contracts/BaseModule.sol#30-40) uses assembly
        - INLINE ASM (contracts/BaseModule.sol#31-39)
Proxy.fallback() (contracts/Proxy.sol#17-49) uses assembly
        - INLINE ASM (contracts/Proxy.sol#21-34)
        - INLINE ASM (contracts/Proxy.sol#36-47)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Base._createProxy(uint256) (contracts/Base.sol#13-32) is never used and should be removed
Base.callInternalModule(uint256,bytes) (contracts/Base.sol#34-38) is never used and should be removed
Base.revertBytes(bytes) (contracts/Base.sol#85-93) is never used and should be removed
BaseModule.emitViaProxy_Approval(address,address,address,uint256) (contracts/BaseModule.sol#56-65) is never used and should be removed
BaseModule.emitViaProxy_Transfer(address,address,address,uint256) (contracts/BaseModule.sol#45-54) is never used and should be removed
BaseModule.unpackTrailingParamMsgSender() (contracts/BaseModule.sol#21-28) is never used and should be removed
BaseModule.unpackTrailingParams() (contracts/BaseModule.sol#30-40) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (contracts/Base.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/BaseIRM.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/BaseModule.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Constants.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Events.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Interfaces.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Proxy.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Storage.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/modules/interest-rate-models/IRMZero.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Base.callInternalModule(uint256,bytes) (contracts/Base.sol#34-38):
        - (success,result) = moduleLookup[moduleId].delegatecall(input) (contracts/Base.sol#35)
Low level call in BaseModule.emitViaProxy_Transfer(address,address,address,uint256) (contracts/BaseModule.sol#45-54):
        - (success) = proxyAddr.call(abi.encodePacked(uint8(3),keccak256(bytes)(bytes(Transfer(address,address,uint256))),bytes32(uint256(uint160(from))),bytes32(uint256(uint160(to))),value)
) (contracts/BaseModule.sol#46-52)
Low level call in BaseModule.emitViaProxy_Approval(address,address,address,uint256) (contracts/BaseModule.sol#56-65):
        - (success) = proxyAddr.call(abi.encodePacked(uint8(3),keccak256(bytes)(bytes(Approval(address,address,uint256))),bytes32(uint256(uint160(owner))),bytes32(uint256(uint160(spender))),
value)) (contracts/BaseModule.sol#57-63)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Modifier Base.FREEMEM() (contracts/Base.sol#62-79) is not in mixedCase
Function BaseModule.emitViaProxy_Transfer(address,address,address,uint256) (contracts/BaseModule.sol#45-54) is not in mixedCase
Function BaseModule.emitViaProxy_Approval(address,address,address,uint256) (contracts/BaseModule.sol#56-65) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable Constants.MODULEID__DTOKEN (contracts/Constants.sol#42) is too similar to Constants.MODULEID__ETOKEN (contracts/Constants.sol#41)
Variable Storage.dTokenLookup (contracts/Storage.sol#87) is too similar to Storage.eTokenLookup (contracts/Storage.sol#86)
```

views_EulerGeneralView.sol

