Crypton (&Studio)

PV01 OnChain bonds – Smart Contract Audit

Ver. - 1.1

Date of Engagement: 11.12.2023

Contents

Version history

Version	Name	Date
0.1	First draft	15.12.2023
0.2	Internal review	18.12.2023
1.0	Recommendation plan	19.12.2023
1.1	Recommendation execution review	11.01.2024

Executive overview

The security assessment was scoped for the smart contracts of **PV01 OnChain Bonds.** At the time of the audit, all source files were located at the link.

The team at CryptonStudio was provided 7 business days for the engagement and assigned one full time security engineer to audit the security of the smart contracts. The security engineers are blockchain and smart contract security experts, with experience in advanced penetration testing, smart contract hacking, and have a deep knowledge in multiple blockchain protocols.

The purpose of this audit to achieve the following:

- Ensure the smart contracts' functions are intended.

- Identify potential security issues with the smart contracts.

In summary, CryptonStudio identified few security risks, and recommends performing further testing to validate extended safety and correctness in context to the whole set of contracts.

Vulnerabilities or issues observed by CryptonStudio are ranked based on the risk assessment methodology by measuring the likelihood of a security incident, and the impact should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. For every vulnerability, a risk level will be calculated on a scale of 1 to 5 with 5 being the highest likelihood or impact.

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL

Scope

CONTRACTS

- AS - AddressScreen.sol [etherscan.]
- SPBV1 - PV01SinglePaymentBondV1.sol [etherscan.]
- BF - PV01BondFactory.sol [etherscan.]
- SU - StringUtils.sol

Codebase

https://github.com/pv01-org/blockchain

Commit

B9883E7B9861899B166BA5298AE8EE3F76AAFBFB

UNDERSTANDING

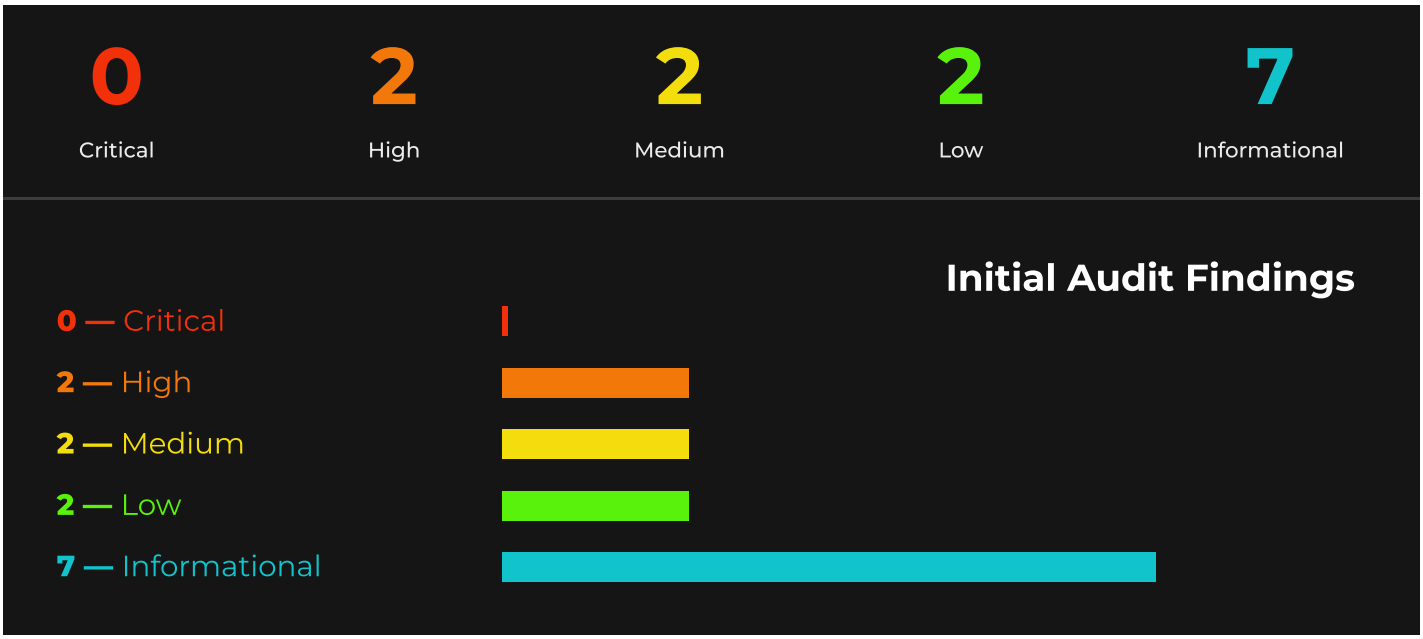
Privileged Functions

In the contracts:

