SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Paribus
Date: March 13, 2023

Document

Name	Smart Contract Code Review and Security Analysis Report for Paribus
Approved By	Marcin Ugarenko | Lead Solidity SC Auditor at Hacken OU
Type	Lending Platform
Platform	EVM
Language	Solidity
Methodology	Link
Website	https://paribus.io/
Changelog	29.01.2023 - Initial Review 22.02.2023 - Second Review 13.03.2023 - Third Review

Table of contents

Introduction 4
Scope 4
Severity Definitions 12
Executive Summary 13
Checked Items 14
System Overview 17
Findings 20
Critical 20
High 20
H01. Non-Finalized Code 20
H02. Non-Finalized Code 20
H03. Requirements Violation 20
H04. Undocumented Behavior 21
Medium 21
M01. Best Practice Violation - Usage of Assert 21
M02. Unscalable Functionality - Shadowing State Variable 22
M03. Contradiction - Missing Validation 22
M04. Unscalable Functionality - Shadowing State Variable 22
Low 23
L01. Floating Pragma 23
L02. Style Guide Violation 24
L03. Unused Function Arguments 25
L04. Best Practices - Modifiers 25
L05. Missing Zero Address Validation 25
L06. Checks-Effects-Interactions Violation 25
L07. Missing Events 26
L08. Functions That Can Be Declared External 26
L09. Boolean Equality 26
L10. Unindexed Events 27
L11. No Messages in Require Conditions 27
L12. Outdated Compiler Version 27
Disclaimers 30

Introduction

Hacken OÜ (Consultant) was contracted by Paribus (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contracts.

Scope

The scope of the project is review and security analysis of smart contracts in the repository:

Initial review scope

Repository	https://github.com/Paribus/paribus-protocol-contracts
Commit	90dcd5f94d746a128dae43de532db3298ac78fc0
Whitepaper	https://paribus.io/documents/PARIBUS-Litepaper-V1.0.pdf
Functional Requirements	https://github.com/Paribus/paribus-protocol-contracts/blob/mainnet-mvp/README.md
Technical Requirements	https://github.com/Paribus/paribus-protocol-contracts/blob/mainnet-mvp/README.md
Contracts	File: ./contracts/Comptroller/ComptrollerCommonImpl.sol SHA3: ee506ddd80149530bef4e157f1a429897fa950837ac150f7d7cde861a43e39e3 File: ./contracts/Comptroller/ComptrollerInterface.sol SHA3: 871fb80caf2f743d7289cd44784c42649c251aec7900b4799fcc3bd370b7ff88 File: ./contracts/Comptroller/ComptrollerPart1.sol SHA3: f7e0ee483131d57692bc2eb9380b2bded41ebca9a46f739393d4d1565c37a2a5 File: ./contracts/Comptroller/ComptrollerPart2.sol SHA3: 86f02a906b0ea0d2d89c47540488966ea048043970df39d8a47cd586db18a607 File: ./contracts/Comptroller/ComptrollerStorage.sol SHA3: 1ce88f8789b2ff09835f84c1980350f21addc1c49a6368e713675ac667bdbd08 File: ./contracts/Comptroller/Unitroller.sol SHA3: 05e3b9305b250217131908f6305e7267a322980809bcbadecc30e65a09ef437f File: ./contracts/ErrorReporter.sol SHA3: c3e19bc83d0ffef1c0338798913fcc0b3ee74c1689433861c22c5755fcc1ea4d File: ./contracts/InterestRateModels/BaseJumpRateModelV2.sol SHA3: 307d993383a78ece804790c87427cad17a537afad436a005deb5d291b9a38723 File: ./contracts/InterestRateModels/DAIInterestRateModelV3.sol SHA3: 60a4d128d7fd9a383da362b84a81887624950230430b688b3c97008d855e0742 File: ./contracts/InterestRateModels/InterestRateModelInterface.sol SHA3: a0c5c4c7f1f3d3cdc09d4d8c765896109817cc5e4c4860328d927aabd932aee4 File: ./contracts/InterestRateModels/JumpRateModel.sol SHA3: 4d3e807589b604de379e786f4c00a83b66befe8991b7f803a9e50312605be0bd File: ./contracts/InterestRateModels/JumpRateModelV2.sol SHA3: bd9ac6b96bbe9a11c567e532deeb1b3754dabd666b8c555e326e050f0072d389

	File: ./contracts/InterestRateModels/LegacyInterestRateModel.sol
	SHA3:
	9c8c9ba78a900c14c8ad6d95a65352d367b3e3ceb31e8f05c4ae1e15ad530b95
	File: ./contracts/InterestRateModels/LegacyJumpRateModelV2.sol
	SHA3:
	3ca2ee9712be05a0e5d89b872b912f4493fe9609a88887ff91e4ff47eda78969
	File: ./contracts/InterestRateModels/WhitePaperInterestRateModel.sol
	SHA3:
	cc4d796b1edaaf50f25e492669924245edf9c2f456fa25a9ab80432096b0b57a
	File: ./contracts/Interfaces/AaveInterfaces.sol
	SHA3:
	a964f41a063d1e62ab0b1566fea95bda8c2a6c4a339e65d08078242974b7a4fe
	File: ./contracts/Interfaces/Api3Interfaces.sol
	SHA3:
	406999c7910f4b29ee7dd8a5249d3ac36e039ccc3149adcc55fd576fa7d7d275
	File: ./contracts/Interfaces/EIP20Interface.sol
	SHA3:
	9f04d6854fb5ac2b37fadb8eed23264034cd2515f5fc0efbf4a942ec474e8506
	File: ./contracts/Interfaces/EIP20NonStandardInterface.sol
	SHA3:
	83b3090f8985051b09b9b8bf131386957ae23819e0da190f503f7e69a250f654
	File: ./contracts/Interfaces/UniswapV2Interfaces.sol
	SHA3:
	f50858933c0c94716fa15da302a8cf46419d457b20a162715984a25ee76419af
	File: ./contracts/Interfaces/UniswapV3Interfaces.sol
	SHA3:
	2353f4bbf1768c6f305a29b030a37d461285e2171883edd42c1a85cec28fa92a
	File: ./contracts/Liquidator.sol
	SHA3:
	ab39b4bfb5b6036e91f49e0f9f7a07484af5ed30b142dc97ace69a8a49f393ee
	File: ./contracts/Maximillion.sol
	SHA3:
	ef6bd55adb315b643a702617211e57175280d7f6722f2c3a56429221b2e6bd21
	File: ./contracts/PBXToken.sol
	SHA3:
	adbd140dbcde8ff22cab439e5954fcafe64da7fdc8115be0d6efdef2fdb1fb57
	File: ./contracts/PriceOracle/ArbitrumPriceOracle.sol
	SHA3:
	edfc09dab65528cf52ea3a0fe9034b0e011eef64bfbc12758a003351e38341e7
	File: ./contracts/PriceOracle/GoerliPriceOracle.sol
	SHA3:
	454287daa907961ac5f44825ce26e3ebb2e9cb96d9aef4d0ced84edb1a0fda44
	File: ./contracts/PriceOracle/Impl/Api3PriceOracle.sol
	SHA3:
	c78e062e17a5ed053775bbea77d049097d1e051bee0daceea3f8ed5fcfbbfca2
	File: ./contracts/PriceOracle/Impl/ChainlinkPriceOracle.sol
	SHA3:
	2392de1d2f14d3ac478e9b5f3e61e40fc7f4012b9ccfec349abd6f9fdc5e4212
	File: ./contracts/PriceOracle/Impl/PriceOracleCommonImpl.sol
	SHA3:
	3aa8347385d87abc746525b17f4d256cd81ae32d781a6c995b66bf3db50952ef
	File: ./contracts/PriceOracle/Impl/StablecoinsPriceOracle.sol
	SHA3:
	a0b4a88f7c483fc34c29b389ee7640d576d2510989e484ea2d2bf701b06d8230
	File: ./contracts/PriceOracle/PolygonPriceOracle.sol
	SHA3:
	ed5981be7e1139f474fddb35fbd66e4e9ef8134377907e01813013e3669c9486
	File: ./contracts/PriceOracle/PriceOracleInterface.sol
	SHA3:
	6006a75b29cfd3b5a492f664d8834fdbe59b11ff77f5fad3dc6a133f6880ab9e
	File: ./contracts/PriceOracle/RinkebyPriceOracle.sol
	SHA3:
	6cca6d53a8ec2a9f614443d895d966032c45d1c11f78cc9170203c46cee72345
	File: ./contracts/PriceOracle/SimplePriceOracle.sol

