SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY

Magnate Protocol

Audit

Security Assessment
02. June, 2023

For

SolidProof_io

@solidproof_io

Disclaimer

SolidProof.io reports are not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team. SolidProof.io do not cover testing or auditing the integration with external contract or services (such as Unicrypt, Uniswap, PancakeSwap etc’...)

SolidProof.io Audits do not provide any warranty or guarantee regarding the absolute bug- free nature of the technology analyzed, nor do they provide any indication of the technology proprietors. SolidProof Audits should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

SolidProof.io Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology. Blockchain technology and cryptographic assets present a high level of ongoing risk. SolidProof’s position is that each company and individual are responsible for their own due diligence and continuous security. SolidProof in no way claims any guarantee of security or functionality of the technology we agree to analyze.

Version	Date	Description
1.0	25. May 2023 - 31. May 2023	• Layout project • Automated- /Manual-Security Testing • Summary

Note - This Audit report consists of a security analysis of the **Magnate Protocol** smart contracts. This analysis did not include functional testing (or unit testing) of the contract’s logic.

Network
Arbitrum One

Website
https://magnate.finance/

Telegram
https://t.me/magnatefi

Twitter
https://twitter.com/MagnateFi

Description

TBA

Project Engagement

During the 25 of May 2023, **Magnate Protocol Team** engaged Solidproof.io to audit smart contracts that they created. The engagement was technical in nature and focused on identifying security flaws in the design and implementation of the contracts. They provided Solidproof.io with access to their code repository.

Logo

Contract Links

v1.0

https://github.com/MagnateArb/Magnate-protocol/tree/main

Commit: abfeeee

Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 - 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon aspossible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as there were discovered.

Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
   i) Review of the specifications, sources, and instructions provided to SolidProof to make sure we understand the size, scope, and functionality of the smart contract.
   ii) Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   iii) Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to SolidProof describe.

2. Testing and automated analysis that includes the following:
   i) Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   ii) Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.

3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.

4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

Tested Contract Files

This audit covered the following files listed below with a SHA-1 Hash.

A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0

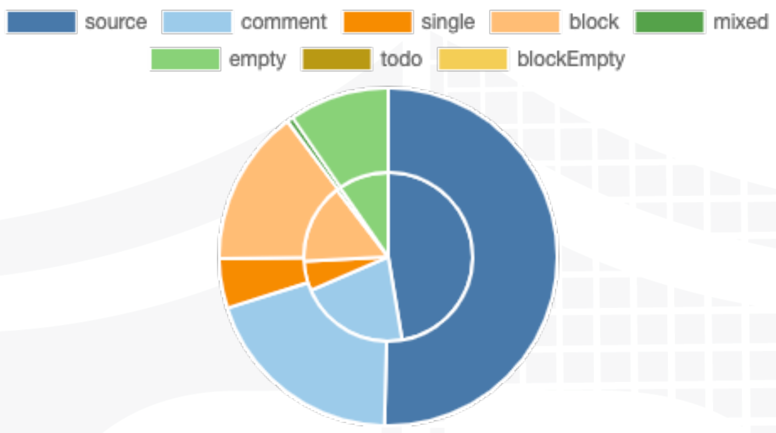
File Name	SHA-1 Hash
contracts/CTokenInterfaces.sol	3913119172e946603cf6574c388f1655e46a7a41
contracts/CEther.sol	1e1467bf5d245b823a33c09d43b68c344ea07106
contracts/CErc20Immutable.sol	f3b9e8970c3968e906858991409fabcbd3f52ca3
contracts/CEtherDelegate.sol	37dfa42d9859abff5a6cf1e6a7143faaf7f78a0c
contracts/CEtherDelegator.sol	bdbaac1a57eee83fa47b0d59112c8f20f58566ce
contracts/ChainlinkPriceOracle.sol	6f6d23c082af3c05f0c3b6c510e2358c29eafd77
contracts/ComptrollerStorage.sol	9d633a875c304f9938c96144b05c8daa5d9f1b3d
contracts/JumpRateModelV2.sol	d90c4ef2bea35d62240b3f7cf5b499d817f06388
contracts/InterestRateModel.sol	a90f903058f47b9ac8ba96a1108811151d3e531c
contracts/EIP20Interface.sol	1557865715c88a7cb11f921a3a2dbc7b5c9dba71
contracts/CarefulMath.sol	76ab5d70e3d2bb236dba285b9940a13f72652bed

