



Audit Report for Euler Protocol - May 7, 2021

Summary

Audit Report prepared by Solidified covering the Euler protocol smart contracts.

Process and Delivery

Three (3) independent Solidified experts performed an unbiased and isolated audit of the code below. The final debrief took place on May 03, 2021, and the results are presented here.

Audited Files

The source code has been supplied in a public source code repository:

<https://github.com/euler-xyz/euler-contracts>

Commit number: **86f81180d4257376f4d4f66fbcbb7c9df64e18c2**

The following files are included in the scope of this audit:

```
contracts
├─ Base.sol
├─ BaseIRM.sol
├─ BaseLogic.sol
├─ BaseModule.sol
├─ Constants.sol
├─ Euler.sol
├─ Events.sol
├─ Interfaces.sol
├─ Proxy.sol
├─ Storage.sol
├─ modules
│   ├── DToken.sol
│   ├── EToken.sol
│   ├── Exec.sol
│   ├── Governance.sol
│   ├── Installer.sol
│   ├── Liquidation.sol
│   ├── Markets.sol
│   ├── RiskManager.sol
│   └─ interest-rate-models
│       ├── IRMFixed.sol
│       ├── IRMLinear.sol
│       └─ IRMLinearRecursive.sol
```



Audit Report for Euler Protocol - May 7, 2021

```
|      └─ IRMZero.sol
|─ vendor
|   └─ RPow.sol
|      └─ TickMath.sol
|─ views
|   └─ EulerGeneralView.sol
```

Intended Behavior

The smart contracts implement a permissionless money market protocol with a focus on reactive interest rates. However, the interest rate model was not fully finalized in the audited version. In addition, the protocol is implemented in a novel modular proxy architecture.

Findings

Smart contract audits are an important step in improving smart contracts' security and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of a smart contract system should exercise caution. To help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**.

Note, that high complexity or lower test coverage does not necessarily equate to a higher risk, although certain bugs are more easily detected in unit testing than a security audit and vice versa.

Criteria	Status	Comment
Code complexity	High	Whilst the code complexity of a project of this size is complex by definition, the auditors feel that the novel modules system and the gas-saving attempts contribute to complexity.
Code readability and clarity	Medium-low	Whilst the code uses clean styling and consistent naming, the unorthodox architecture and heavy use of assembly make the codebase hard to navigate and understand.
Level of Documentation	High	-
Test Coverage	Medium-High	Unit testing and test-suite development were still ongoing during the audit process. A strong commitment to testing is evidenced.



Audit Report for Euler Protocol - May 7, 2021

Test Coverage report:

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	96.17	74.07	98.11	98.21	
Base.sol	95	71.43	100	95.65	87
BaseIRM.sol	100	100	100	100	
BaseLogic.sol	96.37	75.61	100	98.86	399,400
BaseModule.sol	100	50	100	100	
Constants.sol	100	100	100	100	
Euler.sol	92.86	66.67	75	92.86	21
Events.sol	100	100	100	100	
Interfaces.sol	100	100	100	100	
Proxy.sol	100	100	100	100	
Storage.sol	100	100	100	100	
contracts/modules/	88.68	60	89.19	90.81	
DToken.sol	89.87	67.86	80	88.31	148,165,172
EToken.sol	97.22	85	93.33	97.26	34,35
Exec.sol	80	45.83	87.5	82.43	84,90,92,94
Governance.sol	84.62	33.33	75	85.71	20,21
Installer.sol	81.82	50	75	83.33	33,34
Liquidation.sol	87.95	60.71	100	92.77	135,160,161
Markets.sol	95.56	57.14	90.91	97.83	78
RiskManager.sol	87.04	56.82	100	91	109,130,153
contracts/modules/interest-rate-models/	60	100	87.5	60	
IRMFxed.sol	100	100	100	100	
IRMLinear.sol	100	100	100	100	
IRMLinearRecursive.sol	0	100	50	0	15,16
IRMZero.sol	100	100	100	100	
contracts/test/	49.43	33.33	83.33	49.46	
InvariantChecker.sol	0	0	0	0	84,87,90,93
JunkETokenUpgrade.sol	100	100	100	100	
JunkMarketsUpgrade.sol	100	50	100	100	
LiquidationTest.sol	100	100	100	100	
MockUniswapV3Factory.sol	100	50	100	100	
MockUniswapV3Pool.sol	94.12	50	71.43	94.12	40
TestERC20.sol	75	100	75	75	24
contracts/vendor/	100	100	100	100	
RPow.sol	100	100	100	100	
TickMath.sol	100	100	100	100	
contracts/views/	86.05	50	100	90.48	
EulerGeneralView.sol	86.05	50	100	90.48	95,96,98,100
All files	86.45	62.03	91.46	88.06	

Issues Found

Solidified found that the Euler protocol contracts contain no critical issues, no major issues, 4 minor issues, and 4 informational notes.

In addition, a warning on the architecture design has been added.

We recommend issues are amended, while informational notes are up to the team's discretion, as they refer to best practices.