	SHA3: 4452ca6695e37c3fd63ea7367fa1952aedf2f2e66b301feb06d97e8b7387721d File: ./contracts/PToken/PErc20/PErc20.sol SHA3: 0ebcb28b94e9790a827d2a038988d277aa489c95978b66fb120e20a062b00859 File: ./contracts/PToken/PErc20/PErc20Delegate.sol SHA3: 6bb6353dccf5ba0cf9f5e3aaf4674c453776de5872689f2efa8ae5ed4b4d1d8e File: ./contracts/PToken/PErc20/PErc20Delegator.sol SHA3: 19451ba74cc36a6d737957fabf5c3465c7c4e8a25d5a854ef9193d23d860c489 File: ./contracts/PToken/PErc20/PErc20Immutable.sol SHA3: 40dbcf11b985b4c6eca3c46b18b9a56bcaaba45e898fda9fd5d28c38f92aae63 File: ./contracts/PToken/PEther/PEther.sol SHA3: 21db7a023fac76c5f4c9b84d6407aa3fbf67b80ed69203a40b0ade76d0acc5f5 File: ./contracts/PToken/PEther/PEtherDelegate.sol SHA3: 9f71d7b8da92772404e86c9a09ac072cc489a4266a432ce610951001e2e81ccc File: ./contracts/PToken/PEther/PEtherDelegator.sol SHA3: 1c5d8db27fca7d2341d24f5fd3723877cd1f6e64da723f957aba2a88857dfbc6 File: ./contracts/PToken/PEther/PEtherImmutable.sol SHA3: 7768db57bee975e131d68f5478fa2350e0cc2ba777963027dceeb7e31039f83e File: ./contracts/PToken/PToken.sol SHA3: 1e85435e89e0b1f6f7035ced367f30e4cf551dfcd6f508689d50c61aade8c73d File: ./contracts/PToken/PTokenInterfaces.sol SHA3: 9d25c76be4fece75523aad709190538037d4f6b3eb9045bc0b01b8638c174211 File: ./contracts/Utils/ExponentialNoError.sol SHA3: aa5b4a810160e383d6ca2de28e4bfbbce4be2166b6933a5ad7337898609720d2 File: ./contracts/Utils/Ownable.sol SHA3: 6f2afa88571aa431358939eb6487bdb2d04ada8659a09a956e1b9c6e106a8f32 File: ./contracts/Utils/SafeMath.sol SHA3: 60e6dc8f43c9ca59cf273c4691d1d1d7aac5ed724dfee53b0238f84edbf8e14f File: ./contracts/Utils/Timelock.sol SHA3: 6026e1918b8be538d3169280a6b475f59ea240ca0cb1b743d3f05679f9ce9c1d

Second review scope

Repository	https://github.com/Paribus/paribus-protocol-contracts
Commit	d441c2edee027d7a4b68d71b058dffe0722907d5
Contracts	File: ./contracts/Comptroller/ComptrollerCommonImpl.sol SHA3: 673a94efc6ee033486bbacab97006165e99c18fd8bdd91604df0c6e95871bb0f File: ./contracts/Comptroller/ComptrollerInterfaces.sol SHA3: 1bde9b6ce6f69b1be5e07cccc57bda2c4b230d5f4f5d67331d9b54d25b69043a File: ./contracts/Comptroller/ComptrollerPart1.sol SHA3: 98e496e0bdfcc9049dba3554bffc4386930fb2549860ad7b6e1c5d045ca12fd0 File: ./contracts/Comptroller/ComptrollerPart2.sol

	SHA3: e3a5baf62ac0da69a621a82228afa3e8b8bcc34f9e8502f9fda03f6c1f9dfeca File: ./contracts/Comptroller/ComptrollerStorage.sol SHA3: 138374c619cc5a1837117dbac5cfad86db6088bf9e10a966eac4bc240fca4d2f File: ./contracts/Comptroller/Unitroller.sol SHA3: 72324f38e42a2bf88e03650842236c454f671c36842820fb369c5dba721c93d3 File: ./contracts/ErrorReporter.sol SHA3: 88d27f7211ca0bd538e5dac5037a0cbfcd3ffa13e8183ad3462f7fbc737264a6 File: ./contracts/InterestRateModels/BaseJumpRateModelV2.sol SHA3: 6ccfe5c42b5fb6417c2d3593af538117dcf7e22257512b189a98cad40a1a8f68 File: ./contracts/InterestRateModels/InterestRateModelInterface.sol SHA3: 3190eaf2f32012c354d27fb71bea099dc6b37421495f2d4fb714f077e469868f File: ./contracts/InterestRateModels/JumpRateModelV2.sol SHA3: 8c9f0b8107dc77720e91f1e57bdae7df6180c37a28670f8bee9e4ea136788a50 File: ./contracts/Interfaces/Api3Interfaces.sol SHA3: 905f62b3226945893052fbe07b52ae60f56fc6232b43c8645e45e50c93697165 File: ./contracts/Interfaces/EIP20Interface.sol SHA3: 86d8ba61025b1c77e7426d89f01ec149696e2cdc6063b57624847e066d66ca64 File: ./contracts/Interfaces/EIP20NonStandardInterface.sol SHA3: 16f1ec9e2db103cbbafd2e32e3324a506e9e22f4dc2333575ccf558701f9b3d5 File: ./contracts/Maximillion.sol SHA3: 5580ed7d4e6b9e1460b19f43a08d29d4069b073a81ca294ca5bc973354b86fd1 File: ./contracts/PBXToken.sol SHA3: b55ce22e5200ab67f9a400cd04cad4ca589312cca2ceab2235ed7945bc1cca88 File: ./contracts/PriceOracle/ArbitrumPriceOracle.sol SHA3: 369d5f1895622ac6e94612d072da7138d1bc645c1220818b2564027927ee82e2 File: ./contracts/PriceOracle/Impl/Api3PriceOracle.sol SHA3: 2a5aebea598752bd3ca6ab45d3a27c57bcdac4a1b72d5693de3a803ecd5eff80 File: ./contracts/PriceOracle/Impl/ChainlinkPriceOracle.sol SHA3: 3c790722c91c7d3c61223f1fa54101226930242db38729b95002fb7e1e44b023 File: ./contracts/PriceOracle/Impl/PriceOracleCommonImpl.sol SHA3: 2079236cb74f585ba3dcf85caf48d3afb8642aa961465e58f85e59c00ea617c0 File: ./contracts/PriceOracle/Impl/StablecoinsPriceOracle.sol

```
SHA3:
7716d16743a523b569f3c732c1c0503bf278b5166d91a97d8c2bc9da9ceecfba

File: ./contracts/PriceOracle/PriceOracleInterface.sol
SHA3:
e97659d5bd46c398f0eadc1a11cbc16e0e1177717e6932dc9268160c5e1ea8fe

File: ./contracts/PriceOracle/SimplePriceOracle.sol
SHA3:
430e03948b62629489dcf409a92e3ef1784b498fc23289d82f19af51c171fcf2

File: ./contracts/PToken/PErc20/PErc20.sol
SHA3:
63723333589d42424a94d38dd5757fa77905de1fdc3a7637e15a60db5b11a467

File: ./contracts/PToken/PErc20/PErc20Delegate.sol
SHA3:
1fc249acbb9aec4066de7f158f856525d6455080e9ce9362e17d8cdb8bc13307

File: ./contracts/PToken/PErc20/PErc20Delegator.sol
SHA3:
2f588483c9b29479c75ef14fe1dd48553ae21cc83877f227bc85f8365986a39c

File: ./contracts/PToken/PErc20/PErc20Immutable.sol
SHA3:
ef4f6e77837cd56b6700967f841acbb8bd4f94b33c603e87b8eb24a76eb08549

File: ./contracts/PToken/PEther/PEther.sol
SHA3:
7d6a4be7cc6f180d0b3f1a861d9d10d4d5665c68aa8a9977899e8df5558979c6

File: ./contracts/PToken/PEther/PEtherDelegate.sol
SHA3:
ebb9d0077cc87d32a7f20f2cc3ab5af4758091513c734e9c10968e1ca1e4014f

File: ./contracts/PToken/PEther/PEtherDelegator.sol
SHA3:
f2c21403f4a99ed8f6dd81054dd1f8773d3588035caaa10fec7abc9a2ffc7d1c

File: ./contracts/PToken/PEther/PEtherImmutable.sol
SHA3:
ae8d1900790a602f6088c9cfbc6ec01d906771c96a3a0ff6eb48bc6bc08fb27d

File: ./contracts/PToken/PToken.sol
SHA3:
15f1f42ebe255c7b8687071cbc9a9b8a8e506f24b697c005287f4eee9456a56c

File: ./contracts/PToken/PTokenInterfaces.sol
SHA3:
0d18171c2ca725e8c1c263219eae99f731e858731088dd340fa542aad9e1a8d2

File: ./contracts/Utils/ExponentialNoError.sol
SHA3:
19ef154d786b2e85ef0a2449a1a16be4b812cc638845085baf02d768364eb088

File: ./contracts/Utils/Ownable.sol
SHA3:
8ba8cb523b8b5d9d428ee7f8f90924d43e429e0823d1d58bfe328b44543e4758

File: ./contracts/Utils/SafeMath.sol
SHA3:
f3672bc098d0e16e1ca73b5bd3a47fcafb7ea08817ce5e2dd2c38327d5ce9d98
```

Third review scope

Repository	https://github.com/Paribus/paribus-protocol-contracts
Commit	294e42958a502f4a3600629035ad055728df6b5a
Contracts	File: ./contracts/Comptroller/ComptrollerCommonImpl.sol SHA3: 931910015c368807afcdf7bf678164f53c696f192df41864a440ac63772f2c0b File: ./contracts/Comptroller/ComptrollerInterfaces.sol SHA3: db5040835b5c54d50598d346fd718974becdffda002fcdb0259e7e078f700e10 File: ./contracts/Comptroller/ComptrollerPart1.sol SHA3: 78a1408a5e1d4a6dfc68225057d743cbdae583092efd93f366572e8a94d8e79b File: ./contracts/Comptroller/ComptrollerPart2.sol SHA3: e3625428d28045431c217236c035f39bbf494e16e18ff5ac5b81e259faf2a8f3 File: ./contracts/Comptroller/ComptrollerStorage.sol SHA3: 3e9c4268760c432c02f099bb73f252d2bed3da7bac2fe9e350d2a1959d65082c File: ./contracts/Comptroller/Unitroller.sol SHA3: c812a8c33534e2a13df09f51ed252aadaf3d78b943ef007f34cd684900961fe2 File: ./contracts/ErrorReporter.sol SHA3: 88d27f7211ca0bd538e5dac5037a0cbfcd3ffa13e8183ad3462f7fbc737264a6 File: ./contracts/InterestRateModels/BaseJumpRateModelV2.sol SHA3: eb8d74bb1c57498e048cf27fc527dd7a7c43c50b4d472f4eaed6049f49ed754d File: ./contracts/InterestRateModels/InterestRateModelInterface.sol SHA3: 3190eaf2f32012c354d27fb71bea099dc6b37421495f2d4fb714f077e469868f File: ./contracts/InterestRateModels/JumpRateModelV2.sol SHA3: 8c9f0b8107dc77720e91f1e57bdae7df6180c37a28670f8bee9e4ea136788a50 File: ./contracts/Interfaces/Api3Interfaces.sol SHA3: 905f62b3226945893052fbe07b52ae60f56fc6232b43c8645e45e50c93697165 File: ./contracts/Interfaces/EIP20Interface.sol SHA3: 86d8ba61025b1c77e7426d89f01ec149696e2cdc6063b57624847e066d66ca64 File: ./contracts/Interfaces/EIP20NonStandardInterface.sol SHA3: 16f1ec9e2db103cbbafd2e32e3324a506e9e22f4dc2333575ccf558701f9b3d5 File: ./contracts/Maximillion.sol SHA3: 5580ed7d4e6b9e1460b19f43a08d29d4069b073a81ca294ca5bc973354b86fd1 File: ./contracts/PBXToken.sol SHA3: b55ce22e5200ab67f9a400cd04cad4ca589312cca2ceab2235ed7945bc1cca88

	File: ./contracts/PriceOracle/ArbitrumPriceOracle.sol SHA3: 369d5f1895622ac6e94612d072da7138d1bc645c1220818b2564027927ee82e2 File: ./contracts/PriceOracle/Impl/Api3PriceOracle.sol SHA3: f520b3c858130db5c65d739a73267901683de004a5a11336ebc5df97039c6ea5 File: ./contracts/PriceOracle/Impl/ChainlinkPriceOracle.sol SHA3: d47dea7d7538d6df6e163607cd4492c53a170c37e4ea81303710bc32e629376c File: ./contracts/PriceOracle/Impl/PriceOracleCommonImpl.sol SHA3: dfaa326ce0888776972b373079192004c87765c7261080f15c1513db14ff0324 File: ./contracts/PriceOracle/Impl/StablecoinsPriceOracle.sol SHA3: ad659cd9f7882d25d65e2731b6ca7eb32946c4bc16e20bd89d2a02dce8177b90 File: ./contracts/PriceOracle/PriceOracleInterface.sol SHA3: d403fa25a2b3d68130e219cb3dfa71a86b5eec0f33008707457288aba8552874 File: ./contracts/PriceOracle/SimplePriceOracle.sol SHA3: b47032641c3a6e94376e36668662f9dc3834d0042a891346fd6d283360d88546 File: ./contracts/PToken/PErc20/PErc20.sol SHA3: 63723333589d42424a94d38dd5757fa77905de1fdc3a7637e15a60db5b11a467 File: ./contracts/PToken/PErc20/PErc20Delegate.sol SHA3: 56ec0612fe0da193200a7af8aad789431f06e361d3c686d151db85470f3f9424 File: ./contracts/PToken/PErc20/PErc20Delegator.sol SHA3: ee42dc1aa48b0e0e2d42b8dbdebf6caf3349667a2e0adf4bf4ad0036d9b4532e File: ./contracts/PToken/PErc20/PErc20Immutable.sol SHA3: ef4f6e77837cd56b6700967f841acbb8bd4f94b33c603e87b8eb24a76eb08549 File: ./contracts/PToken/PEther/PEther.sol SHA3: 7d6a4be7cc6f180d0b3f1a861d9d10d4d5665c68aa8a9977899e8df5558979c6 File: ./contracts/PToken/PEther/PEtherDelegate.sol SHA3: ebb9d0077cc87d32a7f20f2cc3ab5af4758091513c734e9c10968e1ca1e4014f File: ./contracts/PToken/PEther/PEtherDelegator.sol SHA3: 34e1b117fafa1a03e488c053f55433a9fbc19989e40c6d9093a62f8c801379f7 File: ./contracts/PToken/PEther/PEtherImmutable.sol SHA3: ae8d1900790a602f6088c9cfbc6ec01d906771c96a3a0ff6eb48bc6bc08fb27d File: ./contracts/PToken/PToken.sol SHA3: 8b74965229734b2836ead10d8941c4600ec056f5f065f4e3f142df56637300dd

	File: ./contracts/PToken/PTokenInterfaces.sol SHA3: 5458057206a05fe6012c8a8f6aa85dc04d964662d2a5824c0c6ec32f6d83701e File: ./contracts/Utils/ExponentialNoError.sol SHA3: 19ef154d786b2e85ef0a2449a1a16be4b812cc638845085baf02d768364eb088 File: ./contracts/Utils/Ownable.sol SHA3: 8ba8cb523b8b5d9d428ee7f8f90924d43e429e0823d1d58bfe328b44543e4758 File: ./contracts/Utils/SafeMath.sol SHA3: f3672bc098d0e16e1ca73b5bd3a47fcafb7ea08817ce5e2dd2c38327d5ce9d98

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to the loss of user funds or contract state manipulation by external or internal actors.
High	High vulnerabilities are usually harder to exploit, requiring specific conditions, or have a more limited scope, but can still lead to the loss of user funds or contract state manipulation by external or internal actors.
Medium	Medium vulnerabilities are usually limited to state manipulations but cannot lead to asset loss. Major deviations from best practices are also in this category.
Low	Low vulnerabilities are related to outdated and unused code or minor gas optimization. These issues won't have a significant impact on code execution but affect code quality

Executive Summary

The score measurement details can be found in the corresponding section of the scoring methodology.

Documentation quality

The total Documentation Quality score is **10** out of **10**.

- Functional requirements are available and detailed.
- The differences between Paribus and Compound are visible in the documents provided exclusively for Paribus.

Code quality

The total Code Quality score is **9** out of **10**.

- An outdated compiler version is used.

Test coverage

Code coverage of the project is **71.7%** (branch coverage).

- Deployment and basic user interactions are covered with tests.
- Not all negative cases/exceptions/reverts are tested.
- Interactions with several users are not tested thoroughly.

Security score

As a result of the audit, the code contains **1** low severity issue. The security score is **10** out of **10**.

All found issues are displayed in the “Findings” section.

Summary

According to the assessment, the Customer's smart contract has the following score: **8.7**.

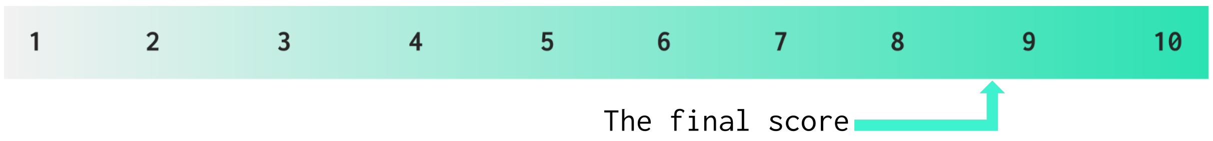


Table. The distribution of issues during the audit

Review date	Low	Medium	High	Critical
29 January 2023	12	3	4	0
22 February 2023	3	2	1	0
13 March 2023	1	0	0	0

Checked Items

We have audited the Customers' smart contracts for commonly known and specific vulnerabilities. Here are some items considered:

Item	Type	Description	Status
Default Visibility	SWC-100 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	Passed
Integer Overflow and Underflow	SWC-101	If unchecked math is used, all math operations should be safe from overflows and underflows.	Passed
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	Failed
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	Passed
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	Passed
Access Control & Authorization	CWE-284	Ownership takeover should not be possible. All crucial functions should be protected. Users could not affect data that belongs to other users.	Passed
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	Not Relevant
Check-Effect-Interaction	SWC-107	Check-Effect-Interaction pattern should be followed if the code performs ANY external call.	Passed
Assert Violation	SWC-110	Properly functioning code should never reach a failing assert statement.	Passed
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	Passed
Delegatecall to Untrusted Callee	SWC-112	Delegatecalls should only be allowed to trusted addresses.	Passed
DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless required.	Passed
Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	Passed

Authorization through tx.origin	SWC-115	tx.origin should not be used for authorization.	Not Relevant
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	Not Relevant
Signature Unique Id	SWC-117 SWC-121 SWC-122 EIP-155 EIP-712	Signed messages should always have a unique id. A transaction hash should not be used as a unique id. Chain identifiers should always be used. All parameters from the signature should be used in signer recovery. EIP-712 should be followed during a signer verification.	Not Relevant
Shadowing State Variable	SWC-119	State variables should not be shadowed.	Passed
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	Not Relevant
Incorrect Inheritance Order	SWC-125	When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order.	Passed
Calls Only to Trusted Addresses	EEA-Level-2 SWC-126	All external calls should be performed only to trusted addresses.	Passed
Presence of Unused Variables	SWC-131	The code should not contain unused variables if this is not justified by design.	Passed
EIP Standards Violation	EIP	EIP standards should not be violated.	Passed
Assets Integrity	**Custom**	Funds are protected and cannot be withdrawn without proper permissions or be locked on the contract.	Passed
User Balances Manipulation	**Custom**	Contract owners or any other third party should not be able to access funds belonging to users.	Passed
Data Consistency	**Custom**	Smart contract data should be consistent all over the data flow.	Passed
Flashloan Attack	**Custom**	When working with exchange rates, they should be received from a trusted source and not be vulnerable to short-term rate changes that can be achieved by using flash loans. Oracles should be used.	Passed

Token Supply Manipulation	**Custom**	Tokens can be minted only according to rules specified in a whitepaper or any other documentation provided by the customer.	Passed
Gas Limit and Loops	**Custom**	Transaction execution costs should not depend dramatically on the amount of data stored on the contract. There should not be any cases when execution fails due to the block Gas limit.	Passed
Style Guide Violation	**Custom**	Style guides and best practices should be followed.	Passed
Requirements Compliance	**Custom**	The code should be compliant with the requirements provided by the Customer.	Passed
Environment Consistency	**Custom**	The project should contain a configured development environment with a comprehensive description of how to compile, build and deploy the code.	Passed
Secure Oracles Usage	**Custom**	The code should have the ability to pause specific data feeds that it relies on. This should be done to protect a contract from compromised oracles.	Passed
Tests Coverage	**Custom**	The code should be covered with unit tests. Test coverage should be sufficient, with both negative and positive cases covered. Usage of contracts by multiple users should be tested.	Failed
Stable Imports	**Custom**	The code should not reference draft contracts, which may be changed in the future.	Passed

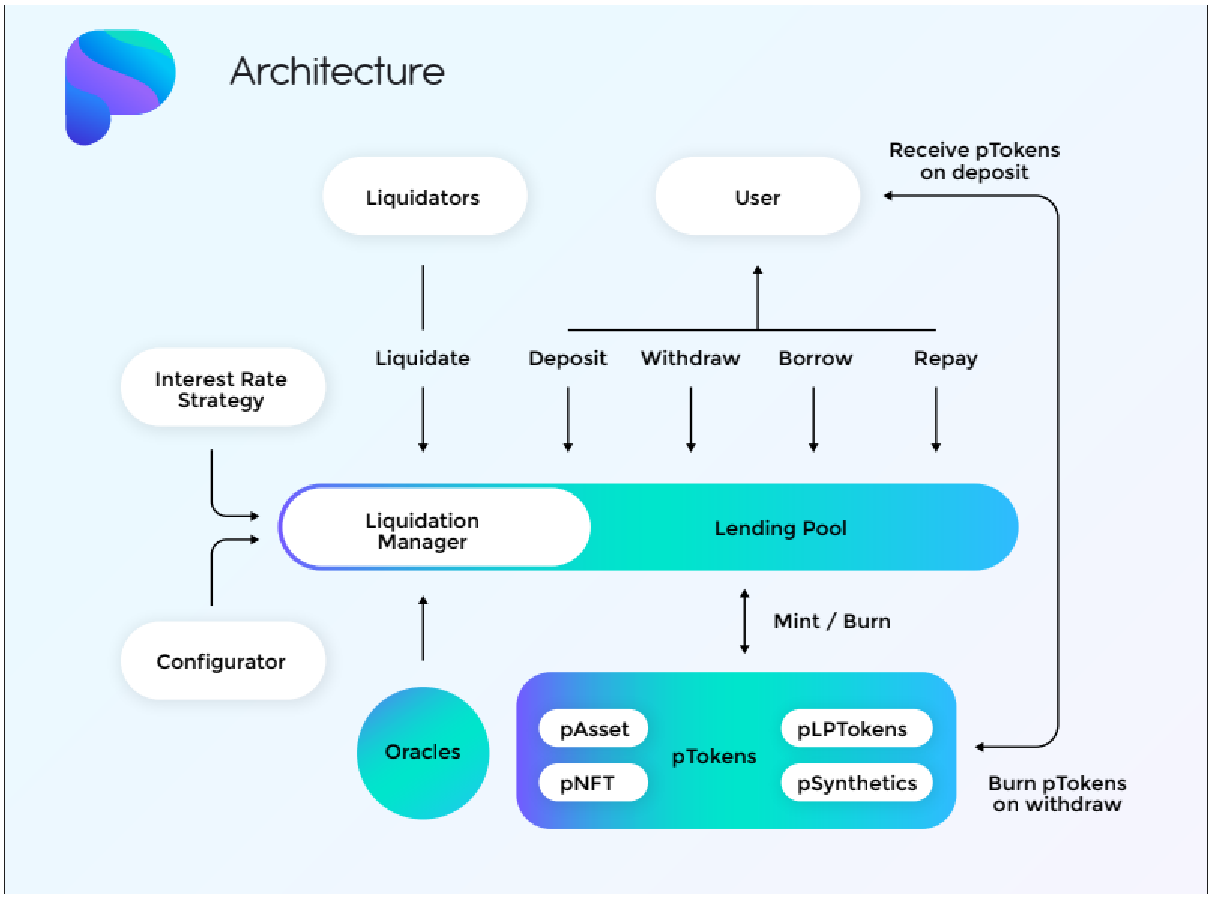
System Overview

The Paribus Protocol is an Ethereum smart contract for supplying or borrowing assets. Through the pToken contracts, accounts on the blockchain supply capital (Ether or ERC-20 tokens) to receive pTokens or borrow assets from the protocol (holding other assets as collateral). The Paribus pToken contracts track these balances and algorithmically set interest rates for borrowers.

The core contracts of the system are the following;

- **PToken, PErc20 and PEther -** The Paribus pTokens, which are self-contained borrowing and lending contracts. PToken contains the core logic and PErc20 and PEther add public interfaces for Erc20 tokens and Ether, respectively. Each PToken is assigned an interest rate and risk model (see InterestRateModel and Comptroller sections), and allows accounts to *mint* (supply capital), *redeem* (withdraw capital), *borrow* and *repay a borrow*. Each PToken is an ERC-20 compliant token where balances represent ownership of the market.
- **Comptroller -** The risk model contract, which validates permissible user actions and disallows actions if they do not fit certain risk parameters. For instance, the Comptroller enforces that each borrowing user must maintain a sufficient collateral balance across all pTokens.
- **Paribus (PBX) -** The Paribus Governance Token (PBX).
- **InterestRateModel -** Contracts which define interest rate models. These models algorithmically determine interest rates based on the current utilization of a given market (that is, how much of the supplied assets are liquid versus borrowed).
- **Careful Math -** Library for safe math operations.
- **ErrorReporter -** Library for tracking error codes and failure conditions.
- **Exponential -** Library for handling fixed-point decimal numbers.
- **SafeToken -** Library for safely handling Erc20 interactions.
- **WhitePaperInterestRateModel -** Initial interest rate model, as defined in the Whitepaper. This contract accepts a base rate and slope parameter in its constructor.

The architecture of the project is described as the following in the litepaper:



Privileged roles

Roles defined in the system are the following:

- **Lenders** - The lender is key in any DeFi platform. Most of the time these users are known as “HODLers” within the cryptocurrency space. They have no plans to outright sell their crypto assets. Paribus will be an additional outlet for this user group to earn a passive income while their underlying assets appreciate in value over time. Lenders can be thought of as liquidity providers within the ecosystem and earn interest in return for doing so. The platform will provide Deposit APR(%) based on factors such as utilization rate. Lenders can at a rough level estimate their earnings based on the Deposit APR(%) for a given asset.
- **Borrowers** - Paribus will operate strictly as a collateralized loan platform. Meaning any borrower must deposit asset(s) in order to borrow against. As a result borrowers are indirectly also liquidity providers to assure the platform is sustainable and self-sufficient. Every borrower is subject to paying a small one-time fixed platform fee as well as the interest accrued over the period of the loan. The fee mechanism is detailed in the latter section of this paper.

- **Admin -** Admins can change proxies implementations to a deployed contract address and change the contract behavior by delegating calls to the new implementation address.
- **Pause Guardian -** Can pause certain actions as a safety mechanism. Actions which allow users to remove their own assets cannot be paused.
- **Borrow Cap Guardian -** Can set borrow caps to any number for any market. Lowering the borrow cap could disable borrowing on the given market.

Risks

- The project uses outdated Solidity pragma versions. Using an outdated compiler version can be problematic, especially if there are publicly disclosed bugs and issues that affect the current compiler version.
- The project uses proxies and is upgradeable which makes it centralized. The upgradeable nature of the contracts puts the implementation at risk in case of logic upgrade. Contracts can have their implementation changed without sufficient time for users to react to bad changes (i.e, the contracts are not using any sort of an implementation changing proposals mechanism that requires a certain delay to be implemented after a request by an admin).

Recommendations

- The system relies on the security of the Admin's private keys, which can impact the execution flow and security of the funds. We recommend this account to be at least ⅗ multi-sig.

Findings

■■■■ Critical

No critical severity issues were found.

■■■ High

H01. Non-Finalized Code

The code contains *TODO*/*check* comments. It means that the code is not finalized, and additional changes will be introduced in the future.

This can lead to incorrect implementation and the loss of user funds.

Paths:
./contracts/Comptroller/ComptrollerPart1.sol
./contracts/Comptroller/ComptrollerPart2.sol
./contracts/InterestRateModels/DAIInterestRateModelV3.sol
./contracts/Liquidator.sol

Recommendation: The code should be finalized, and all *TODO* and *check* comments should be addressed.

Status: Fixed (Revised commit: d441c2e)

H02. Non-Finalized Code

The production code contains functions and contracts that are intended for testing.

The production code should not be mixed with the code that is used solely in the testing environment.

Paths:
./PBXToken : PBXTestTokenMintable
./contracts/PriceOracle/RinkebyPriceOracle.sol : RinkebyPriceOracle
./contracts/PriceOracle/PolygonPriceOracle.sol : MumbaiPriceOracle
./contracts/PriceOracle/GoerliPriceOracle.sol : GoerliPriceOracle
./contracts/PriceOracle/SimplePriceOracle.sol : SimplePriceOracle
./contracts/PriceOracle/ArbitrumPriceOracle.sol : RinkarbyPriceOracle

Recommendation: The code should be finalized. All testing functions, contracts, and mocks should be arranged in a way that makes it easy to distinguish them from production code.

Status: Fixed (Revised commit: d441c2e)

H03. Requirements Violation

In the ComptrollerPart1.sol contract, there are functions that have no implementations which contradict their NatSpec descriptions.

This may lead to unsafe implementations in the future.

Path:
./contracts/Comptroller/ComptrollerPart1.sol : mintVerify(),

borrowVerify(), repayBorrowVerify(), liquidateBorrowVerify(), seizeVerify(), transferVerify()

Recommendation: The code should be finalized according to the NatSpec. Consider updating the documentation or removing dead code.

Status: Mitigated (Mitigated in the commit: d441c2e. The NatSpec for functions indicates that:

"... Now empty, reserved for potential future use.")

H04. Undocumented Behavior

The code should not contain undocumented behavior.

The Paribus production code added to the Compound protocol is undocumented.

This can lead to confusion, misunderstanding and difficulty of further integration or code upgrades.

Paths:
./contracts/Liquidator.sol
./contracts/Comptroller/ComptrollerPart2.sol
./contracts/PriceOracle/PriceOracleInterface.sol
./contracts/PriceOracle/Impl/StablecoinsPriceOracle.sol
./contracts/PriceOracle/Impl/PriceOracleCommonImpl.sol
./contracts/PriceOracle/Impl/ChainlinkPriceOracle.sol
./contracts/PriceOracle/Impl/Api3PriceOracle.sol
./contracts/Utils/Timelock.sol

Recommendation: Provide functional and technical documentation that covers all functionality of the Paribus protocol.

The documentation for the Compound protocol is helpful, but documentation written exclusively for Paribus would be beneficial for the project.

Status: Fixed (Revised commit: 294e429)

(The documentation is now written exclusively for the Paribus Protocol and is comprehensive.)

Medium

M01. Best Practice Violation - Usage of Assert

There are some *assert()* statements in the code for control flow.

Properly functioning code should never reach a failing assert statement. A reachable assertion can mean one of two things:

- A bug exists in the contract that allows it to enter an invalid state.
- The assert statement is used incorrectly, e.g. to validate inputs.

Paths:
./contracts/Liquidator.sol

./contracts/Comptroller/ComptrollerPart2.sol
./contracts/PriceOracle/SimplePriceOracle.sol
./contracts/PriceOracle/Impl/Api3PriceOracle.sol

Recommendation: Consider whether the condition checked in *assert()* is actually an invariant. If not, replace the *assert()* statement with a *require()* statement.

If the exception is indeed caused by unexpected behavior of the code, fix the underlying bug(s) that allow the assertion to be violated.

Status: Fixed (Revised commit: 294e429)

M02. Unscalable Functionality - Shadowing State Variable

In the ComptrollerPart2.sol contract in the *updatePBXSupplyIndex()* function the PBXAccrued local variable is shadowing a storage variable from the contract.

In complex contract systems this condition could go unnoticed and subsequently lead to security issues.

Path:
./contracts/Comptroller/ComptrollerPart2.sol : updatePBXSupplyIndex()

Recommendation: Consider using a different name for the local variable used in the function.

Status: Fixed (Revised commit: d441c2e)

M03. Contradiction - Missing Validation

The *PBXToken* token address should be permanent after it is set in the *_setPBXToken()* function; however, this validation is missing.

This can lead to a situation in which the reward token in the Paribus system is changed, resulting in a loss of trust from its community.

Path:
./contracts/Comptroller/ComptrollerPart1.sol : _setPBXToken()

Recommendation: Implement validation.

Status: Fixed (Revised commit: d441c2e)

M04. Unscalable Functionality - Shadowing State Variable

In the ComptrollerPart2.sol contract in the *updatePBXBorrowIndex()* function the PBXAccrued local variable is shadowing a storage variable from the contract.

In complex contract systems this condition could go unnoticed and subsequently lead to security issues.

Path:
./contracts/Comptroller/ComptrollerPart2.sol : updatePBXBorrowIndex()

Recommendation: Consider using a different name for the local variable used in the function.

Status: Fixed (Revised commit: 294e429)

Low

L01. Floating Pragma

Locking the pragma helps to ensure that contracts are not accidentally deployed using an outdated compiler version that might introduce bugs that affect the contract system negatively.

Paths:
./contracts/Comptroller/ComptrollerCommonImpl.sol
./contracts/Comptroller/ComptrollerInterface.sol
./contracts/Comptroller/ComptrollerPart1.sol
./contracts/Comptroller/ComptrollerPart2.sol
./contracts/Comptroller/ComptrollerStorage.sol
./contracts/Comptroller/Unitroller.sol
./contracts/ErrorReporter.sol
./contracts/InterestRateModels/BaseJumpRateModelV2.sol
./contracts/InterestRateModels/DAIInterestRateModelV3.sol
./contracts/InterestRateModels/InterestRateModelInterface.sol
./contracts/InterestRateModels/JumpRateModel.sol
./contracts/InterestRateModels/JumpRateModelV2.sol
./contracts/InterestRateModels/LegacyInterestRateModel.sol
./contracts/InterestRateModels/LegacyJumpRateModelV2.sol
./contracts/InterestRateModels/WhitePaperInterestRateModel.sol
./contracts/Interfaces/AaveInterfaces.sol
./contracts/Interfaces/Api3Interfaces.sol
./contracts/Interfaces/EIP20Interface.sol
./contracts/Interfaces/EIP20NonStandardInterface.sol
./contracts/Interfaces/UniswapV2Interfaces.sol
./contracts/Interfaces/UniswapV3Interfaces.sol
./contracts/Liquidator.sol
./contracts/Maximillion.sol
./contracts/PBXToken.sol
./contracts/PriceOracle/ArbitrumPriceOracle.sol
./contracts/PriceOracle/GoerliPriceOracle.sol
./contracts/PriceOracle/Impl/Api3PriceOracle.sol
./contracts/PriceOracle/Impl/ChainlinkPriceOracle.sol
./contracts/PriceOracle/Impl/PriceOracleCommonImpl.sol
./contracts/PriceOracle/Impl/StablecoinsPriceOracle.sol
./contracts/PriceOracle/PolygonPriceOracle.sol
./contracts/PriceOracle/PriceOracleInterface.sol
./contracts/PriceOracle/RinkebyPriceOracle.sol
./contracts/PriceOracle/SimplePriceOracle.sol
./contracts/PToken/PErc20/PErc20.sol
./contracts/PToken/PErc20/PErc20Delegate.sol
./contracts/PToken/PErc20/PErc20Delegator.sol
./contracts/PToken/PErc20/PErc20Immutable.sol
./contracts/PToken/PEther/PEther.sol
./contracts/PToken/PEther/PEtherDelegate.sol
./contracts/PToken/PEther/PEtherDelegator.sol
./contracts/PToken/PEther/PEtherImmutable.sol

./contracts/PToken/PToken.sol
./contracts/PToken/PTokenInterfaces.sol
./contracts/Utils/ExponentialNoError.sol
./contracts/Utils/Ownable.sol
./contracts/Utils/SafeMath.sol
./contracts/Utils/Timelock.sol

Recommendation: Consider locking the pragma version whenever possible and avoid using a floating pragma in the final deployment.

Status: Fixed (Revised commit: d441c2e)

L02. Style Guide Violation

The project should follow the official code style guidelines.
Inside each contract, library, or interface, use the following order:

- Type declarations
- State variables
- Events
- Modifiers
- Functions

Functions should be grouped according to their visibility and ordered:

- constructor
- receive function (if exists)
- fallback function (if exists)
- external
- public
- internal
- private

Within a grouping, place the view and pure functions at the end.

Paths:
./contracts/Comptroller/ComptrollerCommonImpl.sol
./contracts/Comptroller/ComptrollerPart1.sol
./contracts/Comptroller/ComptrollerPart2.sol
./contracts/InterestRateModels/BaseJumpRateModelV2.sol
./contracts/InterestRateModels/DAIInterestRateModelV3.sol
./contracts/InterestRateModels/JumpRateModelV2.sol
./contracts/Liquidator.sol
./contracts/PToken/PErc20/PErc20Delegator.sol
./contracts/PToken/PEther/PEtherDelegator.sol
./contracts/PToken/PToken.sol
./contracts/PriceOracle/Impl/PriceOracleCommonImpl.sol

Recommendation: The official Solidity style guidelines should be followed.

Status: Mitigated (Customer follows a custom function order that provides good readability for this specific project.)

L03. Unused Function Arguments

Unused function arguments should be removed from the contracts. This will help lower the Gas cost.

The function `getHypotheticalAccountLiquidityInternal()` receives the argument `redeemTokenId`, but does not use it anywhere.

Path: ./contracts/Comptroller/ComptrollerPart2.sol

Recommendation: Remove unused variables from the code.

Status: Mitigated (Mitigated in the commit: d441c2e. The NatSpec has been updated indicating that *redeemTokenId* is:

“Unused, reserved for NFT code”.)

L04. Best Practices - Modifiers

In the ComptrollerCommonImpl.sol contract, the functions *adminOrInitializing()* and *onlyAdmin()* are only used to check certain conditions before executing other functions.

These functions can be converted into modifiers for better readability.

Path: ./contracts/Comptroller/ComptrollerCommonImpl.sol

Recommendation: Consider converting these functions to modifiers.

Status: Mitigated (With customer notice:

“... the compiled binary code with functions seems to be a bit smaller than the one with modifiers. Contract size limit is a huge problem in Comptroller.”.)

L05. Missing Zero Address Validation

Address parameters are being used without checking against the possibility of 0x0. This can lead to unwanted external calls to 0x0.

The argument `PBXTokenAddress` on `_setPBXToken` doesn’t check if the address is a zero address.

Path: ./contracts/Comptroller/ComptrollerPart1.sol

Recommendation: Implement zero address checks.

Status: Fixed (Revised commit: d441c2e)

L06. Checks-Effects-Interactions Violation

The Checks-Effects-Interactions pattern is violated. In the functions `liquidateBorrowInternal`, `redeemFresh` and `_setImplementation`, some state variables are updated after the external calls.

Paths:
./contracts/PToken/PToken.sol

./contracts/PToken/PEther/PEtherDelegator.sol
./contracts/PToken/PErc20/PErc20Delegator.sol

Recommendation: The code should follow the Checks-Effects-Interactions pattern.

Status: Mitigated (Mentioned functions violate the CEI pattern, but are guarded by a 'nonReentrant' modifier or can only be called by an Admin.)

L07. Missing Events

Events for critical state changes should be emitted for tracking things off-chain.

The function `transferOwnership` does not emit an event for the critical state change.

Path: ./contracts/Utils/Ownable.sol

Recommendation: Create and emit related events.

Status: Fixed (Revised commit: d441c2e)

L08. Functions That Can Be Declared External

"public" functions that are never called by the contract should be declared "external" to save gas.

Paths:
./contracts/Utils/Timelock.sol
./contracts/PToken/PErc20/PErc20Delegate.sol
./contracts/Comptroller/ComptrollerCommonImpl.sol
./contracts/Comptroller/ComptrollerInterface.sol
./contracts/Comptroller/ComptrollerPart1.sol
./contracts/Comptroller/ComptrollerPart2.sol
./contracts/Comptroller/Unitroller.sol

Recommendation: Use the "external" attribute for functions never called from the contract.

Status: Fixed (Revised commit: d441c2e)

L09. Boolean Equality

Boolean values can be checked directly and do not need to be compared to true or false.

Paths:
./contracts/Comptroller/ComptrollerPart1.sol
./contracts/Comptroller/ComptrollerPart2.sol

Recommendation: Remove boolean equality.

Status: Fixed (Revised commit: d441c2e)

L10. Unindexed Events

Having indexed parameters in the events makes it easier to search for these events using indexed parameters as filters.

Paths:
./contracts/ErrorReporter.sol
./contracts/Comptroller/ComptrollerInterface.sol
./contracts/Comptroller/Unitroller.sol
./contracts/InterestRateModels/BaseJumpRateModelV2.sol
./contracts/InterestRateModels/JumpRateModel.sol
./contracts/InterestRateModels/WhitePaperInterestRateModel.sol
./contracts/PToken/PTokenInterfaces.sol

Recommendation: Use the “indexed” keyword for at least one of the event parameters.

Status: Mitigated (With customer notice:

“... decided not to include the indexed parameter on all set events (like event NewReserveFactor(uint oldReserveFactorMantissa, uint newReserveFactorMantissa)) because I highly doubt that anyone will try to search events by parameters like reserveFactorMantissa.”.)

L11. No Messages in Require Conditions

Some require/assert statements are missing error messages.

`PToken.isPToken()` is called as an assert mechanism to make sure the token is compliant with the system but these calls alone will cause reverts without a message due to an incorrect external address ABI layout.

This makes the code harder to test and debug.

Paths:
./contracts/Comptroller/ComptrollerPart1.sol
./contracts/Comptroller/ComptrollerPart2.sol
./contracts/Liquidator.sol
./contracts/PriceOracle/Impl/Api3PriceOracle.sol
./contracts/PriceOracle/SimplePriceOracle.sol
./contracts/PToken/PToken.sol
./contracts/PToken/PErc20/PErc20Delegator.sol
./contracts/PToken/PEther/PEtherDelegator.sol

Recommendations: All require/assert/revert statements should have an error message. The token compliance check should be made safely (e.g checking if target address supports the interface using ERC165) and revert with a reasonable error message.

Status: Fixed (Revised commit: 294e429)

L12. Outdated Compiler Version

Using an outdated compiler version can be problematic especially if there are publicly disclosed bugs and issues that affect the current compiler version.

Using the current version of Solidity is generally considered best practice because it includes the latest updates and bug fixes. Newer versions address security vulnerabilities that may have been discovered in previous versions, making them more secure to use. Additionally, newer versions include new features and improvements that make writing and deploying smart contracts easier and more efficient. Using an outdated version of Solidity may expose your contracts to potential security risks and make it more difficult to take advantage of newer features and capabilities.

There is no reason to use an outdated Solidity version; the Compound protocol contracts that the Paribus protocol uses as its main backbone have been updated to version ^0.8.10.

Paths:
./contracts/Comptroller/ComptrollerCommonImpl.sol
./contracts/Comptroller/ComptrollerInterface.sol
./contracts/Comptroller/ComptrollerPart1.sol
./contracts/Comptroller/ComptrollerPart2.sol
./contracts/Comptroller/ComptrollerStorage.sol
./contracts/Comptroller/Unitroller.sol
./contracts/ErrorReporter.sol
./contracts/InterestRateModels/BaseJumpRateModelV2.sol
./contracts/InterestRateModels/DAIInterestRateModelV3.sol
./contracts/InterestRateModels/InterestRateModelInterface.sol
./contracts/InterestRateModels/JumpRateModel.sol
./contracts/InterestRateModels/JumpRateModelV2.sol
./contracts/InterestRateModels/LegacyInterestRateModel.sol
./contracts/InterestRateModels/LegacyJumpRateModelV2.sol
./contracts/InterestRateModels/WhitePaperInterestRateModel.sol
./contracts/Interfaces/AaveInterfaces.sol
./contracts/Interfaces/Api3Interfaces.sol
./contracts/Interfaces/EIP20Interface.sol
./contracts/Interfaces/EIP20NonStandardInterface.sol
./contracts/Interfaces/UniswapV2Interfaces.sol
./contracts/Interfaces/UniswapV3Interfaces.sol
./contracts/Liquidator.sol
./contracts/Maximillion.sol
./contracts/PBXToken.sol
./contracts/PriceOracle/ArbitrumPriceOracle.sol
./contracts/PriceOracle/GoerliPriceOracle.sol
./contracts/PriceOracle/Impl/Api3PriceOracle.sol
./contracts/PriceOracle/Impl/ChainlinkPriceOracle.sol
./contracts/PriceOracle/Impl/PriceOracleCommonImpl.sol
./contracts/PriceOracle/Impl/StablecoinsPriceOracle.sol
./contracts/PriceOracle/PolygonPriceOracle.sol
./contracts/PriceOracle/PriceOracleInterface.sol
./contracts/PriceOracle/RinkebyPriceOracle.sol
./contracts/PriceOracle/SimplePriceOracle.sol
./contracts/PToken/PErc20/PErc20.sol
./contracts/PToken/PErc20/PErc20Delegate.sol
./contracts/PToken/PErc20/PErc20Delegator.sol
./contracts/PToken/PErc20/PErc20Immutable.sol
./contracts/PToken/PEther/PEther.sol
./contracts/PToken/PEther/PEtherDelegate.sol
./contracts/PToken/PEther/PEtherDelegator.sol

./contracts/PToken/PEther/PEtherImmutable.sol
./contracts/PToken/PToken.sol
./contracts/PToken/PTokenInterfaces.sol
./contracts/Utils/ExponentialNoError.sol
./contracts/Utils/Ownable.sol
./contracts/Utils/SafeMath.sol
./contracts/Utils/Timelock.sol

Recommendations: It is recommended to use a recent version of the Solidity compiler.

Status: Reported (The recent version of the Solidity compiler is not used.)

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed based on best industry practices at the time of the writing of this report, with cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The report contains no statements or warranties on the identification of all vulnerabilities and security of the code. The report covers the code submitted and reviewed, so it may not be relevant after any modifications. Do not consider this report as a final and sufficient assessment regarding the utility and safety of the code, bug-free status, or any other contract statements.

While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

English is the original language of the report. The Consultant is not responsible for the correctness of the translated versions.

Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, the Consultant cannot guarantee the explicit security of the audited smart contracts.