```
ERROR:ContractSolcParsing:Missing function 'name'
INFO:Detectors:
RPow.rpow(uint256,uint256,uint256) (contracts/vendor/RPow.sol#23-45) performs a multiplication on the result of a division:
        -x = xxRound_rpow_asm_0 / base (contracts/vendor/RPow.sol#34)
        -zx_rpow_asm_0 = z * x (contracts/vendor/RPow.sol#36)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
RPow.rpow(uint256,uint256,uint256) (contracts/vendor/RPow.sol#23-45) uses assembly
        - INLINE ASM (contracts/vendor/RPow.sol#24-44)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
EulerGeneralView.populateResponseMarket(EulerGeneralView.Query,EulerGeneralView.ResponseMarket,IMarkets,IExec) (contracts/views/EulerGeneralView.sol#108-142) is never used and should be remo
ved
RPow.rpow(uint256,uint256,uint256) (contracts/vendor/RPow.sol#23-45) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (contracts/Constants.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Interfaces.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/Storage.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/vendor/RPow.sol#20) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (contracts/views/EulerGeneralView.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Variable Constants.MODULEID__DTOKEN (contracts/Constants.sol#42) is too similar to Constants.MODULEID__ETOKEN (contracts/Constants.sol#41)
Variable Storage.dTokenLookup (contracts/Storage.sol#87) is too similar to Storage.eTokenLookup (contracts/Storage.sol#86)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar
INFO:Detectors:
EulerGeneralView (contracts/views/EulerGeneralView.sol#12-143) does not implement functions:
        - EulerGeneralView.populateResponseMarket(EulerGeneralView.Query,EulerGeneralView.ResponseMarket,IMarkets,IExec) (contracts/views/EulerGeneralView.sol#108-142)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions
INFO:Detectors:
Constants.MAX_SANE_AMOUNT (contracts/Constants.sol#8) is never used in EulerGeneralView (contracts/views/EulerGeneralView.sol#12-143)
Constants.MAX_SANE_DEBT_AMOUNT (contracts/Constants.sol#9) is never used in EulerGeneralView (contracts/views/EulerGeneralView.sol#12-143)
Constants.INTERNAL_DEBT_PRECISION (contracts/Constants.sol#10) is never used in EulerGeneralView (contracts/views/EulerGeneralView.sol#12-143)
Constants.MAX_ENTERED_MARKETS (contracts/Constants.sol#11) is never used in EulerGeneralView (contracts/views/EulerGeneralView.sol#12-143)
Constants.MAX_POSSIBLE_ENTERED_MARKETS (contracts/Constants.sol#12) is never used in EulerGeneralView (contracts/views/EulerGeneralView.sol#12-143)
Constants.CONFIG_FACTOR_SCALE (contracts/Constants.sol#13) is never used in EulerGeneralView (contracts/views/EulerGeneralView.sol#12-143)
Constants.INITIAL_INTEREST_ACCUMULATOR (contracts/Constants.sol#14) is never used in EulerGeneralView (contracts/views/EulerGeneralView.sol#12-143)
Constants.REENTRANCYLOCK__UNLOCKED (contracts/Constants.sol#19) is never used in EulerGeneralView (contracts/views/EulerGeneralView.sol#12-143)
Constants.REENTRANCYLOCK__LOCKED (contracts/Constants.sol#20) is never used in EulerGeneralView (contracts/views/EulerGeneralView.sol#12-143)
Constants.PRICINGTYPE__PEGGED (contracts/Constants.sol#25) is never used in EulerGeneralView (contracts/views/EulerGeneralView.sol#12-143)
Constants.PRICINGTYPE__UNISWAP3_TWAP (contracts/Constants.sol#26) is never used in EulerGeneralView (contracts/views/EulerGeneralView.sol#12-143)
Constants.MODULEID__INSTALLER (contracts/Constants.sol#32) is never used in EulerGeneralView (contracts/views/EulerGeneralView.sol#12-143)
Constants.MODULEID__LIQUIDATION (contracts/Constants.sol#34) is never used in EulerGeneralView (contracts/views/EulerGeneralView.sol#12-143)
Constants.MODULEID__GOVERNANCE (contracts/Constants.sol#35) is never used in EulerGeneralView (contracts/views/EulerGeneralView.sol#12-143)
Constants.MAX_EXTERNAL_SINGLE_PROXY_MODULEID (contracts/Constants.sol#38) is never used in EulerGeneralView (contracts/views/EulerGeneralView.sol#12-143)
Constants.MODULEID__ETOKEN (contracts/Constants.sol#41) is never used in EulerGeneralView (contracts/views/EulerGeneralView.sol#12-143)
Constants.MODULEID__DTOKEN (contracts/Constants.sol#42) is never used in EulerGeneralView (contracts/views/EulerGeneralView.sol#12-143)
Constants.MAX_EXTERNAL_MODULEID (contracts/Constants.sol#44) is never used in EulerGeneralView (contracts/views/EulerGeneralView.sol#12-143)
Constants.MODULEID__RISK_MANAGER (contracts/Constants.sol#47) is never used in EulerGeneralView (contracts/views/EulerGeneralView.sol#12-143)
Constants.MODULEID__IRM_DEFAULT (contracts/Constants.sol#50) is never used in EulerGeneralView (contracts/views/EulerGeneralView.sol#12-143)
Constants.MODULEID__IRM_ZERO (contracts/Constants.sol#51) is never used in EulerGeneralView (contracts/views/EulerGeneralView.sol#12-143)
```

- All multiplication on the result of a division issues were manually checked and the code logic is aware of it. This is used to truncate

value precision from internal to external representation and is expected behaviour.

- All `is never initialized` issues are false positive since the compiler is not aware of the modularity of the code. The variables are declared on the `Storage` contract which all of the contracts depend on.
- Re-entrancy issue is not present since a lock is used to protect against using a modifier named `nonReentrant`.
- Issue regarding `floating pragma` has been already mentioned in the above report.
- Use of `block.timestamp` issue is already mentioned in the above report.

# 5.2 AUTOMATED SECURITY SCAN

MYTHX:

Halborn used automated security scanners to assist with detection of well-known security issues, and to identify low-hanging fruit 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 testers machine and sent the compiled results to the analyzers to locate any vulnerabilities. Only security-related findings are shown below.

Results:

BaseIRM.sol