Issue #	Description	Severity	Status
1	Unorthodox Architecture may lead to undetected security risks	Warning	-
2	Base.sol: Dangerous modifier	Minor	Acknowledged
3	Permissionless protocols are at risk from non-standard or malicious tokens	Minor	Acknowledged
4	BaseLogic.sol: Reliance on gas cost for protection from malicious code in external call	Minor	Acknowledged
5	modules/EToken.sol and modules/DToken.sol: CALLER() not future proof	Minor	Acknowledged
6	Unsafe Math Operations	Note	-
7	Sub-account XOR functionality changes the security model of addresses	Note	-
8	Mock of Highly Sophisticated Contracts	Note	-

Warnings

1. Unorthodox Architecture may lead to undetected security risks

The codebase uses a novel proxy-based module system, aimed at reduced gas consumption. This unorthodox architecture relies on excessive use of assembly and a code structure that defies common object-oriented convention. As a result, a lot of code is grouped into a large `BaseLogic` contract which all modules inherit, relying on the compiler to remove unused methods in each module. This code organization makes the codebase harder to navigate and in the view of the three auditors more fragile in terms of security.

Recommendation

Consider using a more conventional architecture at the cost of higher gas consumption.

Critical Issues

No critical issues have been found.

Major Issues

No major issues have been found.

Minor Issues

2. `Base.sol`: Dangerous modifier

The `FREEMEM()` modifier aimed at saving gas can have dangerous side effects if used in the wrong circumstances. Gas savings due to this modifier are marginal in comparison to the risks involved.

Recommendation

Consider not using this modifier.

Team Reply

"We agree that FREEMEM must be used carefully. After we sent you the code we have since added a development mode that fills in the freed memory with garbage to hopefully expose any bugs:

<https://github.com/euler-xyz/euler-contracts/blob/07e25aa8a779bb6bfb3bf64af648dd35586bbba0/contracts/Base.sol#L71-L74>"

3. Permissionless protocols are at risk from non-standard or malicious tokens

The codebase goes to great lengths to mitigate against malicious token implementations. However, it is impossible to protect against all possible attacks by malicious tokens or copycat tokens that impersonate well-known tokens.

Recommendation

One way to protect against this is to use an admin- or governance-controlled token registry. Initially, this could be admin-controlled and then turned over to community governance. At the very least, it is recommended to clearly warn users of this risk at the UI level and/or provide trusted token lists at the UI level.

Team Reply

"Acknowledged. This is one of the key differentiators of our product, however, so we need to accept this risk and secure against malicious tokens as much as possible. The UI will be using one of the token lists (probably coingecko) to protect casual users from copycat tokens."

4. BaseLogic.sol: Reliance on gas cost for protection from malicious code in external call

The `callBalanceOf()` method relies on limiting the amount of gas forwarded to the underlying token to prevent malicious behavior of the token. However, the gas cost for operations has been changed in various recent network updates, rendering this code not future-proof. If gas cost changes in the future, the external call could always fail.

Recommendation

One option to deal with this would be to make the gas parameter governance controlled.

Team Reply

“Acknowledged. We consider it unlikely that gas prices will be raised such that an honest token’s balanceOf() exceeds 20k gas, and in any case governance can elect to upgrade the contracts if necessary.”

5. modules/EToken.sol and modules/DToken.sol: CALLER() not future proof

The `CALLER()` method is declared in both of these modules and is aimed at extracting calldata fed through the proxy that is not specified in the function argument definition.

This is not future-proof, since it depends on the non-guaranteed assumption that calldata will work consistently across future Solidity versions.

Recommendation

One option to deal with this is to lock the pragma to a fixed compiler version.

Informational Notes

6. Unsafe Math Operations

In the EToken contract deposit function, there is math that has been excluded for under/overflow protections through the use of the “unchecked” keyword. While documentation is given if the pool size was to underflow the logic assumptions would break.

Recommendation

Consider using a temp variable rather than modifying the pool storage.

7. Sub-account XOR functionality changes the security model of addresses

The way sub-accounts are created by XOR’ing the last byte of an address with an index changes the security property of addresses if there is ever a weakness in keccak256 that makes collisions more likely. As such future weaknesses in the hash function may have applicable

value in this protocol compared to others that don't effectively truncate 8 bits of security from the hash output.

Recommendation

Consider an alternative approach that doesn't impact the collision resistance of keccak256.

8. Mock of Highly Sophisticated Contracts

Simple mock contracts have been created for the Uniswap V3 pool infrastructure. The Uniswap V3 contract ecosystem is highly complex, and simplified mocks may be incomplete, or not catch edge cases that the full implementation would reject.

Recommendation

Utilize the Uniswap V3 contracts library to deploy the ecosystem locally using the [provided local testing](#).



Audit Report for Euler Protocol - May 7, 2021

Disclaimer

Solidified audit is not a security warranty, investment advice, or an endorsement of Euler or its products. This audit does not provide a security or correctness guarantee of the audited smart contract. Securing smart contracts is a multistep process, therefore running a bug bounty program as a complement to this audit is strongly recommended.

The individual audit reports are anonymized and combined during a debrief process, in order to provide an unbiased delivery and protect the auditors of Solidified platform from legal and financial liability.

Solidified Technologies Inc.