- AddressScreen.sol
- PV01BondFactory.sol
- PV01SinglePaymentBondV1.sol
- StringUtils.sol

The role “owner” has access to privileged functions. For AddressScreen and PV01BondFactory, the owner is basically the admin and the governor that manages these contracts. In the AddressScreen contract, the owner can add or remove addresses to allow or deny lists, which can restrict your ability to use claim() function on bond contract, or transfer bond tokens. In the PV01SinglePaymentBondV1 contract, the owner can use sweep, sweepOther, pause and unpause functions. The first two allow the owner to sweep asset tokens or any other tokens that are held by the contract and send them to the bond issuer's address. Rest of the functions are managing contract state and restricts to call settleBondPurchase, claim, and transfer functions when contract is “paused”. Although the owner can change the maturityDate, it is worth noting that the claim function is not affected by the maturityDate. An investor can claim once funds are available, even if maturityDate is changed.

Assessment summary & findings overview



All issues were addressed either by being fixed or because they were by design. **There are no open issues.**

Issue	Severity	Status	Comment from developer
Gen-1	Informational	Acknowledged	**This is by design.** Longer revert messages cost little extra gas and allow more meaningful messages.
BF-1	Informational	Acknowledged	**This is by design.** This is a small gas saving at a cost of having to manipulate structures potentially leaving the code harder to reason about.
BF-2	Informational	Acknowledged	Since this function is only called by queries, never by transactions costing gas, and will have at most a handful of values, we will leave it as is.
BF-3	Informational	Fixed	Fixed.
SU-1	Informational	Fixed	
SPBV1-1	Low	Acknowledged	**This is by design.**
SPBV1-2	Medium	Acknowledged	**This is by design.**
SPBV1-3	High	Acknowledged	**This is by design.**
SPBV1-4	Informational	Acknowledged	**This is by design.** This is a small gas saving at the cost of having to manipulate structures potentially leaving the code harder to reason about.
SPBV1-5	Low	Fixed	The function has been updated.
SPBV1-6	Informational	Fixed	Function order fixed.
SPBV1-7	High	Fixed	Fixed.
SPBV1-8	Medium	Acknowledged	**This is by design.** It is possible that the maturity date needs to be brought forward.

Findings & tech details

General-1

Severity — Informational

Status — Acknowledged

Description

Most of require statements contain revert strings longer than 32 bytes. It leads to extra gas usage. If message data can fit into 32 bytes, it is always a better idea to use BYTES32 datatype rather than bytes or strings as using BYTES32 datatype is much cheaper in Solidity. Thats a good practice to use custom errors and if->revert statements since they always fit into 32 bytes.

Reference

Recommendation

Check all revert strings and consider using 32 bytes at maximum for revert messages or custom errors.

Alleviation

[PV01-team, 22.12.23]: This is by design. Longer revert messages cost little extra gas and allow more meaningful messages.

BF-1

Severity — Informational

Status — Acknowledged

Description

There is a possibility to optimize BondSpec structure with packing variables in one slot. It can be done with filling type_ in bytes31 type.

Reference

Recommendation

Optimize BondSpec structure and fix logic for it.

Alleviation

[PV01-team, 22.12.23]: This is by design. This is a small gas saving at a cost of having to manipulate structures potentially leaving the code harder to reason about.

BF-2

Severity — Informational

Status — Acknowledged

Description

It has been observed that the for loop in the contract is not optimized. Suboptimal for loops can cost too much gas. These for loops can be optimized with the suggestions above:

1. In Solidity (pragma 0.8.0 and later), adding the unchecked keyword for arithmetical operations, can reduce gas usage on contracts where underflow/overflow is unrealistic. It is possible to save gas by using this keyword in multiple code locations.

2. In all for loops, the index variable is incremented using +=. It is known that, in loops, using ++i costs less gas per iteration than +=. This also affects incremented variables within the loop code block.

Recommendation

It is recommended to apply the following pattern for Solidity pragma version 0.8.0 and later.

```
for (uint256 i = 0; i < numberOfIterations; ) {
  ...
  unchecked {
    ++i;
  }
}
```

Alleviation

[PV01-team, 22.12.23]: Since this function is only called by queries, never by transactions costing gas, and will have at most a handful of values, we will leave it as is.

BF-3

Severity — Informational

Status — Fixed

Description

Comment line for mapping bondsToSpec doesn't describe its purpose.

Recommendation

Revisit comment to this variable.

Alleviation

[PV01-team, 22.12.23]: Fixed.

SU-1

Severity — Informational

Status — Fixed

Description

No reason to allocate string_ to memory in _stringToBytes32 function, in context of given contracts.

When a function parameter is a string that will not be changed, it should ideally be allocated in calldata rather than memory.

Calldata is a non-modifiable, non-persistent area where function arguments are stored, and it costs less gas to access. Parameters stored in calldata do not need to be copied to memory, thus saving gas when the function is called externally.