contracts/Unitroller.sol	843028710818b9d2aaae104bb19578 7f760ad0bc
contracts/ErrorReporter.sol	b670d0c68fc84406b8b3f49342f8a50 736122420
contracts/SafeMath.sol	5aed165cc366b6fb8ea08e34035602 a3627d4987
contracts/Migrate.sol	e1441ad1f5113b897978ad9a17795a af75270357
contracts/RewardDistributor.sol	ef9ced08d67a6eea353f48454c5f484 6178bdde5
contracts/ ExponentialNoError.sol	72b12974348bbf7055925d8d2816f8a 5f3a9520b
contracts/CErc20Delegate.sol	7bbd85075e55fbe59f9beb904fa2ea6 bdd39481e
contracts/CErc20Delegator.sol	159716bb3637e865d8c49a34a87da5 f718823c27
contracts/CToken.sol	26db009f30970bd9970b531e763f846 68bacf7c8
contracts/PriceOracle.sol	b44597bb3bbc536fd63fe7cec4e41ae 6feaffb26
contracts/Exponential.sol	3dc4aa5312d273cbf408d6536e2761 e7461e8591
contracts/CErc20.sol	481459b2694eeecf0ed277201c14a3f 05b449551
contracts/Comptroller.sol	0f84da9fd101f392009a12c389c0de3 e937f607a
contracts/BasePriceOracle.sol	1ceb6aa5c0b4fa61b9e5fe30c23d21b 20e626d35
contracts/ ComptrollerInterface.sol	f28b2568be489bf1e7eabfa4a2248e0f 156d3a43

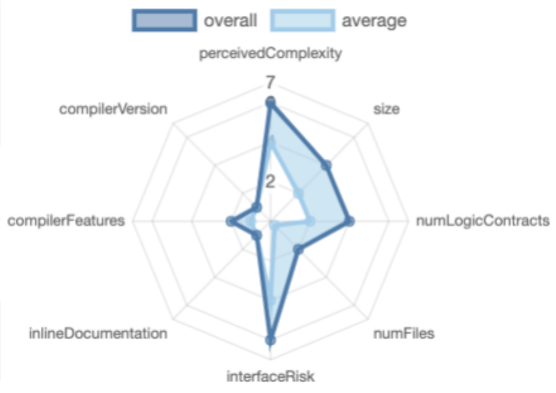
contracts/ EIP20NonStandardInterface.so l	8ed791235ad9c37d79a3211abad9c1 bfb56afdb7