```
Report for BaseIRM.sol
https://dashboard.mythx.io/#/console/analyses/b0f34a0e-aff1-4ac8-b47f-e51c80f9d0aa
```

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

```
Report for BaseIRM.sol
https://dashboard.mythx.io/#/console/analyses/3eed9a04-cf18-4c9b-8917-3792b3c33ec1
```

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

AUTOMATED TESTING

BaseLogic.sol

```
Report for BaseLogic.sol
https://dashboard.mythx.io/#/console/analyses/ab49394c-5894-4efd-b2cb-f36a4ae5615c
```

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

```
Report for BaseLogic.sol
https://dashboard.mythx.io/#/console/analyses/f61dfe37-827f-4fb0-8c75-1d7582478c68
```

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

BaseModule.sol

```
Report for BaseModule.sol
https://dashboard.mythx.io/#/console/analyses/ba315d89-7953-442f-9b08-26d3fecfbf49
```

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

```
Report for BaseModule.sol
https://dashboard.mythx.io/#/console/analyses/9476fa09-cefa-411d-b92d-e0a403a9fc01
```

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

Base.sol

```
Report for Base.sol
https://dashboard.mythx.io/#/console/analyses/403719ae-be16-4d85-b86a-29fc9e9868ee
```

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

```
Report for Base.sol
https://dashboard.mythx.io/#/console/analyses/ac7c996d-c30c-473e-bb96-474884e71b05
```

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

AUTOMATED TESTING

## Euler.sol

```
Report for Euler.sol
https://dashboard.mythx.io/#/console/analyses/52447aca-005a-4298-bca0-cc7b4c07b546
```

| Line | SWC Title | Severity | Short Description |
|---|---|---|---|
| 3 | (SWC-103) Floating Pragma | Low | A floating pragma is set. |
| 9 | (SWC-100) Function Default Visibility | Low | Function visibility is not set. |

```
Report for Euler.sol
https://dashboard.mythx.io/#/console/analyses/349f368e-38d9-4f90-9be9-65bc65d0bf85
```

| Line | SWC Title | Severity | Short Description |
|---|---|---|---|
| 3 | (SWC-103) Floating Pragma | Low | A floating pragma is set. |
| 9 | (SWC-100) Function Default Visibility | Low | Function visibility is not set. |

## Proxy.sol

```
Report for Proxy.sol
https://dashboard.mythx.io/#/console/analyses/0ea43a90-9bbe-4d79-850d-c44797d34a3c
```

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

```
Report for Proxy.sol
https://dashboard.mythx.io/#/console/analyses/e67bab04-c6a0-442a-bec5-6c7d3894be41
```

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

## modules/DToken.sol

```
Report for modules/DToken.sol
https://dashboard.mythx.io/#/console/analyses/8b345c60-f596-4f46-8f3e-c54c54d1dd7c
```

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

AUTOMATED TESTING

modules/EToken.sol

```
Report for modules/EToken.sol
https://dashboard.mythx.io/#/console/analyses/9fdc63c3-001c-4c5b-a201-4531933dde42
```

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

modules/Exec.sol

```
Report for modules/Exec.sol
https://dashboard.mythx.io/#/console/analyses/36c7830e-fe9d-4b34-9ca7-d062d049b671
```

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

modules/Governance.sol

```
Report for modules/Governance.sol
https://dashboard.mythx.io/#/console/analyses/9d33f2eb-17d0-4665-9307-47fbae6519d7
```

| Line | SWC Title | Severity | Short Description |
|---|---|---|---|
| 3 | (SWC-103) Floating Pragma | Low | A floating pragma is set. |
| 10 | (SWC-100) Function Default Visibility | Low | Function visibility is not set. |

AUTOMATED TESTING

modules/Installer.sol

```
Report for modules/Installer.sol
https://dashboard.mythx.io/#/console/analyses/e62513df-780b-4b17-9413-823e173e85b8
```

| Line | SWC Title | Severity | Short Description |
|---|---|---|---|
| 3 | (SWC-103) Floating Pragma | Low | A floating pragma is set. |
| 10 | (SWC-100) Function Default Visibility | Low | Function visibility is not set. |

modules/Liquidation.sol