Recommendation

Replace memory to calldata for string_ variable in _stringToBytes32() function.

Alleviation

[PV01-team, 09.01.24]: Function _stringToBytes32 uses calldata now.

SPBV1-1

Severity — Low

Status — Acknowledged

Description

settleBondPurchase function missing onlyAllowListed modifier. It can lead to non-whitelisted users, which address somehow contained in the funding commitments array, to use this function.

Recommendation

Add onlyAllowListed modifier according to comment to function.

Alleviation

[PV01-team, 22.12.23]: This is by design. Function settleBondPurchase can only succeed if a funding commitment is present with an address that matches the address trying to settle.

A funding commitment can only be present if the address has already passed KYC/AML checks done on the platform (outside of this bond contract).

The onlyAllowListed modifier would not bring anything extra to this settleBondPurchase. The modifier is intended to be used during transfers and is fairly gas expensive.

SPBV1-2

Severity — Medium

Status — Acknowledged

Description

There are no guarantees that the bond contract will be fulfilled with tokens to claim. It can lead to unavailability to claim any tokens and to loss of all investors funds.

Recommendation

Make sure that contract is fulfilled with tokens to claim.

Alleviation

[PV01-team, 22.12.23]: This is true, it is by design. A bond default as described would be an off-chain legal process.

We implemented a mechanism that only allows claims to be made if there are enough asset tokens on the bond contract to repay EVERY investor before paying ANY investor.

SPBV1-3

Severity — High

Status — Acknowledged

Description

Centralization functionality risk. There is a possibility to sweep all bond contract funds right after the maturity date + sweep delay period, without waiting for users to claim their funds. In case, when the user was unable to claim for that period, he will lose his funds.

Recommendation

Set the value of sweeping tokens to:

```
transferAmount = assetToken.balanceOf(address(this)) -
        _calculateAmount(totalSupply_, interestRateFraction);
```

It should help to prevent loss of user's funds.

Alleviation (SPBV1-3)

[PV01-team, 22.12.23]: This is true, it is by design. The reason we cannot use the code you suggested is as follows:

An investor can choose to trade their bond token as an ERC20 on the secondary market. A subsequent bond token holder (not the investor) may be deny listed, and we don't want them to be able to claim, ever. Legally we're not allowed to give that deny-listed bond token holder any asset tokens.

In this case, the asset tokens due for them should be leftover on the bond, and should be returned to the bond issuer after all other claims are completed.

SPBV1-4

Severity — Informational

Status — Acknowledged

Description

There is a possibility to optimize FundingCommitmentInternal structure with packing variables in two slots.

Recommendation

Optimize FundingCommitmentInternal structure and fix logic for it.

Alleviation

[PV01-team, 22.12.23]: This is by design. This is a small gas saving at a cost of having to manipulate structures potentially leaving the code harder to reason about.

SPBV1-5

Severity — Low

Status — Fixed

Description

isRepaidInFull() function returns false, if totalSupply == 0. In case that all bonds were burnt, this is incorrect, according to comment line of this function: `@return hasBeenRepaid bool: did the bond issuer pay back the debt in full?`

Recommendation

Revisit comment or logic of this function.

Alleviation

[PV01-team, 22.12.23]: The function has been updated.

SPBV1-6

Severity — Informational

Status — Fixed

Description

Function order in the contract does not meet solidity style-guide standards.

According to Solidity Style Guide, function must be presented in the next order:

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

Within a grouping, place the view and pure functions last.

Recommendation

Some functions that are not in order:

- receive()
- decimals()
- pause()
- unpause()
- getAmountToRepay()
- etc.

Revisit function order in the PV01SinglePaymentBondV1.sol contract.

Alleviation

[PV01-team, 22.12.23]: Function order fixed.

SPBV1-7

Severity — High

Status — Fixed

Description

Centralization functionality risk. Owner Functions:

- setAssetValueMultiplier()
- setAssetTokenContract()

Can change the corresponding values, if there are no pending funding commitments. In case all funding commitments proceed, the owner can change any of these values, before users claim. This can lead to a significant amount of loss to users.

Also there is an exploit for the sweep() function. If the owner changes the asset token, as described above, he will get access to transfer “ex” asset tokens via function sweepOther(), which hasn’t restriction for extra sweep period like in sweep() function.

Recommendation

Revamp logic of accessibility to these functions.

Alleviation

[PV01-team, 22.12.23]: Decided to remove setAssetValueMultiplier(). We are removing this function and the asset value multiplier field completely, it won't be present as a variable or in calculations.

Decided to remove setAssetTokenContract(). We are removing this function, the asset token contract will be fixed on the bond contract at bond creation time.

After completing the above, the sweepOther() function will then only ever be able to sweep tokens that are not the main asset token.