Metrics

Source Lines
v1.0



Risk Level
v1.0



Capabilities

Components

📝Contracts	📚Libraries	🔍Interfaces	🎨Abstract
43	1	5	0

Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

🌐Public	💰Payable
298	23

External	Internal	Private	Pure	View
209	368	0	66	83

StateVariables

Total	🌐Public
93	77

Capabilities

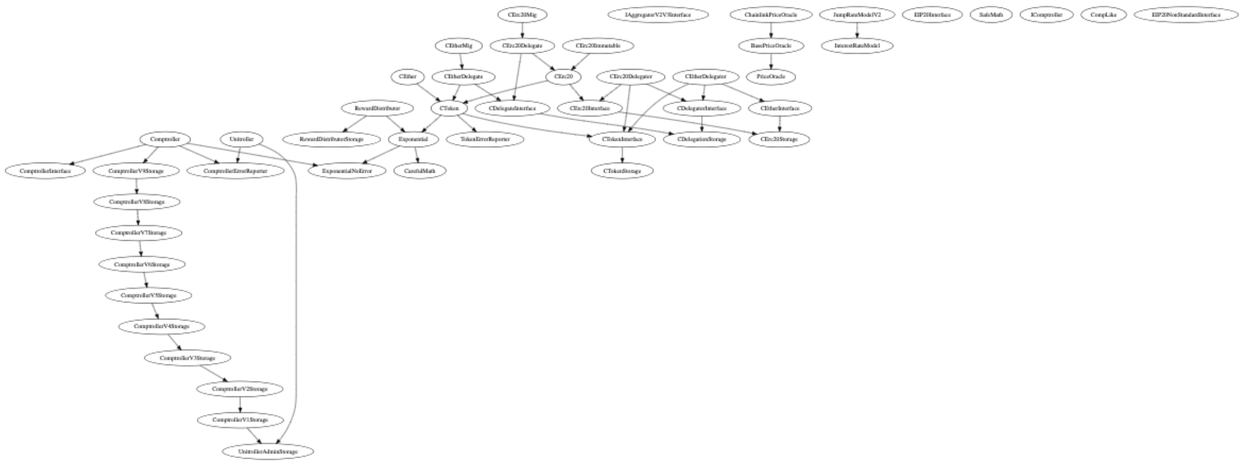
Solidity Versions observed	🧪 Experimental Features	💰 Can Receive Funds	🖥 Uses Assembly	💣 Has Destroyable Contracts
`^0.5.16`	`ABIEncoderV2`	`yes`	`yes` (9 asm blocks)	

📤 Transfers ETH	⚡ Low-Level Calls	👥 DelegateCall	🧮 Uses Hash Functions	🔖 ECRecover	🌀 New/Create/Create2
`yes`	`yes`	`yes`	`yes`		