```
Report for modules/Liquidation.sol
https://dashboard.mythx.io/#/console/analyses/ef3e6afd-45b7-44e0-9700-88c6d04ad735
```

| Line | SWC Title | Severity | Short Description |
|---|---|---|---|
| 3 | (SWC-103) Floating Pragma | Low | A floating pragma is set. |
| 10 | (SWC-100) Function Default Visibility | Low | Function visibility is not set. |

modules/Markets.sol

```
Report for modules/Markets.sol
https://dashboard.mythx.io/#/console/analyses/de4715a1-c866-4c59-9914-79c184c1ec2a
```

| Line | SWC Title | Severity | Short Description |
|---|---|---|---|
| 3 | (SWC-103) Floating Pragma | Low | A floating pragma is set. |
| 10 | (SWC-100) Function Default Visibility | Low | Function visibility is not set. |

AUTOMATED TESTING

modules/RiskManager.sol

```
Report for modules/RiskManager.sol
https://dashboard.mythx.io/#/console/analyses/43343e77-3cc6-4821-a1c6-7a00a6cab69d
```

| Line | SWC Title | Severity | Short Description |
|---|---|---|---|
| 3 | (SWC-103) Floating Pragma | Low | A floating pragma is set. |
| 224 | (SWC-109) Uninitialized Storage Pointer | Medium | Dangerous use of uninitialized storage variables. |

modules/interest-rate-models/IRMDefault.sol

```
Report for modules/interest-rate-models/IRMDefault.sol
https://dashboard.mythx.io/#/console/analyses/ebc69d56-d849-4580-82b7-c3a0cc538549
```

| Line | SWC Title | Severity | Short Description |
|---|---|---|---|
| 1 | (SWC-103) Floating Pragma | Low | A floating pragma is set. |
| 7 | (SWC-100) Function Default Visibility | Low | Function visibility is not set. |

modules/interest-rate-models/IRMFixed.sol

```
Report for modules/interest-rate-models/IRMFixed.sol
https://dashboard.mythx.io/#/console/analyses/3aa5d66d-fb1f-4cf8-ac04-ca0711cfcfef
```

| Line | SWC Title | Severity | Short Description |
|---|---|---|---|
| 3 | (SWC-103) Floating Pragma | Low | A floating pragma is set. |
| 11 | (SWC-100) Function Default Visibility | Low | Function visibility is not set. |

AUTOMATED TESTING

modules/interest-rate-models/IRMLinearRecursive.sol

```
Report for modules/interest-rate-models/IRMLinearRecursive.sol
https://dashboard.mythx.io/#/console/analyses/8a2c0f34-24fd-4382-819d-7c2a6e7355f6
```

| Line | SWC Title | Severity | Short Description |
|---|---|---|---|
| 3 | (SWC-103) Floating Pragma | Low | A floating pragma is set. |
| 10 | (SWC-100) Function Default Visibility | Low | Function visibility is not set. |

modules/interest-rate-models/IRMLinear.sol

```
Report for modules/interest-rate-models/IRMLinear.sol
https://dashboard.mythx.io/#/console/analyses/48a07706-6008-4416-9132-c6b6b8158d04
```

| Line | SWC Title | Severity | Short Description |
|---|---|---|---|
| 3 | (SWC-103) Floating Pragma | Low | A floating pragma is set. |
| 11 | (SWC-100) Function Default Visibility | Low | Function visibility is not set. |

modules/interest-rate-models/IRMZero.sol

```
Report for modules/interest-rate-models/IRMZero.sol
https://dashboard.mythx.io/#/console/analyses/e2afa393-5c6e-4476-b5ce-f9ee8c46a0e2
```

| Line | SWC Title | Severity | Short Description |
|---|---|---|---|
| 3 | (SWC-103) Floating Pragma | Low | A floating pragma is set. |
| 11 | (SWC-100) Function Default Visibility | Low | Function visibility is not set. |

- Issue regarding `floating pragma` has been already mentioned in the above report.
- All `function default visibility` issues refer to the visibility of the constructor which by default is `public` but could be either `public` or `internal`.
- The `uninitialized` issue was manually checked. A struct variable named `AssetStorage storage assetStorage` is declared without assignment. However, it is never used without first setting its pointer to an already declared `storage` variable of type `AssetStorage`.

THANK YOU FOR CHOOSING

// HALBORN