QuillAudits

Audit Report
February, 2022

For

PROXY

Contents

Contents

Scope of the Audit

The scope of this audit was to analyze and document the PRXY Bond smart contract codebase for quality, security, and correctness.

Checked Vulnerabilities

We have scanned the smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered:

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level

Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis
In this step, we have analysed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis
Static analysis of smart contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis
Manual analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analysed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption
In this step, we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit
Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis, Theo.

Issue Categories

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below.

Risk-level	Description
High	A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.
Medium	The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.
Low	Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.
Informational	These are severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Number of issues per severity

Type	High	Medium	Low	Informational
Open	0	0	0	0
Acknowledged	0	0	0	3
Closed	0	3	3	1

Introduction

During the period of **Jan 21, 2022 to Feb 05, 2022** - QuillAudits Team performed a security audit for **PRXY Bond** smart contracts.

The code for the audit was taken from following the official link:
https://github.com/Proxy-Protocol/prxy-bond-contracts

V	Date	Commit ID	Branch
1	Jan 30	2315e97d017cbe5443e49 abcccdd08c3449797e0	new_usdc_bond
2	Feb 05	edee94c4062864292c78 7ccf42cb481abff0c02f	prxy-bond-contracts

Issues Found – Code Review / Manual Testing

A.Contract - PRXY Bond

High severity issues

No issues were found.

Medium severity issues

1. **Everyone can redeem without depositor's consent**

   Description

   As the **redeem()** function is set to external, hence, everyone can call this function and redeem the token on behalf of any depositors without their consent.

```
function redeem(address _recipient, bool _stake) external returns (uint) {
        require(redeemStatus, "Redeem is disabled");
        Bond memory info = bondInfo[_recipient];
```

   Any caller placing a valid _receipient address associated with a bond can redeem the bond.

   We recommend changing the business logic for this function, therefore, the validated depositors can redeem by themselves.

   Status: Fixed in version 02

2. **Possibility of Reentrancy attack**

   Description

   External call was detected in the **deposit()** function and **redeem()** function, assume that malicious code might execute. Even if that external contract is not malicious, malicious code can be executed by any contracts it calls.

   One particular danger is malicious code may hijack the control flow, leading to vulnerabilities due to reentrancy.

   We recommend adding nonReentrant() for the aforementioned functions to avoid Reentrancy weaknesses.

   Status: Fixed in version 02

3. Unchecked transfer

The return value of an external transfer/transferFrom call is not checked.

```
function stakeOrSend(address _recipient, bool _stake, uint _amount) internal returns (uint)
{
        if (!_stake) {// if user does not want to stake
            IERC20(PRXY).transfer(_recipient, _amount);
            // send payout
        } else {// if user wants to stake
            if (useHelper) {// use if staking warmup is 0
                IERC20(PRXY).approve(stakingHelper, _amount);
                IStakingHelper(stakingHelper).stake(_amount, _recipient);
            } else {
                IERC20(PRXY).approve(staking, _amount);
                IStaking(staking).stake(_amount, _recipient);
            }
        }
        return _amount;
    }
```

Please consider adding the require() check for the external calls at line 886 or using SafeERC20, or ensure that the transfer/transferFrom return value is checked.

Status: Fixed in version 02

Low severity issues

4. Redundant code

Line	Code

954	function _bondPrice() internal view returns (uint price_) {
	uint _assetPrice = assetPrice();
	price_ =
	_assetPrice.sub(_assetPrice.mul(terms.discountPercentage).div(100));
	}

Please consider removing the aforementioned function if it’s not used.

Status: Fixed in version 02

5. Input value checks

The function initializeBondTerms does not sanitize its arguments, can take any values for all except for discountPercentage. However, the function setBondTerms seems to indicate that inputs for a bond like _vestingTerm can not be less than 36 hours. Therefore bonds with terms less than 36 hours or too high maxPayout or maxDebt can be created. There is no clear indication as to whether initialieBondTerms is called first and then setBondTerms and whether for every initialized bond, setBondTerms has to be called to restrict parameters to these constraints.

```
function initializeBondTerms(
        uint _vestingTerm,
        uint _discountPercentage,
        uint _maxPayout,
        uint _fee,
        uint _maxDebt,
        uint _initialDebt
    ) external onlyPolicy() {
        require(_discountPercentage <= 99, "Discount can not be more than 99");
        terms = Terms({
        vestingTerm : _vestingTerm,
        discountPercentage : _discountPercentage,
        maxPayout : _maxPayout,
        fee : _fee,
        maxDebt : _maxDebt
        });
        totalDebt = _initialDebt;
        lastDecay = block.number;
    }
```

It is recommended to create state variables for the bondConstraints like _vestingTerm, _discountPercentage and make sanity checks for these values when calling initializeBondTerms. For example use require checks.

```
const public minVestingLength = 36 hours;
require(_input >= minVestingLength, "Vesting must be longer than 36 hours");
```

Status: Fixed in version 02

6. Public functions that can be made external

Public functions that are not called within the contract can be made external. External functions cost less than public functions.

```
function calculateValues(uint _amount) public view returns (uint value, uint payout, uint
fee, uint profit, uint send_) {
        value = ITreasury(treasury).valueOf(principle, _amount);
        payout = payoutFor(value);
        fee = payout.mul(terms.fee).div(10000);
        profit = value.sub(payout).sub(fee);
        uint newValue = ITreasury(treasury).valueOf(principle, _amount);
        send_ = newValue.sub(profit);
    }

function currentDebt() public view returns (uint) {
        return totalDebt.sub(debtDecay());
    }
```

It is recommended to make the function calculateValues() and currentDebt() external functions to save on gas costs.

Status: Fixed in version 02

Informational issues

7. Missing Events for Significant Transactions

Description

The missing event makes it difficult to track off-chain l. An event should be emitted for significant transactions calling the following functions:

- initializeBondTerms()
- setBondTerms()
- setStaking()

Remediation

We recommend emitting an event to log the update of the above functions.

Status: Acknowledged

8. Old version of Solidity is used

Description

solc frequently releases new compiler versions. Using an old version prevents access to new Solidity security checks.

We also recommend using the solidity version >= 0.8.0.

Status: Acknowledged

9. No error messages in require() functions

Description

There are few places in the entire codebase where the require() statement does not contain any error message. As error messages are intended to notify users about failing conditions, they should provide enough information so that appropriate corrections can be made to interact with the system. Below is a non-exhaustive list of identified instances:

- L685: require(_PRXY != address(0));
- L687: require(_principle != address(0));
- L689: require(_treasury != address(0));
- L691: require(_DAO != address(0));
- L693: require(_feed != address(0));
- L761: require(_staking != address(0));
- L963: require(PRXY != address(0));
- L964: require(principle != address(0));
- L1044: require(_token != PRXY);
- L1045: require(_token != principle);

Remediation

Lack of error messages greatly damage the overall user experience, thus lowering the system's quality. Consider not only fixing the specific instances mentioned above, but also reviewing the entire codebase to make sure every error message is informative and user-friendly.

Status: Fixed in version 02

10. Variables declared as uint instead of uint256

Description

To favor explicitness, consider changing all instances of uint into uint256 in the entire codebase.

Status: Acknowledged

Functional Tests

Setter functions

Name of the test	Status
✓ get the DAO address of PRXYUSDCBond	Passed
✓ should call initializeBondTerms only by owner (68ms)	Passed
✓ should call setBondTerms only by owner (62ms)	Passed
✓ should revert if vestingTerm < 10000	Passed
✓ should revert if maxPayout >= 1000	Passed
✓ should revert if fee >= 10000	Passed
✓ should set maxDebt to any value	Passed
✓ should revert if discountPercentage >= 99	Passed
✓ should call setStaking() only by owner (40ms)	Passed
✓ should call setDepositStatus() only by owner (41ms)	Passed
✓ should call setRedeemStatus() only by owner	Passed
✓ should call renounceManagement() only by owner	Passed
✓ should call pushManagement() only by owner	Passed
✓ should call recoverETH() only by owner	Passed
✓ should call recoverExcessToken() only by owner (55ms)	Passed
✓ should not pass the require check of recoverLostToken()	Passed

Deposit

Name of the test	Status
✓ should revert if Deposit is disabled (52ms)	Passed
✓ should revert if _depositor is zero address	Passed
✓ should call recoverExcessToken only by owner	Passed
✓ testing deposit (34589ms)	Passed

Redeem

Name of the test	Status
✓ should revert if setRedeemStatus is disabled	Passed
✓ testing Redeem	Passed

Automated Tests

Slither

```
INFO:Detectors:
PRXYUSDCBond.stakeOrSend(address,bool,uint256) (PRXYUSDCBond.sol#884-898) ignores return value by IERC20(PRXY).transfer(_recipient,_amount) (PRXYUSDCBond.sol#
886)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
FullMath.fullDiv(uint256,uint256,uint256) (PRXYUSDCBond.sol#512-531) performs a multiplication on the result of a division:
        -d /= pow2 (PRXYUSDCBond.sol#518)
        -r *= 2 - d * r (PRXYUSDCBond.sol#522)
FullMath.fullDiv(uint256,uint256,uint256) (PRXYUSDCBond.sol#512-531) performs a multiplication on the result of a division:
        -d /= pow2 (PRXYUSDCBond.sol#518)
        -r *= 2 - d * r (PRXYUSDCBond.sol#523)
FullMath.fullDiv(uint256,uint256,uint256) (PRXYUSDCBond.sol#512-531) performs a multiplication on the result of a division:
        -d /= pow2 (PRXYUSDCBond.sol#518)
        -r *= 2 - d * r (PRXYUSDCBond.sol#524)
FullMath.fullDiv(uint256,uint256,uint256) (PRXYUSDCBond.sol#512-531) performs a multiplication on the result of a division:
        -d /= pow2 (PRXYUSDCBond.sol#518)
        -r *= 2 - d * r (PRXYUSDCBond.sol#525)
FullMath.fullDiv(uint256,uint256,uint256) (PRXYUSDCBond.sol#512-531) performs a multiplication on the result of a division:
        -d /= pow2 (PRXYUSDCBond.sol#518)
        -r *= 2 - d * r (PRXYUSDCBond.sol#526)
FullMath.fullDiv(uint256,uint256,uint256) (PRXYUSDCBond.sol#512-531) performs a multiplication on the result of a division:
        -d /= pow2 (PRXYUSDCBond.sol#518)
        -r *= 2 - d * r (PRXYUSDCBond.sol#527)
FullMath.fullDiv(uint256,uint256,uint256) (PRXYUSDCBond.sol#512-531) performs a multiplication on the result of a division:
        -d /= pow2 (PRXYUSDCBond.sol#518)
        -r *= 2 - d * r (PRXYUSDCBond.sol#528)
FullMath.fullDiv(uint256,uint256,uint256) (PRXYUSDCBond.sol#512-531) performs a multiplication on the result of a division:
        -d /= pow2 (PRXYUSDCBond.sol#518)
        -r *= 2 - d * r (PRXYUSDCBond.sol#529)
FullMath.fullDiv(uint256,uint256,uint256) (PRXYUSDCBond.sol#512-531) performs a multiplication on the result of a division:
        -l /= pow2 (PRXYUSDCBond.sol#519)
        -l * r (PRXYUSDCBond.sol#530)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
policy() should be declared external:
        - Ownable.policy() (PRXYUSDCBond.sol#27-29)
renounceManagement() should be declared external:
        - Ownable.renounceManagement() (PRXYUSDCBond.sol#36-39)
pushManagement(address) should be declared external:
        - Ownable.pushManagement(address) (PRXYUSDCBond.sol#41-45)
pullManagement() should be declared external:
        - Ownable.pullManagement() (PRXYUSDCBond.sol#47-51)
symbol() should be declared external:
        - ERC20.symbol() (PRXYUSDCBond.sol#287-289)
decimals() should be declared external:
        - ERC20.decimals() (PRXYUSDCBond.sol#291-293)
totalSupply() should be declared external:
        - ERC20.totalSupply() (PRXYUSDCBond.sol#295-297)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (PRXYUSDCBond.sol#299-301)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (PRXYUSDCBond.sol#303-306)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (PRXYUSDCBond.sol#308-310)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (PRXYUSDCBond.sol#312-315)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (PRXYUSDCBond.sol#317-321)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (PRXYUSDCBond.sol#323-326)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (PRXYUSDCBond.sol#328-331)
permit(address,address,uint256,uint256,uint8,bytes32,bytes32) should be declared external:
        - ERC20Permit.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (PRXYUSDCBond.sol#436-457)
nonces(address) should be declared external:
        - ERC20Permit.nonces(address) (PRXYUSDCBond.sol#459-461)
calculateValues(uint256) should be declared external:
        - PRXYUSDCBond.calculateValues(uint256) (PRXYUSDCBond.sol#924-931)
currentDebt() should be declared external:
        - PRXYUSDCBond.currentDebt() (PRXYUSDCBond.sol#984-986)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

```
INFO:Detectors:
FixedPoint.slitherConstructorConstantVariables() (PRXYUSDCBond.sol#547-585) uses literals with too many digits:
        - Q112 = 0x10000000000000000000000000000 (PRXYUSDCBond.sol#558)
FixedPoint.slitherConstructorConstantVariables() (PRXYUSDCBond.sol#547-585) uses literals with too many digits:
        - Q224 = 0x100000000000000000000000000000000000000000000000000000000 (PRXYUSDCBond.sol#559)
PRXYUSDCBond.deposit(uint256,address) (PRXYUSDCBond.sol#782-834) uses literals with too many digits:
        - require(bool,string)(payout >= 10000000000000000,Bond too small) (PRXYUSDCBond.sol#797)
PRXYUSDCBond.maxPayout() (PRXYUSDCBond.sol#917-919) uses literals with too many digits:
        - IERC20(PRXY).totalSupply().mul(terms.maxPayout).div(100000) (PRXYUSDCBond.sol#918)
PRXYUSDCBond.bondPriceInUSD() (PRXYUSDCBond.sol#976-978) uses literals with too many digits:
        - price_ = bondPrice().mul(10 ** IERC20(principle).decimals()).div(1000000) (PRXYUSDCBond.sol#977)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
ERC20.ERC20TOKEN_ERC1820_INTERFACE_ID (PRXYUSDCBond.sol#263) is never used in ERC20Permit (PRXYUSDCBond.sol#409-462)
FixedPoint.Q224 (PRXYUSDCBond.sol#559) is never used in FixedPoint (PRXYUSDCBond.sol#547-585)
FixedPoint.LOWER_MASK (PRXYUSDCBond.sol#560) is never used in FixedPoint (PRXYUSDCBond.sol#547-585)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
```

```
INFO:Detectors:
Variable Ownable._owner (PRXYUSDCBond.sol#16) is not in mixedCase
Variable Ownable._newOwner (PRXYUSDCBond.sol#17) is not in mixedCase
Parameter Address.addressToString(address)._address (PRXYUSDCBond.sol#220) is not in mixedCase
Variable ERC20._balances (PRXYUSDCBond.sol#265) is not in mixedCase
Variable ERC20._allowances (PRXYUSDCBond.sol#267) is not in mixedCase
Variable ERC20._totalSupply (PRXYUSDCBond.sol#269) is not in mixedCase
Variable ERC20._name (PRXYUSDCBond.sol#271) is not in mixedCase
Variable ERC20._symbol (PRXYUSDCBond.sol#273) is not in mixedCase
Variable ERC20._decimals (PRXYUSDCBond.sol#275) is not in mixedCase
Variable ERC20Permit.DOMAIN_SEPARATOR (PRXYUSDCBond.sol#417) is not in mixedCase
Struct FixedPoint.uq112x112 (PRXYUSDCBond.sol#549-551) is not in CapWords
Struct FixedPoint.uq144x112 (PRXYUSDCBond.sol#553-555) is not in CapWords
Parameter PRXYUSDCBond.initializeBondTerms(uint256,uint256,uint256,uint256,uint256,uint256)._vestingTerm (PRXYUSDCBond.sol#707) is not in mixedCase
Parameter PRXYUSDCBond.initializeBondTerms(uint256,uint256,uint256,uint256,uint256,uint256)._discountPercentage (PRXYUSDCBond.sol#708) is not in mi
Parameter PRXYUSDCBond.initializeBondTerms(uint256,uint256,uint256,uint256,uint256,uint256)._maxPayout (PRXYUSDCBond.sol#709) is not in mixedCase
Parameter PRXYUSDCBond.initializeBondTerms(uint256,uint256,uint256,uint256,uint256,uint256)._fee (PRXYUSDCBond.sol#710) is not in mixedCase
Parameter PRXYUSDCBond.initializeBondTerms(uint256,uint256,uint256,uint256,uint256,uint256)._maxDebt (PRXYUSDCBond.sol#711) is not in mixedCase
Parameter PRXYUSDCBond.initializeBondTerms(uint256,uint256,uint256,uint256,uint256,uint256)._initialDebt (PRXYUSDCBond.sol#712) is not in mixedCase
Parameter PRXYUSDCBond.setBondTerms(PRXYUSDCBond.PARAMETER,uint256)._parameter (PRXYUSDCBond.sol#737) is not in mixedCase
Parameter PRXYUSDCBond.setBondTerms(PRXYUSDCBond.PARAMETER,uint256)._input (PRXYUSDCBond.sol#737) is not in mixedCase
Parameter PRXYUSDCBond.setStaking(address,bool)._staking (PRXYUSDCBond.sol#760) is not in mixedCase
Parameter PRXYUSDCBond.setStaking(address,bool)._helper (PRXYUSDCBond.sol#760) is not in mixedCase
Parameter PRXYUSDCBond.deposit(uint256,address)._amount (PRXYUSDCBond.sol#783) is not in mixedCase
Parameter PRXYUSDCBond.deposit(uint256,address)._depositor (PRXYUSDCBond.sol#784) is not in mixedCase
Parameter PRXYUSDCBond.redeem(address,bool)._recipient (PRXYUSDCBond.sol#842) is not in mixedCase
Parameter PRXYUSDCBond.redeem(address,bool)._stake (PRXYUSDCBond.sol#842) is not in mixedCase
Parameter PRXYUSDCBond.stakeOrSend(address,bool,uint256)._recipient (PRXYUSDCBond.sol#884) is not in mixedCase
Parameter PRXYUSDCBond.stakeOrSend(address,bool,uint256)._stake (PRXYUSDCBond.sol#884) is not in mixedCase
Parameter PRXYUSDCBond.stakeOrSend(address,bool,uint256)._amount (PRXYUSDCBond.sol#884) is not in mixedCase
Parameter PRXYUSDCBond.calculateValues(uint256)._amount (PRXYUSDCBond.sol#924) is not in mixedCase
Parameter PRXYUSDCBond.payoutFor(uint256)._value (PRXYUSDCBond.sol#938) is not in mixedCase
Parameter PRXYUSDCBond.percentVestedFor(address)._depositor (PRXYUSDCBond.sol#1006) is not in mixedCase
Parameter PRXYUSDCBond.pendingPayoutFor(address)._depositor (PRXYUSDCBond.sol#1023) is not in mixedCase
Parameter PRXYUSDCBond.recoverLostToken(address)._token (PRXYUSDCBond.sol#1043) is not in mixedCase
Parameter PRXYUSDCBond.setDepositStatus(bool)._depositStatus (PRXYUSDCBond.sol#1062) is not in mixedCase
Parameter PRXYUSDCBond.setRedeemStatus(bool)._redeemStatus (PRXYUSDCBond.sol#1070) is not in mixedCase
Variable PRXYUSDCBond.PRXY (PRXYUSDCBond.sol#624) is not in mixedCase
Variable PRXYUSDCBond.DAO (PRXYUSDCBond.sol#627) is not in mixedCase
```

```
INFO:Detectors:
Address._verifyCallResult(bool,bytes,string) (PRXYUSDCBond.sol#204-218) is never used and should be removed
Address.addressToString(address) (PRXYUSDCBond.sol#220-235) is never used and should be removed
Address.functionCall(address,bytes) (PRXYUSDCBond.sol#136-138) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (PRXYUSDCBond.sol#144-146) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (PRXYUSDCBond.sol#148-155) is never used and should be removed
Address.functionDelegateCall(address,bytes) (PRXYUSDCBond.sol#192-194) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (PRXYUSDCBond.sol#196-202) is never used and should be removed
Address.functionStaticCall(address,bytes) (PRXYUSDCBond.sol#180-182) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (PRXYUSDCBond.sol#184-190) is never used and should be removed
Address.sendValue(address,uint256) (PRXYUSDCBond.sol#128-134) is never used and should be removed
Counters.decrement(Counters.Counter) (PRXYUSDCBond.sol#404-406) is never used and should be removed
ERC20._burn(address,uint256) (PRXYUSDCBond.sol#352-360) is never used and should be removed
ERC20._mint(address,uint256) (PRXYUSDCBond.sol#344-350) is never used and should be removed
FixedPoint.decode(FixedPoint.uq112x112) (PRXYUSDCBond.sol#562-564) is never used and should be removed
PRXYUSDCBond._bondPrice() (PRXYUSDCBond.sol#954-957) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (PRXYUSDCBond.sol#476-482) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (PRXYUSDCBond.sol#489-492) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (PRXYUSDCBond.sol#484-487) is never used and should be removed
SafeMath.mod(uint256,uint256) (PRXYUSDCBond.sol#95-97) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (PRXYUSDCBond.sol#99-102) is never used and should be removed
SafeMath.sqrrt(uint256) (PRXYUSDCBond.sol#104-115) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (PRXYUSDCBond.sol#128-134):
        - (success) = recipient.call{value: amount}() (PRXYUSDCBond.sol#132)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (PRXYUSDCBond.sol#148-155):
        - (success,returndata) = target.call{value: value}(data) (PRXYUSDCBond.sol#153)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (PRXYUSDCBond.sol#157-178):
        - (success,returndata) = target.call{value: weiValue}(data) (PRXYUSDCBond.sol#161)
Low level call in Address.functionStaticCall(address,bytes,string) (PRXYUSDCBond.sol#184-190):
        - (success,returndata) = target.staticcall(data) (PRXYUSDCBond.sol#188)
Low level call in Address.functionDelegateCall(address,bytes,string) (PRXYUSDCBond.sol#196-202):
        - (success,returndata) = target.delegatecall(data) (PRXYUSDCBond.sol#200)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

```
INFO:Detectors:
Reentrancy in PRXYUSDCBond.deposit(uint256,address) (PRXYUSDCBond.sol#782-834):
        External calls:
        - IERC20(principle).safeTransferFrom(msg.sender,address(this),_amount) (PRXYUSDCBond.sol#810)
        - IERC20(principle).approve(address(treasury),_amount) (PRXYUSDCBond.sol#811)
        - ITreasury(treasury).deposit(_amount,principle,profit) (PRXYUSDCBond.sol#812)
        - IERC20(PRXY).safeTransfer(DAO,fee) (PRXYUSDCBond.sol#815)
        Event emitted after the call(s):
        - BondCreated(_amount,payout,block.number.add(terms.vestingTerm),priceInUSD) (PRXYUSDCBond.sol#830)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
ERC20Permit.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (PRXYUSDCBond.sol#436-457) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= deadline,Permit: expired deadline) (PRXYUSDCBond.sol#445)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (PRXYUSDCBond.sol#120-126) uses assembly
        - INLINE ASM (PRXYUSDCBond.sol#124)
Address._functionCallWithValue(address,bytes,uint256,string) (PRXYUSDCBond.sol#157-178) uses assembly
        - INLINE ASM (PRXYUSDCBond.sol#170-173)
Address._verifyCallResult(bool,bytes,string) (PRXYUSDCBond.sol#204-218) uses assembly
        - INLINE ASM (PRXYUSDCBond.sol#210-213)
ERC20Permit.constructor() (PRXYUSDCBond.sol#419-434) uses assembly
        - INLINE ASM (PRXYUSDCBond.sol#421-423)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
INFO:Detectors:
Reentrancy in PRXYUSDCBond.deposit(uint256,address) (PRXYUSDCBond.sol#782-834):
        External calls:
        - IERC20(principle).safeTransferFrom(msg.sender,address(this),_amount) (PRXYUSDCBond.sol#810)
        - IERC20(principle).approve(address(treasury),_amount) (PRXYUSDCBond.sol#811)
        - ITreasury(treasury).deposit(_amount,principle,profit) (PRXYUSDCBond.sol#812)
        - IERC20(PRXY).safeTransfer(DAO,fee) (PRXYUSDCBond.sol#815)
        State variables written after the call(s):
        - totalDebt = totalDebt.add(value) (PRXYUSDCBond.sol#819)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
PRXYUSDCBond.deposit(uint256,address) (PRXYUSDCBond.sol#782-834) ignores return value by IERC20(principle).approve(address(treasury),_amount) (PRXYUSDCBond.so
l#811)
PRXYUSDCBond.deposit(uint256,address) (PRXYUSDCBond.sol#782-834) ignores return value by ITreasury(treasury).deposit(_amount,principle,profit) (PRXYUSDCBond.s
ol#812)
PRXYUSDCBond.stakeOrSend(address,bool,uint256) (PRXYUSDCBond.sol#884-898) ignores return value by IERC20(PRXY).approve(stakingHelper,_amount) (PRXYUSDCBond.so
l#890)
PRXYUSDCBond.stakeOrSend(address,bool,uint256) (PRXYUSDCBond.sol#884-898) ignores return value by IERC20(PRXY).approve(staking,_amount) (PRXYUSDCBond.sol#893)
PRXYUSDCBond.stakeOrSend(address,bool,uint256) (PRXYUSDCBond.sol#884-898) ignores return value by IStaking(staking).stake(_amount,_recipient) (PRXYUSDCBond.so
l#894)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
PRXYUSDCBond.initializeBondTerms(uint256,uint256,uint256,uint256,uint256,uint256) (PRXYUSDCBond.sol#706-724) should emit an event for:
        - totalDebt = _initialDebt (PRXYUSDCBond.sol#722)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
Reentrancy in PRXYUSDCBond.deposit(uint256,address) (PRXYUSDCBond.sol#782-834):
        External calls:
        - IERC20(principle).safeTransferFrom(msg.sender,address(this),_amount) (PRXYUSDCBond.sol#810)
        - IERC20(principle).approve(address(treasury),_amount) (PRXYUSDCBond.sol#811)
        - ITreasury(treasury).deposit(_amount,principle,profit) (PRXYUSDCBond.sol#812)
        - IERC20(PRXY).safeTransfer(DAO,fee) (PRXYUSDCBond.sol#815)
        State variables written after the call(s):
        - bondInfo[_depositor] = Bond(bondInfo[_depositor].payout.add(payout),terms.vestingTerm,block.number,priceInUSD) (PRXYUSDCBond.sol#822-827)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

SOLHINT LINTER

PRXYUSDCBond.sol

2:1	error	Compiler version 0.7.5 does not satisfy the ^0.5.8 semver requirement	compiler-version
22:5	warning	Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity >=0.7.0)	func-visibility
42:9	warning	Error message for require is too long	reason-string
48:9	warning	Error message for require is too long	reason-string
80:9	warning	Error message for require is too long	reason-string
133:9	warning	Error message for require is too long	reason-string
149:9	warning	Error message for require is too long	reason-string
185:9	warning	Error message for require is too long	reason-string
197:9	warning	Error message for require is too long	reason-string
210:17	warning	Avoid to use inline assembly. It is acceptable only in rare cases	no-inline-assembly
222:9	warning	Variable name must be in mixedCase	var-name-mixedcase
225:20	error	Use double quotes for string literals	quotes
226:20	error	Use double quotes for string literals	quotes
277:5	warning	Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity >=0.7.0)	func-visibility
334:9	warning	Error message for require is too long	reason-string
335:9	warning	Error message for require is too long	reason-string
353:9	warning	Error message for require is too long	reason-string
363:9	warning	Error message for require is too long	reason-string
364:9	warning	Error message for require is too long	reason-string
370:97	warning	Code contains empty blocks	no-empty-blocks
417:5	warning	Variable name must be in mixedCase	var-name-mixedcase
419:5	warning	Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity >=0.7.0)	func-visibility
421:9	warning	Avoid to use inline assembly. It is acceptable only in rare cases	no-inline-assembly
445:17	warning	Avoid to make time-based decisions in your business logic	not-rely-on-time
453:9	warning	Error message for require is too long	reason-string
478:9	warning	Error message for require is too long	reason-string
499:13	warning	Error message for require is too long	reason-string
505:66	warning	Avoid to use letters 'I', 'l', 'O' as identifiers	use-forbidden-name
513:9	warning	Avoid to use letters 'I', 'l', 'O' as identifiers	use-forbidden-name
538:10	warning	Avoid to use letters 'I', 'l', 'O' as identifiers	use-forbidden-name
542:24	error	Use double quotes for string literals	quotes
549:5	warning	Contract name must be in CamelCase	contract-name-camelcase
553:5	warning	Contract name must be in CamelCase	contract-name-camelcase
572:9	warning	Error message for require is too long	reason-string
572:34	error	Use double quotes for string literals	quotes
577:45	error	Use double quotes for string literals	quotes
581:45	error	Use double quotes for string literals	quotes
624:5	warning	Variable name must be in mixedCase	var-name-mixedcase
627:5	warning	Variable name must be in mixedCase	var-name-mixedcase
678:5	warning	Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity >=0.7.0)	func-visibility
679:9	warning	Variable name must be in mixedCase	var-name-mixedcase
682:9	warning	Variable name must be in mixedCase	var-name-mixedcase
685:9	warning	Provide an error message for require	reason-string
687:9	warning	Provide an error message for require	reason-string
689:9	warning	Provide an error message for require	reason-string
691:9	warning	Provide an error message for require	reason-string
693:9	warning	Provide an error message for require	reason-string
739:13	warning	Error message for require is too long	reason-string
761:9	warning	Provide an error message for require	reason-string
963:9	warning	Provide an error message for require	reason-string
964:9	warning	Provide an error message for require	reason-string
1044:9	warning	Provide an error message for require	reason-string
1045:9	warning	Provide an error message for require	reason-string

Closing Summary

In this report, we have considered the security of the **PRXY Bond** platform. We performed our audit according to the procedure described above.

The audit showed several medium, low, and informational severity issues. In the end, the majority of the issues were fixed or acknowledged by the Auditee.

Disclaimer

Quillhash audit is not a security warranty, investment advice, or an endorsement of the **PRXY Bond** platform. This audit does not provide a security or correctness guarantee of the audited smart contracts. The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the **PRXY Bond** Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

Audit Report
February, 2022

For

PROXY

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com