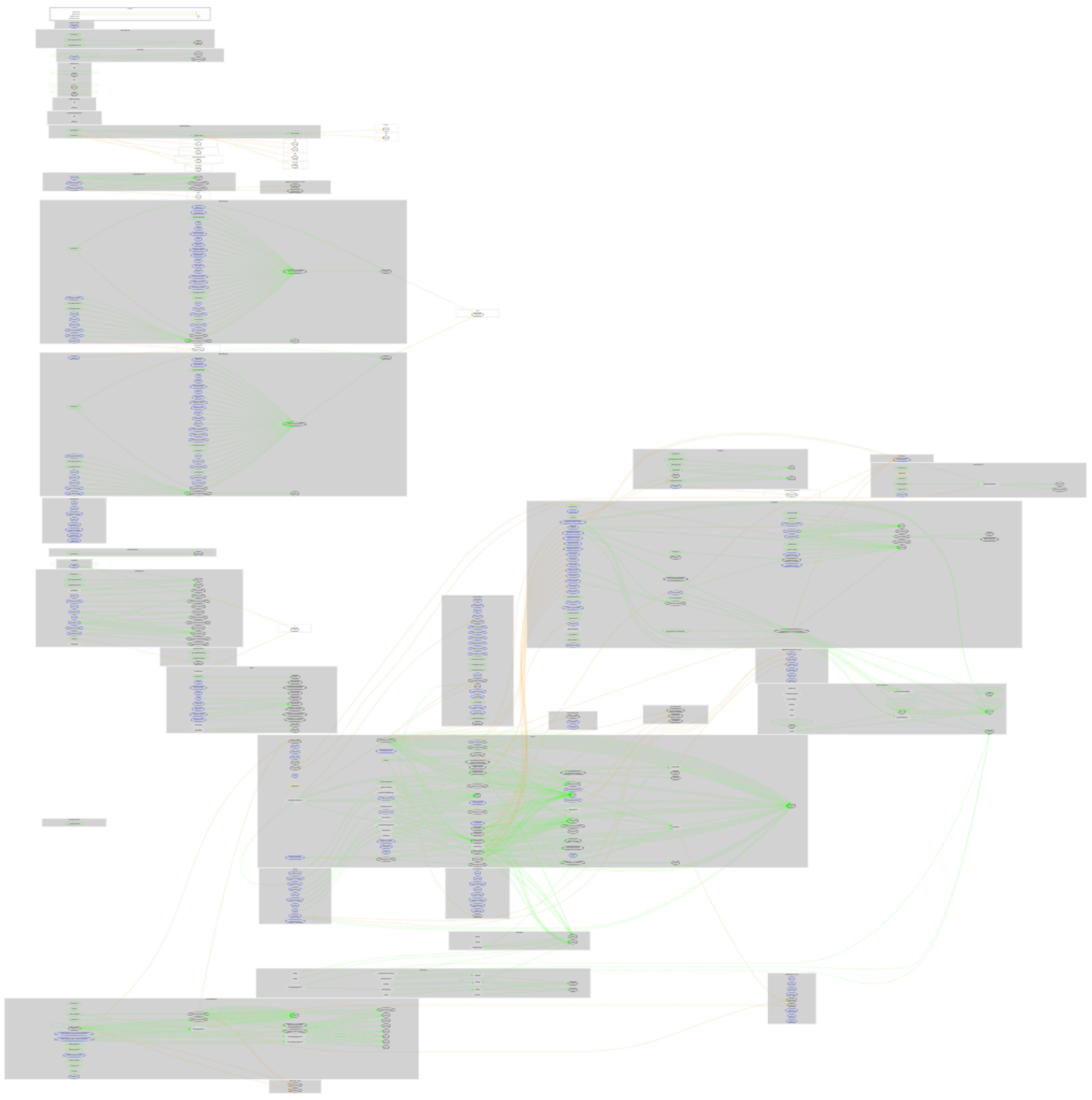
♻️ TryCatch	Σ Unchecked

Inheritance Graph

v1.0



CallGraph
v1.0



Scope of Work/Verify Claims

The above token Team provided us with the files that needs to be tested (Github, Bscscan, Etherscan, files, etc.). The scope of the audit is the main contract (usual the same name as team appended with .sol).

We will verify the following claims:

1. Overall checkup (Smart Contract Security)

Overall checkup (Smart Contract Security)

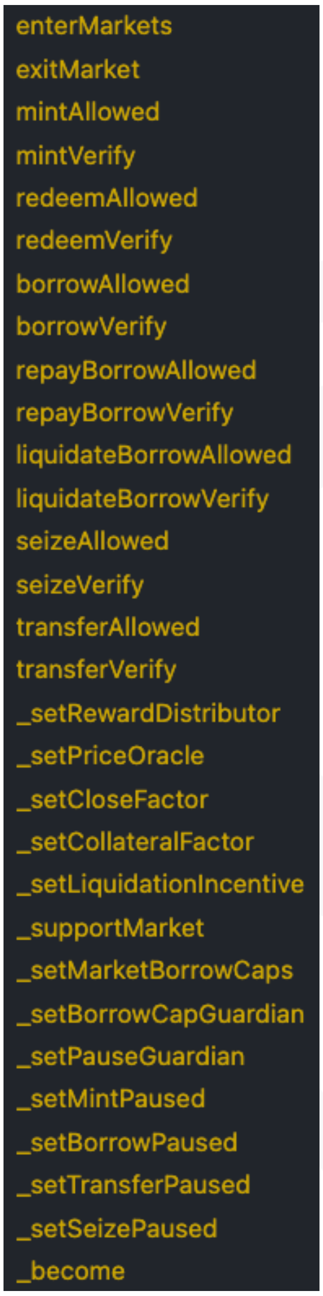
Tested	Verified
✓	✓

Legend

Attribute	Symbol
Verified / Checked	✓
Partly Verified	⚑
Unverified / Not checked	✗
Not available	–