SPBV1-8

Severity — Medium

Status — Acknowledged

Description

Centralization functionality risk. It has been observed that there is a possibility to set scheduledMaturityDate() before actual maturity date. It can lead to misunderstandings by users and then to possibly lose their funds, because they didn’t claim in time.

Recommendation

We are suggesting to add restriction:

```
require(newDate > scheduledMaturityDate, "scheduledMaturityDate must be later than the previous");
```

Alleviation

[PV01-team, 22.12.23]: This is by design. It is possible that the maturity date needs to be brought forward. There is a minimum 3 day delay between maturity and sweep date. Maturity date cannot be moved into the past.

[Crypton.studio, 09.01.24]: Function renamed to setMaturityDate().

STATIC ANALYSIS SCAN

Description

Crypton.studio used automated testing techniques to enhance coverage of certain areas of the scoped contracts. Slither, a Solidity static analysis framework, was used for static analysis. After Crypton.studio verified all the contracts in the repository and was able to compile them correctly into their ABI and binary formats. This tool can statically verify mathematical relationships between Solidity variables to detect invalid or inconsistent usage of the contracts' APIs across the entire code-base.

After static analysis no audit issues were detected, no remedial actions are required.

RESULTS

INFO:Detectors:
PV01BondFactory.setImplAddress(string,address) (contracts/PV01BondFactory.sol#73-82) ignores return value by types.add(typeBytes32) (contracts/PV01BondFactory.sol#77)
PV01BondFactory.createBond(string,uint8,bytes) (contracts/PV01BondFactory.sol#91-107) ignores return value by Address.functionCall(contractAddress,data_) (contracts/PV01BondFactory.sol#102)
PV01BondFactory.createBond(string,uint8,bytes) (contracts/PV01BondFactory.sol#91-107) ignores return value by bonds[typeBytes32].add(contractAddress) (contracts/PV01BondFactory.sol#104)
PV01SinglePaymentBondV1.settleBondPurchase() (contracts/bonds/SinglePaymentBond/V1/PV01SinglePaymentBondV1.sol#660-678) ignores return value by addressWithFundingCommitment.remove(msg.sender) (contracts/bonds/SinglePaymentBond/V1/PV01SinglePaymentBondV1.sol#670)

INFO:Detectors:
PV01BondFactory.constructor(address)._owner (contracts/PV01BondFactory.sol#64) shadows:
- Ownable._owner (../../node_modules/@openzeppelin/contracts/access/Ownable.sol#21) (state variable)

INFO:Detectors:
Reentrancy in PV01BondFactory.createBond(string,uint8,bytes) (contracts/PV01BondFactory.sol#91-107):
External calls:
- Address.functionCall(contractAddress,data_) (contracts/PV01BondFactory.sol#102)
State variables written after the call(s):
- bondsToSpec[contractAddress] = BondSpec(typeBytes32,version_) (contracts/PV01BondFactory.sol#105)
Reentrancy in PV01SinglePaymentBondV1.settleBondPurchase() (contracts/bonds/SinglePaymentBond/V1/PV01SinglePaymentBondV1.sol#660-678):
External calls:
- assetToken.safeTransferFrom(msg.sender,bondIssuer,proceed) (contracts/bonds/SinglePaymentBond/V1/PV01SinglePaymentBondV1.sol#674)
State variables written after the call(s):
- _mint(msg.sender,bondTokenAmount) (contracts/bonds/SinglePaymentBond/V1/PV01SinglePaymentBondV1.sol#675)
- _balances[account] += amount (../../node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#272)
- _mint(msg.sender,bondTokenAmount) (contracts/bonds/SinglePaymentBond/V1/PV01SinglePaymentBondV1.sol#675)
- _totalSupply += amount (../../node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#269)

INFO:Detectors:
Reentrancy in PV01SinglePaymentBondV1.claim() (contracts/bonds/SinglePaymentBond/V1/PV01SinglePaymentBondV1.sol#600-624):
External calls:
- assetToken.safeTransfer(msg.sender,assetTokenAmount) (contracts/bonds/SinglePaymentBond/V1/PV01SinglePaymentBondV1.sol#617)
Event emitted after the call(s):
- Claim(msg.sender,bondTokenAmount,assetTokenAmount) (contracts/bonds/SinglePaymentBond/V1/PV01SinglePaymentBondV1.sol#619)
- Redeemed() (contracts/bonds/SinglePaymentBond/V1/PV01SinglePaymentBondV1.sol#622)
Reentrancy in PV01BondFactory.createBond(string,uint8,bytes) (contracts/PV01BondFactory.sol#91-107):
External calls:
- Address.functionCall(contractAddress,data_) (contracts/PV01BondFactory.sol#102)
Event emitted after the call(s):
- NewBond(typeBytes32,version_,contractAddress) (contracts/PV01BondFactory.sol#106)
Reentrancy in PV01SinglePaymentBondV1.settleBondPurchase() (contracts/bonds/SinglePaymentBond/V1/PV01SinglePaymentBondV1.sol#660-678):
External calls:
- assetToken.safeTransferFrom(msg.sender,bondIssuer,proceed) (contracts/bonds/SinglePaymentBond/V1/PV01SinglePaymentBondV1.sol#674)
Event emitted after the call(s):
- SettleBondPurchase(msg.sender,bondTokenAmount,proceed) (contracts/bonds/SinglePaymentBond/V1/PV01SinglePaymentBondV1.sol#677)
- Transfer(address(0),account,amount) (../../node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#274)
- _mint(msg.sender,bondTokenAmount) (contracts/bonds/SinglePaymentBond/V1/PV01SinglePaymentBondV1.sol#675)
Reentrancy in PV01SinglePaymentBondV1.sweep() (contracts/bonds/SinglePaymentBond/V1/PV01SinglePaymentBondV1.sol#685-694):
External calls:
- assetToken.safeTransfer(bondIssuer,transferAmount) (contracts/bonds/SinglePaymentBond/V1/PV01SinglePaymentBondV1.sol#692)
Event emitted after the call(s):
- Sweep(bondIssuer,transferAmount) (contracts/bonds/SinglePaymentBond/V1/PV01SinglePaymentBondV1.sol#693)
Reentrancy in PV01SinglePaymentBondV1.sweepOther(address) (contracts/bonds/SinglePaymentBond/V1/PV01SinglePaymentBondV1.sol#701-708):
External calls:
- otherToken.safeTransfer(bondIssuer,transferAmount) (contracts/bonds/SinglePaymentBond/V1/PV01SinglePaymentBondV1.sol#706)
Event emitted after the call(s):
- SweepOther(bondIssuer,otherTokenAddress,transferAmount) (contracts/bonds/SinglePaymentBond/V1/PV01SinglePaymentBondV1.sol#707)

INFO:Detectors:
PV01SinglePaymentBondV1.settleBondPurchase() (contracts/bonds/SinglePaymentBond/V1/PV01SinglePaymentBondV1.sol#660-678) uses timestamp for comparisons
Dangerous comparisons:
- require(bool,string)(block.timestamp <= date_,Funding commitment date already passed) (contracts/bonds/SinglePaymentBond/V1/PV01SinglePaymentBondV1.sol#666)
- require(bool,string)(block.timestamp > date_ - (86400),Timeout period has not finished yet) (contracts/bonds/SinglePaymentBond/V1/PV01SinglePaymentBondV1.sol#667)
PV01SinglePaymentBondV1.sweep() (contracts/bonds/SinglePaymentBond/V1/PV01SinglePaymentBondV1.sol#685-694) uses timestamp for comparisons

Dangerous comparisons:
- require(bool,string)(block.timestamp >= (maturityDate + sweepDelaySecondsFromMaturityDate),Sweep date not yet passed) (contracts/bonds/SinglePaymentBond/V1/PV01SinglePaymentBondV1.sol#686-689)
PV01SinglePaymentBondV1._setMaturityDate(uint64,string) (contracts/bonds/SinglePaymentBond/V1/PV01SinglePaymentBondV1.sol#830-838) uses timestamp for comparisons
Dangerous comparisons:
- require(bool,string)(newDate > block.timestamp,maturityDate must be in the future) (contracts/bonds/SinglePaymentBond/V1/PV01SinglePaymentBondV1.sol#831)

INFO:Detectors:
AddressUpgradeable._revert(bytes,string) (../../node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#206-218) uses assembly
- INLINE ASM (../../node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#211-214)
Clones.clone(address) (../../node_modules/@openzeppelin/contracts/proxy/Clones.sol#25-34) uses assembly
- INLINE ASM (../../node_modules/@openzeppelin/contracts/proxy/Clones.sol#26-32)
Clones.cloneDeterministic(address,bytes32) (../../node_modules/@openzeppelin/contracts/proxy/Clones.sol#43-52) uses assembly
- INLINE ASM (../../node_modules/@openzeppelin/contracts/proxy/Clones.sol#44-50)
Clones.predictDeterministicAddress(address,bytes32,address) (../../node_modules/@openzeppelin/contracts/proxy/Clones.sol#57-72) uses assembly
- INLINE ASM (../../node_modules/@openzeppelin/contracts/proxy/Clones.sol#62-71)
Address.isContract(address) (../../node_modules/@openzeppelin/contracts/utils/Address.sol#27-37) uses assembly
- INLINE ASM (../../node_modules/@openzeppelin/contracts/utils/Address.sol#33-35)
Address.verifyCallResult(bool,bytes,string) (../../node_modules/@openzeppelin/contracts/utils/Address.sol#196-216) uses assembly
- INLINE ASM (../../node_modules/@openzeppelin/contracts/utils/Address.sol#208-211)
EnumerableSet.values(EnumerableSet.AddressSet) (../../node_modules/@openzeppelin/contracts/utils/structs/EnumerableSet.sol#274-283) uses assembly
- INLINE ASM (../../node_modules/@openzeppelin/contracts/utils/structs/EnumerableSet.sol#278-280)
EnumerableSet.values(EnumerableSet.UintSet) (../../node_modules/@openzeppelin/contracts/utils/structs/EnumerableSet.sol#347-356) uses assembly
- INLINE ASM (../../node_modules/@openzeppelin/contracts/utils/structs/EnumerableSet.sol#351-353)
StringUtils._stringToBytes32(string) (contracts/libs/StringUtils.sol#14-25) uses assembly
- INLINE ASM (contracts/libs/StringUtils.sol#21-23)

INFO:Detectors:
Different versions of Solidity are used:
- Version used: ['0.8.15', '^0.8.0', '^0.8.1', '^0.8.2']
- 0.8.15 (contracts/AddressScreen.sol#2)
- 0.8.15 (contracts/PV01BondFactory.sol#2)
- 0.8.15 (contracts/bonds/SinglePaymentBond/V1/PV01SinglePaymentBondV1.sol#8)
- 0.8.15 (contracts/interfaces/IAddressScreen.sol#2)
- 0.8.15 (contracts/interfaces/IERC20Standard.sol#2)
- 0.8.15 (contracts/interfaces/ISanctionsList.sol#3)
- 0.8.15 (contracts/libs/PreciseUnitMath.sol#2)
- 0.8.15 (contracts/libs/StringUtils.sol#2)
- ^0.8.0 (../../node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#4)
- ^0.8.0 (../../node_modules/@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#4)
- ^0.8.0 (../../node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#4)
- ^0.8.0 (../../node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol#4)
- ^0.8.0 (../../node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/extensions/IERC20MetadataUpgradeable.sol#4)
- ^0.8.0 (../../node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#4)
- ^0.8.0 (../../node_modules/@openzeppelin/contracts/access/Ownable.sol#4)
- ^0.8.0 (../../node_modules/@openzeppelin/contracts/proxy/Clones.sol#4)
- ^0.8.0 (../../node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#4)

- ^0.8.0 (../../node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4)
- ^0.8.0 (../../node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#4)
- ^0.8.0 (../../node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#4)
- ^0.8.0 (../../node_modules/@openzeppelin/contracts/utils/Address.sol#4)
- ^0.8.0 (../../node_modules/@openzeppelin/contracts/utils/Context.sol#4)
- ^0.8.0 (../../node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#4)
- ^0.8.0 (../../node_modules/@openzeppelin/contracts/utils/structs/EnumerableSet.sol#4)
- ^0.8.1 (../../node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#4)
- ^0.8.2 (../../node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#4)

INFO:Detectors:
PV01SinglePaymentBondV1._removeFundingCommitment(address) (contracts/bonds/SinglePaymentBond/V1/PV01SinglePaymentBondV1.sol#800-807) has costly operations inside a loop:
- delete fundingCommitments[userAddress] (contracts/bonds/SinglePaymentBond/V1/PV01SinglePaymentBondV1.sol#805)

INFO:Detectors:
PreciseUnitMath.abs(int256) (contracts/libs/PreciseUnitMath.sol#186-188) is never used and should be removed
PreciseUnitMath.approximatelyEquals(uint256,uint256,uint256) (contracts/libs/PreciseUnitMath.sol#179-181) is never used and should be removed
PreciseUnitMath.conservativeDiv(int256,int256) (contracts/libs/PreciseUnitMath.sol#156-158) is never used and should be removed
PreciseUnitMath.conservativeMul(int256,int256) (contracts/libs/PreciseUnitMath.sol#148-150) is never used and should be removed
PreciseUnitMath.div(int256,int256) (contracts/libs/PreciseUnitMath.sol#98-100) is never used and should be removed
PreciseUnitMath.div(uint256,uint256) (contracts/libs/PreciseUnitMath.sol#91-93) is never used and should be removed
PreciseUnitMath.divCeil(int256,int256) (contracts/libs/PreciseUnitMath.sol#115-127) is never used and should be removed
PreciseUnitMath.divCeil(uint256,uint256) (contracts/libs/PreciseUnitMath.sol#105-109) is never used and should be removed
PreciseUnitMath.divDown(int256,int256) (contracts/libs/PreciseUnitMath.sol#132-142) is never used and should be removed
PreciseUnitMath.maxInt256() (contracts/libs/PreciseUnitMath.sol#50-52) is never used and should be removed
PreciseUnitMath.maxUint256() (contracts/libs/PreciseUnitMath.sol#43-45) is never used and should be removed
PreciseUnitMath.minInt256() (contracts/libs/PreciseUnitMath.sol#57-59) is never used and should be removed
PreciseUnitMath.mul(int256,int256) (contracts/libs/PreciseUnitMath.sol#73-75) is never used and should be removed
PreciseUnitMath.mulCeil(uint256,uint256) (contracts/libs/PreciseUnitMath.sol#81-86) is never used and should be removed
PreciseUnitMath.neg(int256) (contracts/libs/PreciseUnitMath.sol#193-196) is never used and should be removed
PreciseUnitMath.preciseUnit() (contracts/libs/PreciseUnitMath.sol#29-31) is never used and should be removed
PreciseUnitMath.preciseUnitInt() (contracts/libs/PreciseUnitMath.sol#36-38) is never used and should be removed
PreciseUnitMath.safePower(uint256,uint256) (contracts/libs/PreciseUnitMath.sol#163-174) is never used and should be removed

INFO:Detectors:
Pragma version^0.8.0 (../../node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#4) allows old versions
Pragma version^0.8.2 (../../node_modules/@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#4) allows old versions
Pragma version^0.8.0 (../../node_modules/@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#4) allows old versions

Pragma version^0.8.0 (../../node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#4) allows old versions
Pragma version^0.8.0 (../../node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol#4) allows old versions
Pragma version^0.8.0 (../../node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/extensions/IERC20MetadataUpgradeable.sol#4) allows old versions
Pragma version^0.8.1 (../../node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (../../node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (../../node_modules/@openzeppelin/contracts/access/Ownable.sol#4) allows old versions
Pragma version^0.8.0 (../../node_modules/@openzeppelin/contracts/proxy/Clones.sol#4) allows old versions
Pragma version^0.8.0 (../../node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#4) allows old versions
Pragma version^0.8.0 (../../node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#4) allows old versions
Pragma version^0.8.0 (../../node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#4) allows old versions
Pragma version^0.8.0 (../../node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#4) allows old versions
Pragma version^0.8.0 (../../node_modules/@openzeppelin/contracts/utils/Address.sol#4) allows old versions
Pragma version^0.8.0 (../../node_modules/@openzeppelin/contracts/utils/Context.sol#4) allows old versions
Pragma version^0.8.0 (../../node_modules/@openzeppelin/contracts/utils/math/SafeCast.sol#4) allows old versions
Pragma version^0.8.0 (../../node_modules/@openzeppelin/contracts/utils/structs/EnumerableSet.sol#4) allows old versions
Pragma version0.8.15 (contracts/AddressScreen.sol#2) allows old versions
Pragma version0.8.15 (contracts/PV01BondFactory.sol#2) allows old versions
Pragma version0.8.15 (contracts/bonds/SinglePaymentBond/V1/PV01SinglePaymentBondV1.sol#8) allows old versions
Pragma version0.8.15 (contracts/interfaces/IAddressScreen.sol#2) allows old versions
Pragma version0.8.15 (contracts/interfaces/IERC20Standard.sol#2) allows old versions
Pragma version0.8.15 (contracts/interfaces/ISanctionsList.sol#3) allows old versions
Pragma version0.8.15 (contracts/libs/PreciseUnitMath.sol#2) allows old versions
Pragma version0.8.15 (contracts/libs/StringUtils.sol#2) allows old versions
solc-0.8.15 is not recommended for deployment

INFO:Detectors:
Low level call in AddressUpgradeable.sendValue(address,uint256) (../../node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#60-65):
- (success) = recipient.call{value: amount}() (../../node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#63)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (../../node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#128-137):
- (success,returndata) = target.call{value: value}(data) (../../node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#135)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (../../node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#155-162):
- (success,returndata) = target.staticcall(data) (../../node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#160)
Low level call in Address.sendValue(address,uint256) (../../node_modules/@openzeppelin/contracts/utils/Address.sol#55-60):
- (success) = recipient.call{value: amount}() (../../node_modules/@openzeppelin/contracts/utils/Address.sol#58)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (../../node_modules/@openzeppelin/contracts/utils/Address.sol#123-134):

- (success,returndata) = target.call{value: value}(data) (../../node_modules/@openzeppelin/contracts/utils/Address.sol#132)
Low level call in Address.functionStaticCall(address,bytes,string) (../../node_modules/@openzeppelin/contracts/utils/Address.sol#152-161):
- (success,returndata) = target.staticcall(data) (../../node_modules/@openzeppelin/contracts/utils/Address.sol#159)
Low level call in Address.functionDelegateCall(address,bytes,string) (../../node_modules/@openzeppelin/contracts/utils/Address.sol#179-188):
- (success,returndata) = target.delegatecall(data) (../../node_modules/@openzeppelin/contracts/utils/Address.sol#186)

INFO:Detectors:
Function OwnableUpgradeable.__Ownable_init() (../../node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#29-31) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init_unchained() (../../node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#33-35) is not in mixedCase
Variable OwnableUpgradeable.__gap (../../node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#94) is not in mixedCase
Function PausableUpgradeable.__Pausable_init() (../../node_modules/@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#34-36) is not in mixedCase
Function PausableUpgradeable.__Pausable_init_unchained() (../../node_modules/@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#38-40) is not in mixedCase
Variable PausableUpgradeable.__gap (../../node_modules/@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol#116) is not in mixedCase
Function ERC20Upgradeable.__ERC20_init(string,string) (../../node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#55-57) is not in mixedCase
Function ERC20Upgradeable.__ERC20_init_unchained(string,string) (../../node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#59-62) is not in mixedCase
Variable ERC20Upgradeable.__gap (../../node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#400) is not in mixedCase
Function ContextUpgradeable.__Context_init() (../../node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#18-19) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (../../node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#21-22) is not in mixedCase
Variable ContextUpgradeable.__gap (../../node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#36) is not in mixedCase

INFO:Detectors:
Variable PV01BondFactory.setImplAddress(string,address).implementation_ (contracts/PV01BondFactory.sol#73) is too similar to PV01BondFactory.implementations (contracts/PV01BondFactory.sol#42)
Variable PV01SinglePaymentBondV1._calculateAmount(uint256,uint256)._interestRateFraction (contracts/bonds/SinglePaymentBond/V1/PV01SinglePaymentBondV1.sol#778) is too similar to

INFO:Detectors:
PV01SinglePaymentBondV1.initialize(string,string,PV01SinglePaymentBondV1.FundingCommitmentExternal[],address,uint64,uint64,address,address,uint64,address,string).interestRateFraction_ (contracts/bonds/SinglePaymentBond/V1/PV01SinglePaymentBondV1.sol#474)

INFO:Detectors:
Clones.clone(address) (../../node_modules/@openzeppelin/contracts/proxy/Clones.sol#25-34) uses literals with too many digits:

- mstore(uint256,uint256)
(ptr_clone_asm_0,0x3d602d80600a3d3981f3363d3d373d3d3d363d730000000000000000000000000000) (../../
node_modules/@openzeppelin/contracts/proxy/Clones.sol#28)
Clones.clone(address) (../../node_modules/@openzeppelin/contracts/proxy/Clones.sol#25-34) uses literals with too
many digits:
- mstore(uint256,uint256)(ptr_clone_asm_0 +
0x28,0x5af43d82803e903d91602b57fd5bf3000000000000000000000000000000000000) (../../node_modules/
@openzeppelin/contracts/proxy/Clones.sol#30)
Clones.cloneDeterministic(address,bytes32) (../../node_modules/@openzeppelin/contracts/proxy/Clones.sol#43-52)
uses literals with too many digits:
- mstore(uint256,uint256)
(ptr_cloneDeterministic_asm_0,0x3d602d80600a3d3981f3363d3d373d3d3d363d730000000000000000000000000000)
(../../node_modules/@openzeppelin/contracts/proxy/Clones.sol#46)
Clones.cloneDeterministic(address,bytes32) (../../node_modules/@openzeppelin/contracts/proxy/Clones.sol#43-52)
uses literals with too many digits:
- mstore(uint256,uint256)(ptr_cloneDeterministic_asm_0 +
0x28,0x5af43d82803e903d91602b57fd5bf3000000000000000000000000000000000000) (../../node_modules/
@openzeppelin/contracts/proxy/Clones.sol#48)
Clones.predictDeterministicAddress(address,bytes32,address) (../../node_modules/@openzeppelin/contracts/proxy/
Clones.sol#57-72) uses literals with too many digits:
- mstore(uint256,uint256)
(ptr_predictDeterministicAddress_asm_0,0x3d602d80600a3d3981f3363d3d373d3d3d363d7300000000000000000000
00000000) (../../node_modules/@openzeppelin/contracts/proxy/Clones.sol#64)
Clones.predictDeterministicAddress(address,bytes32,address) (../../node_modules/@openzeppelin/contracts/proxy/
Clones.sol#57-72) uses literals with too many digits:
- mstore(uint256,uint256)(ptr_predictDeterministicAddress_asm_0 +
0x28,0x5af43d82803e903d91602b57fd5bf3ff0000000000000000000000000000000000) (../../node_modules/
@openzeppelin/contracts/proxy/Clones.sol#66)

INFO: Slither:. analyzed (29 contracts with 85 detectors), 108 result(s) found

Contacts

crypton.studio

info@crypton.studio

+371 261 19 169

OUR MEDIA

Scan the QR code to open our sources