Modifiers and public functions
v1.0

Comptroller



```
enterMarkets
exitMarket
mintAllowed
mintVerify
redeemAllowed
redeemVerify
borrowAllowed
borrowVerify
repayBorrowAllowed
repayBorrowVerify
liquidateBorrowAllowed
liquidateBorrowVerify
seizeAllowed
seizeVerify
transferAllowed
transferVerify
_setRewardDistributor
_setPriceOracle
_setCloseFactor
_setCollateralFactor
_setLiquidationIncentive
_supportMarket
_setMarketBorrowCaps
_setBorrowCapGuardian
_setPauseGuardian
_setMintPaused
_setBorrowPaused
_setTransferPaused
_setSeizePaused
_become
```

Note:

❖ General fork from Compound Protocol

- Contracts inside are the same as the pancake-smart-contracts directory

- https://github.com/compound-finance/compound-protocol/tree/master/contracts
- Differences between Magnate Protocol and Compound Protocol contracts are the following:
  - The comptroller contract has the borrow pause functionality implemented in the “redeemAllowed” function. The admin can add a new reward distributor address, and the existing distribution functionality has been completely removed. Please note that the reward contract was not included in the audit scope. The admin of the Magnate contract
  - New error-throwing mechanisms have been added to the Ctoken contract

Ownership/Admin Privileges

❖ *Comptroller.sol* -

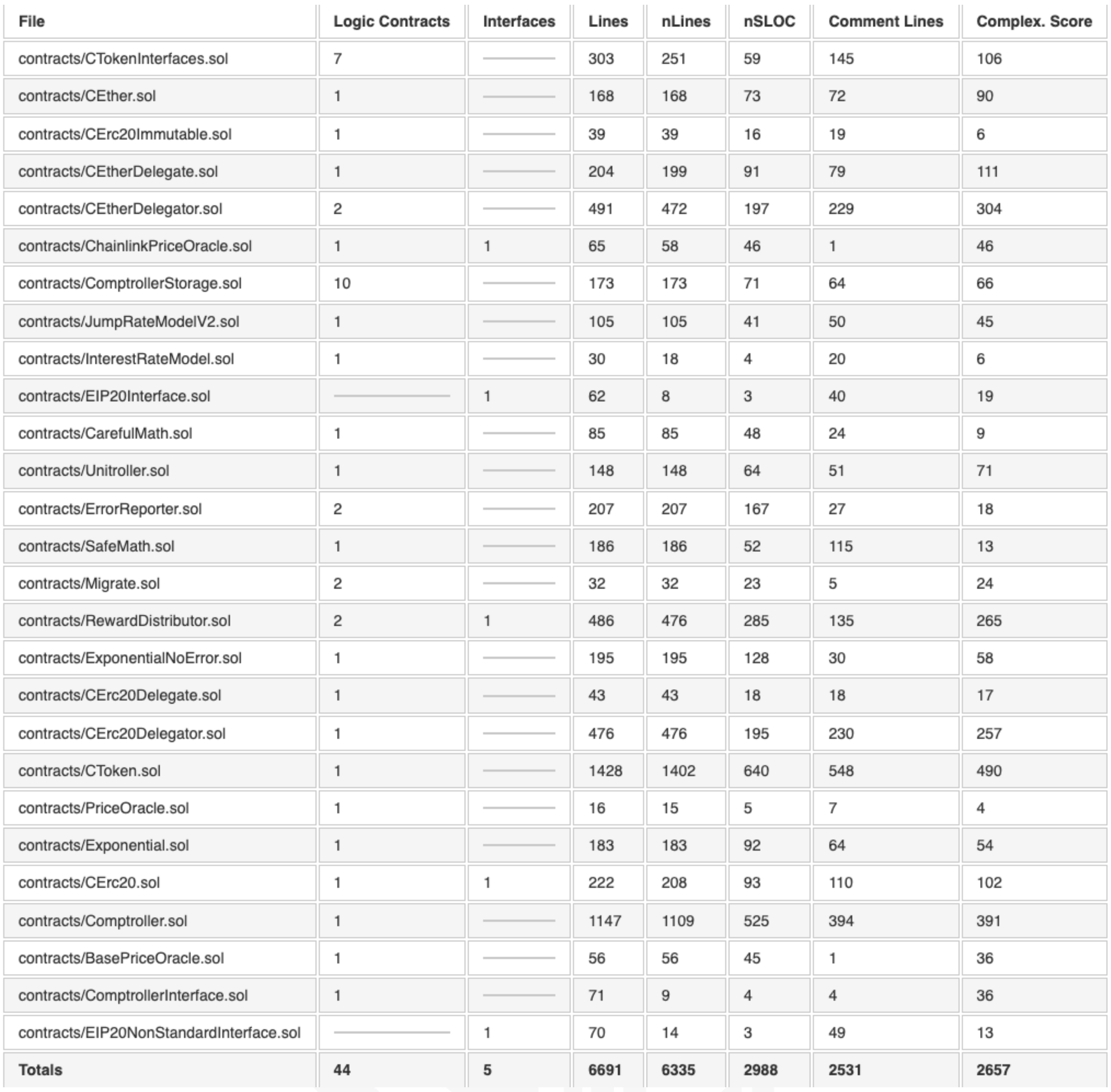
- Pause/Unpause minting, borrowing, transfer, and seize
- Set reward distributor, price oracle address
- Set close factor, collateral Factor, liquidation incentive, and market borrow caps.
- Set borrow cap guardian address.

❖ *CToken.sol* -

- Set admin address
- Set comptroller address
- Set reserve factor

Please check if an OnlyOwner or similar restrictive modifier has been forgotten.

Source Units in Scope
v1.0



File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score
contracts/CTokenInterfaces.sol	7	—	303	251	59	145	106
contracts/CEther.sol	1	—	168	168	73	72	90
contracts/CErc20Immutable.sol	1	—	39	39	16	19	6
contracts/CEtherDelegate.sol	1	—	204	199	91	79	111
contracts/CEtherDelegator.sol	2	—	491	472	197	229	304
contracts/ChainlinkPriceOracle.sol	1	1	65	58	46	1	46
contracts/ComptrollerStorage.sol	10	—	173	173	71	64	66
contracts/JumpRateModelV2.sol	1	—	105	105	41	50	45
contracts/InterestRateModel.sol	1	—	30	18	4	20	6
contracts/EIP20Interface.sol	—	1	62	8	3	40	19
contracts/CarefulMath.sol	1	—	85	85	48	24	9
contracts/Unitroller.sol	1	—	148	148	64	51	71
contracts/ErrorReporter.sol	2	—	207	207	167	27	18
contracts/SafeMath.sol	1	—	186	186	52	115	13
contracts/Migrate.sol	2	—	32	32	23	5	24
contracts/RewardDistributor.sol	2	1	486	476	285	135	265
contracts/ExponentialNoError.sol	1	—	195	195	128	30	58
contracts/CErc20Delegate.sol	1	—	43	43	18	18	17
contracts/CErc20Delegator.sol	1	—	476	476	195	230	257
contracts/CToken.sol	1	—	1428	1402	640	548	490
contracts/PriceOracle.sol	1	—	16	15	5	7	4
contracts/Exponential.sol	1	—	183	183	92	64	54
contracts/CErc20.sol	1	1	222	208	93	110	102
contracts/Comptroller.sol	1	—	1147	1109	525	394	391
contracts/BasePriceOracle.sol	1	—	56	56	45	1	36
contracts/ComptrollerInterface.sol	1	—	71	9	4	4	36
contracts/EIP20NonStandardInterface.sol	—	1	70	14	3	49	13
Totals	**44**	**5**	**6691**	**6335**	**2988**	**2531**	**2657**

Legend

Attribute	Description
Lines	total lines of the source unit
nLines	normalised lines of the source unit (e.g. normalises functions spanning multiple lines)
nSLOC	normalised source lines of code (only source-code lines; no comments, no blank lines)
Comment Lines	lines containing single or block comments

Complexity Score	a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

Audit Results

Critical issues

No critical issues

High issues

No high issues

Medium issues

No medium issues

Low issues

Issue	File	Type	Line	Description
#1	All	Multiple pragma is set	—	Some of the contracts contain different pragma versions which is not recommended for deployment. We recommend to have the same pragma in all contracts and also to update the old pragma versions to the new ones.
#2	ProtocolEarnings.sol	Missing Events Arithmetic	28-28, 57	Emit an event for critical parameter changes
#3	All	Old Compiler Version	—	The contracts use a very old compiler version which is not recommended for deployment as it is susceptible to known vulnerabilities. We strongly recommend the team to use the latest compiler version in order to avoin over/underflow errors automatically.

Informational issues

Issue	File	Type	Line	Description

#1	All	Contract doesn't import npm packages from source (like OpenZeppelin etc.)	—	We recommend importing all packages from npm directly without flattening the contract. Functions could be modified or can be susceptible to vulnerabilities

Audit Comments

We recommend you use the particular form of comments (NatSpec Format, Follow the link for more information https://docs.soliditylang.org/en/latest/natspec-format.html) for your contracts to provide rich documentation for functions, return variables and more. This helps investors to make clear what those variables, functions etc. do.

02. June 2023:

- This project consists of the following forks
  - Compound Protocol
- Unit tests with at least 95% code coverage and a Whitepaper were not provided to SolidProof, so we cannot ensure the complete functional correctness of the code's logic.
- We recommend Magnate Protocol team conduct unit and fuzz tests thoroughly to rule out the possibilities of unwanted logical and calculation errors.
- Read the whole report and modifiers section for more information
- The low issues that remain unfixed in the Compound Protocol codebase still exist in the forked code.
- We recommend using a multi-sig wallet for the owner's address to prevent any risk of the loss of a private key
- Do your own research here before investing.

SWC Attacks

ID	Title	Relationships	Status
SWC-136	Unencrypted Private Data On-Chain	CWE-767: Access to Critical Private Variable via Public Method	PASSED
SWC-135	Code With No Effects	CWE-1164: Irrelevant Code	PASSED
SWC-134	Message call with hardcoded gas amount	CWE-655: Improper Initialization	PASSED
SWC-133	Hash Collisions With Multiple Variable Length Arguments	CWE-294: Authentication Bypass by Capture-replay	PASSED
SWC-132	Unexpected Ether balance	CWE-667: Improper Locking	PASSED
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	PASSED
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	PASSED
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	PASSED
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	PASSED

SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	**PASSED**
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	**PASSED**
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	**PASSED**
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	**PASSED**
SWC-122	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	**PASSED**
SWC-121	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	**PASSED**
SWC-120	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	**PASSED**
SWC-119	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	**PASSED**
SWC-118	Incorrect Constructor Name	CWE-665: Improper Initialization	**PASSED**
SWC-117	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	**PASSED**

SWC-116	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SWC-115	Authorization through tx.origin	CWE-477: Use of Obsolete Function	PASSED
SWC-114	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	PASSED
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	PASSED
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	PASSED
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	PASSED
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	PASSED
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	PASSED
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	PASSED

SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	**PASSED**
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	**PASSED**
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	**PASSED**
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	**NOT PASSED**
SWC-101	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	**PASSED**
SWC-100	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	**PASSED**

SolidProof_io

@solidproof_io

Solid Proofed

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY