SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: UNN

Date: 09/05/21

Platform: EVM

Language: Solidity

This document may contain confidential information about IT systems and the customer's intellectual property and information about potential vulnerabilities and exploitation methods.
The report contains confidential information. This information can be used internally by the customer. The customer can release the information after fixing all vulnerabilities.

Document

Name	UNN SC Protection
Platform	EVM
Date	09/05/21

Table of contents

Introduction

This report presents the Customer` s smart contract's security assessment findings and its code review conducted between April 27 2021 – May 9, 2021

Scope

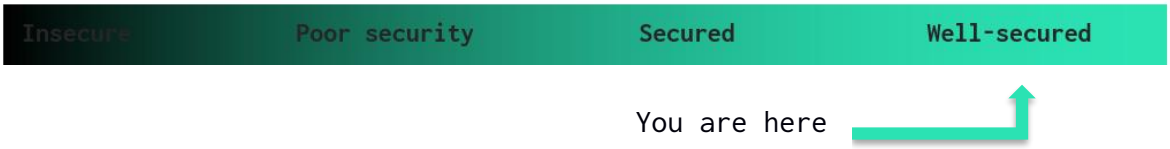
The scope of the project is UNN SC Protections smart contract.

We have scanned this smart contract for commonly known and more specific vulnerabilities. Here are some of the widely known vulnerabilities that are considered (the full list includes them but does not limit by them):

- Reentrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with (Unexpected) Throw
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Style guide violation
- Transfer forwards all gas
- ERC20 API violation
- Compiler version not fixed
- Unchecked external call – Unchecked math
- Unsafe type inference
- Implicit visibility level

Executive Summary

According to the assessment, Customer' smart contracts are well-secured.



Our team performed an analysis of code functionality, manual audit, and automated checks with Slither and remix IDE (see Appendix B pic 1-3). All issues found during automated analysis reviewed have been manually, and application vulnerabilities are presented in the Audit overview section. A general overview is presented in the AS-IS section, and all found issues can be found in the Audit overview section.

We found three low issues in a smart contract that couldn't have any significant security impact.

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to tokens loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also significantly do not impact smart contract execution, e.g., public access to crucial functions.
Medium	Medium-level vulnerabilities are essential to fix; however, they can't lead to tokens loss.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc., code snippets that can't significantly impact execution.
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations, and info statements can't affect smart contract execution and can be ignored.

AS-IS overview

Union Protocol contract consists of the next smart contracts:

1. **SCPClaims.sol, SCProtections.sol, SignLib.sol** contracts – Union Protocol Classes
2. **Initializable.sol,AccessControlUpgradeable.sol, PausableUpgradeable.sol,SafeMathUpgradeable.sol, IERC20Upgradeable.sol, SafeERC20Upgradeable.sol** contracts – Openzeppelin
3. **ISCProtections.sol,ISCPClaims.sol,ISCPool.sol, IUUNNRegistry.sol** contracts – interfaces

Contracts from point 1 will be detailed in the report.

Contracts from point 2 were compared to original "Openzeppelin" templates no logic differences were found. They are considered as secure.

Contracts from point 3 are interfaces that include header files.

AS-IS SCPClaims overview

SCPClaims.sol contract inherits the class Initializable, AccessControlUpgradeable, PausableUpgradeable and ISCPClaims.

SCPClaims.sol contract **init** functions:

initialize function was called with following parameters:

- address(_admin)
- address(_scProtections)

setClaimingParams function was called with following parameters:

- uint64(_createPaymentAmt)
- uint64(_duplicationMultiplier)

setProtectionPayoutPeriod function was called with following parameters:

- uint64(_protectionPayoutPeriod)

fillClaim function was called with following parameters:

- bytes32(_ppID)
- uint64(_timestamp)

fillClaimForPool function was called with following parameters:

- address(_poolAddress)
- uint64(_timestamp)

claimFeeRefund function was called with following parameters:

- bytes32(_ppID)
- uint64(_timestamp)

setClaimInReview function was called with following parameters:

- bytes32(_ppID)
- uint64(_timestamp)

setClaimApproved function was called with following parameters:

- bytes32(_ppID)
- uint64(_timestamp)
- uint8(_payAmountPercentage)

setClaimRejected function was called with following parameters:

- bytes32(_ppID)
- uint64(_timestamp)

releaseLockOnExpiredClaim function was called with following parameters:

- bytes32(_ppID)
- uint64(_timestamp)

getNextClaimFillPayment function was called with following parameters:

- bytes32(_ppID)
- uint64(_timestamp)

_fillClaim function was called with following parameters:

- ISCPool(_pool)
- uint64(_timestamp)

_nextClaimFillPayment function was called with following parameters:

- bytes32(_ppID)
- uint64(_timestamp)
- ISCPool(_pool)

getClaimData function was called with following parameters:

- bytes32(_ppID)
- uint64(_timestamp)

version function was called without parameters.

pause function was called without parameters.

unpause function was called without parameters.

getClaimingParams function was called without parameters.

Audit overview

Critical

No critical severity vulnerabilities were found.

High

No high severity vulnerabilities were found.

Medium

No medium severity vulnerabilities were found.

Low

Different versions of Solidity are used in Version used: ['0.6.12', >=0.6.12, '>=0.6.2<0.8.0', '^0.6.12'] (see Appendix A pic. 1 for evidence)

AS-IS SCProtections overview

SCProtections.sol contract inherits the class Initializable, AccessControlUpgradeable, PausableUpgradeable, SignLib and ISCProtections.

SCProtections.sol contract **init** functions:

initialize function was called with following parameters:

- address(_admin)
- address(_uunn)
- address(_claimStorage)

create function was called with following parameters:

- uint256[9] memory(data)
- bytes memory(signature)
- uint256(deadline)

createTo function was called with following parameters:

- uint256[9] memory(data)
- bytes memory(signature)
- address(erc721Receiver)
- uint256(deadline)

withdrawPremium function was called with following parameters:

- uint256(_id)
- uint256(_premium)

setActiveSCProtectionPoolAddress function was called with following parameters:

- bytes32(_ppID)
- address(_value)

getActiveSCProtectionPool function was called with following parameters:

- bytes32(_ppID)

exercise function was called with following parameters:

- uint256(_id)
- uint64(_timestamp)

setClaimLock function was called with following parameters:

- address(_pool)
- uint64(_timestamp)

releaseClaimLock function was called with following parameters:

- address(_pool)
- uint64(_timestamp)

isClaimLocked function was called with following parameters:

- address(_pool)

getProtectionData function was called with following parameters:

- uint256(id)

setDocument function was called with following parameters:

- uint256(id)
- bytes32(name)
- string calldata(uri)
- bytes32(documentHash)

removeDocument function was called with following parameters:

- uint256(id)
- bytes32(name)

getDocument function was called with following parameters:

- uint256(id)
- bytes32(_name)

getAllDocuments function was called with following parameters:

- uint64(id)

version function was called without parameters.

pause function was called without parameters.

unpause function was called without parameters.

Audit overview

Critical

No critical severity vulnerabilities were found.

High

No high severity vulnerabilities were found.

Medium

No medium severity vulnerabilities were found.

Low

Different versions of Solidity are used in Version used: ['0.6.12', >=0.6.12, '>=0.6.2<0.8.0', '^0.6.12'] (see Appendix A pic. 2 for evidence)

AS-IS UnionSCPool overview

UnionSCPool.sol contract inherits the class UnionERC20Pool and ISCPool.

UnionSCPool.sol contract **init** functions:

getWriterDataExtended function was called with following parameters:

- address(_writer)

initialize function was called with following parameters:

- address(admin)
- address(_basicToken)
- bytes32(_ppID)
- address(_scProtectionStorage)
- string memory(_description)

setLockupPeriod function was called with following parameters:

- uint256(value)

onProtectionPremium function was called with following parameters:

- address(buyer)
- uint256[7] memory(data)

unlockPremium function was called with following parameters:

- uint256[] calldata(_ids)

onPayoutCoverage function was called with following parameters:

- uint256(_id)
- uint256(_premiumToUnlock)
- uint256(_coverageToPay)
- address(_beneficiary)

withdrawWithData function was called with following parameters:

- uint256(_requestID)

- uint256(_amount)
- uint256[5] memory(_data)
- bytes memory(_signature)

_beforeTokenTransfer function was called with following parameters:

- address(from)
- address()
- uint256()

_afterDeposit function was called with following parameters:

- uint256(amountTokenSent)
- uint256(amountLiquidityGot)
- address(sender)
- address(holder)

version function was called without parameters.

poolType function was called without parameters.

onlySCProtection function was called without parameters.

ppID function was called without parameters.

Audit overview

Critical

No critical severity vulnerabilities were found.

High

No high severity vulnerabilities were found.

Medium

No medium severity vulnerabilities were found.

Low

Different versions of Solidity are used in Version used: ['0.6.12', >=0.6.12, '>=0.6.2<0.8.0', '^0.6.12'] (see Appendix A pic. 3 for evidence)

Conclusion

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. For the contract, a high-level description of functionality was presented in the report's As-is overview section.

The audit report contains all found security vulnerabilities and other issues in the reviewed code.

The overall quality of the reviewed contracts is well-secured. Security engineers found three low vulnerabilities, which couldn't have any significant security impact.

Disclaimers

Disclaimer

The smart contracts given for audit had been analyzed following the best industry practices at the date of this report, concerning: 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).

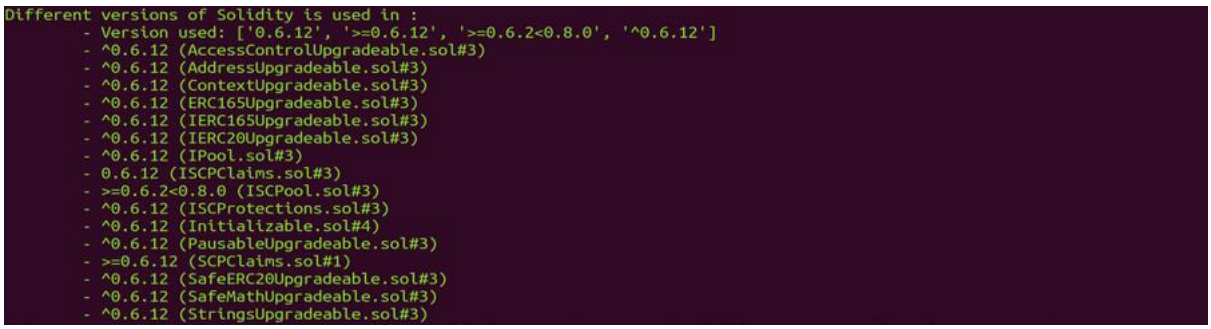
The audit makes no statements or warranties on the security of the code. It can also not be considered a sufficient assessment regarding the code's utility and safety, bug-free status, or any other contract statements. While we have done our best to conduct the analysis and produce this report, it is 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.

Technical Disclaimer

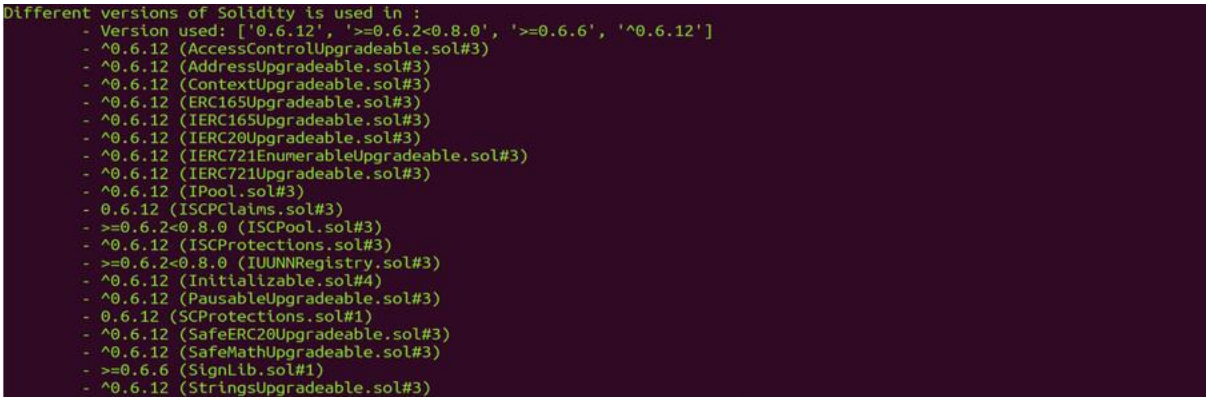
Smart contracts are deployed and executed on the blockchain platform. The platform, programming language, and other software related to the smart contract can have their vulnerabilities leading to hacks. Thus, the audit can't guarantee the explicit security of the audited smart contracts.

Appendix A. Evidences

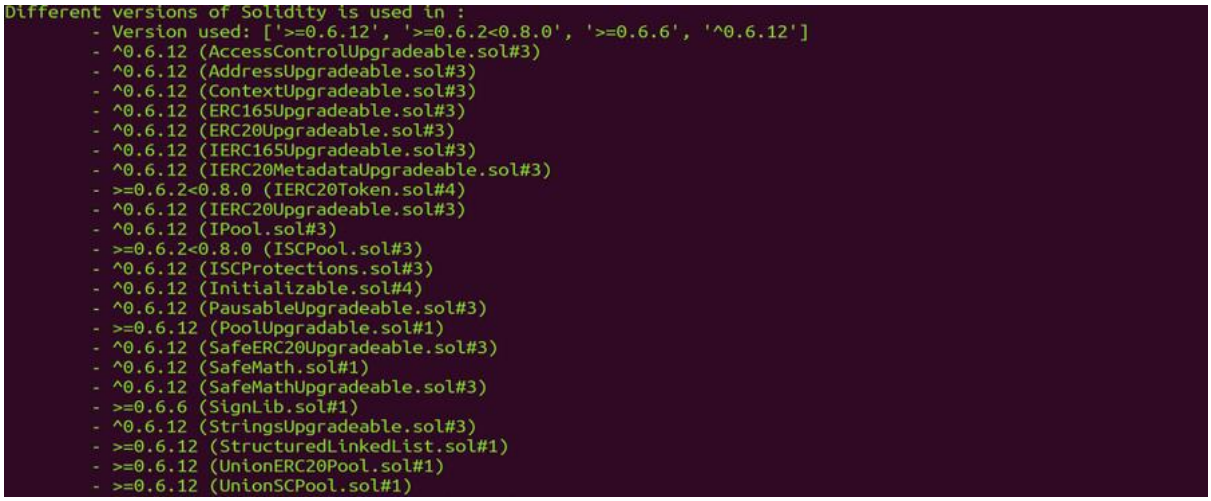
Pic 1. Used different pragma directives:



Pic 2. Used different pragma directives:



Pic 3. Used different pragma directives:



Appendix B. Automated tools reports

Pic 1. **SCPClaims** Slither automated report:

```
INFO:Detectors:
AccessControlUpgradeable.__gap (AccessControlUpgradeable.sol#252) shadows:
        - ERC165Upgradeable.__gap (ERC165Upgradeable.sol#35)
        - ContextUpgradeable.__gap (ContextUpgradeable.sol#31)
PausableUpgradeable.__gap (PausableUpgradeable.sol#96) shadows:
        - ContextUpgradeable.__gap (ContextUpgradeable.sol#31)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing
INFO:Detectors:
SCPClaims._fillClaim(ISCPool,uint64) (SCPClaims.sol#167-190) performs a multiplication on the result of a division:
        -timestamp = (_timestamp / 86400) * 86400 (SCPClaims.sol#172)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
Reentrancy in SCPClaims._fillClaim(ISCPool,uint64) (SCPClaims.sol#167-190):
        External calls:
        - require(bool,string)(scProtections.setClaimLock(address(_pool),timestamp),Can't lock the pool) (SCPClaims.sol#179)
        - IERC20Upgradeable(_pool.getBasicToken()).safeTransferFrom(msg.sender,address(this),claimFillPayment) (SCPClaims.sol#183)
        State variables written after the call(s):
        - claims[_ppID][timestamp].claimers.push(msg.sender) (SCPClaims.sol#185)
        - claims[_ppID][timestamp].claimPayments[msg.sender] = claims[_ppID][timestamp].claimPayments[msg.sender].add(claimFillPayment) (SCPClaims.sol#1
86)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
SCPClaims.setClaimingParams(uint64,uint64) (SCPClaims.sol#74-77) should emit an event for:
        - duplicationMultiplier = _duplicationMultiplier (SCPClaims.sol#75)
        - claimCreateAmount = _createPaymentAmt (SCPClaims.sol#76)
SCPClaims.setProtectionPayoutPeriod(uint64) (SCPClaims.sol#80-82) should emit an event for:
        - protectionPayoutPeriod = _protectionPayoutPeriod (SCPClaims.sol#81)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
```

```
INFO:Detectors:
Reentrancy in SCPClaims._fillClaim(ISCPool,uint64) (SCPClaims.sol#167-190):
        External calls:
        - require(bool,string)(scProtections.setClaimLock(address(_pool),timestamp),Can't lock the pool) (SCPClaims.sol#179)
        - IERC20Upgradeable(_pool.getBasicToken()).safeTransferFrom(msg.sender,address(this),claimFillPayment) (SCPClaims.sol#183)
        Event emitted after the call(s):
        - ClaimFilled(msg.sender,_ppID,timestamp) (SCPClaims.sol#188)
Reentrancy in SCPClaims.claimFeeRefund(bytes32,uint64) (SCPClaims.sol#111-118):
        External calls:
        - IERC20Upgradeable(claims[_ppID][_timestamp].pool.getBasicToken()).safeTransfer(msg.sender,claimingFeePaid) (SCPClaims.sol#115)
        Event emitted after the call(s):
        - ClaimFeeRefunded(msg.sender,_ppID,_timestamp,claimingFeePaid) (SCPClaims.sol#116)
Reentrancy in SCPClaims.setClaimRejected(bytes32,uint64) (SCPClaims.sol#141-149):
        External calls:
        - require(bool,string)(scProtections.releaseClaimLock(address(claims[_ppID][_timestamp].pool),_timestamp),Can't release pool lock) (SCPClaims.so
l#147)
        Event emitted after the call(s):
        - ClaimStatusChanged(_ppID,_timestamp,uint8(ClaimStatus.InReview),uint8(ClaimStatus.Rejected)) (SCPClaims.sol#148)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
SCPClaims.releaseLockOnExpiredClaim(bytes32,uint64) (SCPClaims.sol#151-157) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(claims[_ppID][_timestamp].lastStatusUpdateTimestamp + protectionPayoutPeriod < uint64(block.timestamp),Payout period is n
ot expired yet) (SCPClaims.sol#155)
SCPClaims._fillClaim(ISCPool,uint64) (SCPClaims.sol#167-190) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_timestamp < now,Incorrect claim timestamp) (SCPClaims.sol#171)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
AddressUpgradeable.isContract(address) (AddressUpgradeable.sol#26-35) uses assembly
        - INLINE ASM (AddressUpgradeable.sol#33)
AddressUpgradeable._verifyCallResult(bool,bytes,string) (AddressUpgradeable.sol#147-164) uses assembly
        - INLINE ASM (AddressUpgradeable.sol#156-159)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
INFO:Detectors:
Different versions of Solidity is used in :
        - Version used: ['0.6.12', '>=0.6.12', '>=0.6.2<0.8.0', '^0.6.12']
        - ^0.6.12 (AccessControlUpgradeable.sol#3)
        - ^0.6.12 (AddressUpgradeable.sol#3)
        - ^0.6.12 (ContextUpgradeable.sol#3)
        - ^0.6.12 (ERC165Upgradeable.sol#3)
        - ^0.6.12 (IERC165Upgradeable.sol#3)
        - ^0.6.12 (IERC20Upgradeable.sol#3)
        - ^0.6.12 (IPool.sol#3)
        - 0.6.12 (ISCPClaims.sol#3)
        - >=0.6.2<0.8.0 (ISCPool.sol#3)
        - ^0.6.12 (ISCProtections.sol#3)
        - ^0.6.12 (Initializable.sol#4)
        - ^0.6.12 (PausableUpgradeable.sol#3)
        - >=0.6.12 (SCPClaims.sol#1)
        - ^0.6.12 (SafeERC20Upgradeable.sol#3)
        - ^0.6.12 (SafeMathUpgradeable.sol#3)
        - ^0.6.12 (StringsUpgradeable.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Pragma version>=0.6.2<0.8.0 (ISCPool.sol#3) is too complex
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in AddressUpgradeable.sendValue(address,uint256) (AddressUpgradeable.sol#53-59):
        - (success) = recipient.call{value: amount}() (AddressUpgradeable.sol#57)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (AddressUpgradeable.sol#114-121):
        - (success,returndata) = target.call{value: value}(data) (AddressUpgradeable.sol#119)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (AddressUpgradeable.sol#139-145):
        - (success,returndata) = target.staticcall(data) (AddressUpgradeable.sol#143)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

```
INFO:Detectors:
Function AccessControlUpgradeable.__AccessControl_init() (AccessControlUpgradeable.sol#60-64) is not in mixedCase
Function AccessControlUpgradeable.__AccessControl_init_unchained() (AccessControlUpgradeable.sol#66-67) is not in mixedCase
Variable AccessControlUpgradeable.__gap (AccessControlUpgradeable.sol#252) is not in mixedCase
Function ContextUpgradeable.__Context_init() (ContextUpgradeable.sol#17-19) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (ContextUpgradeable.sol#21-22) is not in mixedCase
Variable ContextUpgradeable.__gap (ContextUpgradeable.sol#31) is not in mixedCase
Function ERC165Upgradeable.__ERC165_init() (ERC165Upgradeable.sol#23-25) is not in mixedCase
Function ERC165Upgradeable.__ERC165_init_unchained() (ERC165Upgradeable.sol#27-28) is not in mixedCase
Variable ERC165Upgradeable.__gap (ERC165Upgradeable.sol#35) is not in mixedCase
Function PausableUpgradeable.__Pausable_init() (PausableUpgradeable.sol#33-36) is not in mixedCase
Function PausableUpgradeable.__Pausable_init_unchained() (PausableUpgradeable.sol#38-40) is not in mixedCase
Variable PausableUpgradeable.__gap (PausableUpgradeable.sol#96) is not in mixedCase
Parameter SCPClaims.initialize(address,address)._admin (SCPClaims.sol#48) is not in mixedCase
Parameter SCPClaims.initialize(address,address)._scProtections (SCPClaims.sol#48) is not in mixedCase
Parameter SCPClaims.setClaimingParams(uint64,uint64)._createPaymentAmt (SCPClaims.sol#74) is not in mixedCase
Parameter SCPClaims.setClaimingParams(uint64,uint64)._duplicationMultiplier (SCPClaims.sol#74) is not in mixedCase
Parameter SCPClaims.setProtectionPayoutPeriod(uint64)._protectionPayoutPeriod (SCPClaims.sol#80) is not in mixedCase
Parameter SCPClaims.fillClaim(bytes32,uint64)._ppID (SCPClaims.sol#98) is not in mixedCase
Parameter SCPClaims.fillClaim(bytes32,uint64)._timestamp (SCPClaims.sol#98) is not in mixedCase
Parameter SCPClaims.fillClaimForPool(address,uint64)._poolAddress (SCPClaims.sol#105) is not in mixedCase
Parameter SCPClaims.fillClaimForPool(address,uint64)._timestamp (SCPClaims.sol#105) is not in mixedCase
Parameter SCPClaims.claimFeeRefund(bytes32,uint64)._ppID (SCPClaims.sol#111) is not in mixedCase
Parameter SCPClaims.claimFeeRefund(bytes32,uint64)._timestamp (SCPClaims.sol#111) is not in mixedCase
Parameter SCPClaims.setClaimInReview(bytes32,uint64)._ppID (SCPClaims.sol#120) is not in mixedCase
Parameter SCPClaims.setClaimInReview(bytes32,uint64)._timestamp (SCPClaims.sol#120) is not in mixedCase
Parameter SCPClaims.setClaimApproved(bytes32,uint64,uint8)._ppID (SCPClaims.sol#129) is not in mixedCase
Parameter SCPClaims.setClaimApproved(bytes32,uint64,uint8)._timestamp (SCPClaims.sol#129) is not in mixedCase
Parameter SCPClaims.setClaimApproved(bytes32,uint64,uint8)._payAmountPercentage (SCPClaims.sol#129) is not in mixedCase
Parameter SCPClaims.setClaimRejected(bytes32,uint64)._ppID (SCPClaims.sol#141) is not in mixedCase
Parameter SCPClaims.setClaimRejected(bytes32,uint64)._timestamp (SCPClaims.sol#141) is not in mixedCase
Parameter SCPClaims.releaseLockOnExpiredClaim(bytes32,uint64)._ppID (SCPClaims.sol#151) is not in mixedCase
Parameter SCPClaims.releaseLockOnExpiredClaim(bytes32,uint64)._timestamp (SCPClaims.sol#151) is not in mixedCase
Parameter SCPClaims.getNextClaimFillPayment(bytes32,uint64)._ppID (SCPClaims.sol#159) is not in mixedCase
Parameter SCPClaims.getNextClaimFillPayment(bytes32,uint64)._timestamp (SCPClaims.sol#159) is not in mixedCase
Parameter SCPClaims.getClaimData(bytes32,uint64)._ppID (SCPClaims.sol#211) is not in mixedCase
Parameter SCPClaims.getClaimData(bytes32,uint64)._timestamp (SCPClaims.sol#211) is not in mixedCase
Constant StringsUpgradeable.alphabet (StringsUpgradeable.sol#9) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

```
INFO:Detectors:
Redundant expression "this (ContextUpgradeable.sol#28)" inContextUpgradeable (ContextUpgradeable.sol#16-32)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
SCPClaims.version() (SCPClaims.sol#43-46) uses literals with too many digits:
        - uint32(1000001) (SCPClaims.sol#45)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
PausableUpgradeable.__gap (PausableUpgradeable.sol#96) is never used in SCPClaims (SCPClaims.sol#14-225)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
```

```
INFO:Detectors:
grantRole(bytes32,address) should be declared external:
        - AccessControlUpgradeable.grantRole(bytes32,address) (AccessControlUpgradeable.sol#172-174)
revokeRole(bytes32,address) should be declared external:
        - AccessControlUpgradeable.revokeRole(bytes32,address) (AccessControlUpgradeable.sol#185-187)
renounceRole(bytes32,address) should be declared external:
        - AccessControlUpgradeable.renounceRole(bytes32,address) (AccessControlUpgradeable.sol#203-207)
version() should be declared external:
        - SCPClaims.version() (SCPClaims.sol#43-46)
initialize(address,address) should be declared external:
        - SCPClaims.initialize(address,address) (SCPClaims.sol#48-56)
setClaimingParams(uint64,uint64) should be declared external:
        - SCPClaims.setClaimingParams(uint64,uint64) (SCPClaims.sol#74-77)
setProtectionPayoutPeriod(uint64) should be declared external:
        - SCPClaims.setProtectionPayoutPeriod(uint64) (SCPClaims.sol#80-82)
pause() should be declared external:
        - SCPClaims.pause() (SCPClaims.sol#88-90)
unpause() should be declared external:
        - SCPClaims.unpause() (SCPClaims.sol#94-96)
fillClaim(bytes32,uint64) should be declared external:
        - SCPClaims.fillClaim(bytes32,uint64) (SCPClaims.sol#98-103)
fillClaimForPool(address,uint64) should be declared external:
        - SCPClaims.fillClaimForPool(address,uint64) (SCPClaims.sol#105-109)
claimFeeRefund(bytes32,uint64) should be declared external:
        - SCPClaims.claimFeeRefund(bytes32,uint64) (SCPClaims.sol#111-118)
setClaimInReview(bytes32,uint64) should be declared external:
        - SCPClaims.setClaimInReview(bytes32,uint64) (SCPClaims.sol#120-127)
setClaimApproved(bytes32,uint64,uint8) should be declared external:
        - SCPClaims.setClaimApproved(bytes32,uint64,uint8) (SCPClaims.sol#129-139)
setClaimRejected(bytes32,uint64) should be declared external:
        - SCPClaims.setClaimRejected(bytes32,uint64) (SCPClaims.sol#141-149)
releaseLockOnExpiredClaim(bytes32,uint64) should be declared external:
        - SCPClaims.releaseLockOnExpiredClaim(bytes32,uint64) (SCPClaims.sol#151-157)
getNextClaimFillPayment(bytes32,uint64) should be declared external:
        - SCPClaims.getNextClaimFillPayment(bytes32,uint64) (SCPClaims.sol#159-165)
getClaimData(bytes32,uint64) should be declared external:
        - SCPClaims.getClaimData(bytes32,uint64) (SCPClaims.sol#211-213)
getClaimingParams() should be declared external:
        - SCPClaims.getClaimingParams() (SCPClaims.sol#220-222)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:SCPClaims.sol analyzed (18 contracts with 72 detectors), 77 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
ethsec@96a4877474d4:/src/contracts$
```

Pic 2. **SCProtections** Slither automated report:

```
INFO:Detectors:
AccessControlUpgradeable.__gap (AccessControlUpgradeable.sol#252) shadows:
        - ERC165Upgradeable.__gap (ERC165Upgradeable.sol#35)
        - ContextUpgradeable.__gap (ContextUpgradeable.sol#31)
PausableUpgradeable.__gap (PausableUpgradeable.sol#96) shadows:
        - ContextUpgradeable.__gap (ContextUpgradeable.sol#31)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing
INFO:Detectors:
SCProtections.exercise(uint256,uint64) (SCProtections.sol#181-203) uses a dangerous strict equality:
        - require(bool,string)(claimStatus == uint8(ISCPClaims.ClaimStatus.Approved),Can't exercise with unapproved claim) (SCProtections.sol#190)
SCProtections.exercise(uint256,uint64) (SCProtections.sol#181-203) uses a dangerous strict equality:
        - require(bool,string)(claimPool == address(protections[_id].pool),Claim pool reference doesn't match protection pool) (SCProtections.sol#191)
SCProtections.withdrawPremium(uint256,uint256) (SCProtections.sol#166-170) uses a dangerous strict equality:
        - require(bool,string)(msg.sender == address(protections[_id].pool) && msg.sender != address(0),Premium can be withdrawn by backed pool only) (S
CProtections.sol#167)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
SCProtections.createTo(uint256[9],bytes,address,uint256) (SCProtections.sol#129-164) ignores return value by IERC20Upgradeable(ISCPool(address(data[4]))
.getBasicToken()).approve(address(data[4]),data[1]) (SCProtections.sol#140)
SCProtections.exercise(uint256,uint64) (SCProtections.sol#181-203) ignores return value by protections[_id].pool.onPayoutCoverage(_id,protections[_id].p
remium,amountToPay,msg.sender) (SCProtections.sol#200)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
IUUNNRegistry.mint(uint256,address,address).protectionContract (IUUNNRegistry.sol#9) shadows:
        - IUUNNRegistry.protectionContract(uint256) (IUUNNRegistry.sol#11) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
SCProtections.isClaimLocked(address) (SCProtections.sol#234-251) has external calls inside a loop: (claimStatus,payoutPercent,claimPool,lastStatusUpdate
Timestamp) = claimStorage.getClaimData(_ppID,activeClaimsForPool[_pool][i]) (SCProtections.sol#241)
SCProtections.isClaimLocked(address) (SCProtections.sol#234-251) has external calls inside a loop: lastStatusUpdateTimestamp + claimStorage.protectionPa
youtPeriod() >= uint64(block.timestamp) (SCProtections.sol#243)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
```

```
INFO:Detectors:
SCProtections.isClaimLocked(address) (SCProtections.sol#234-251) has external calls inside a loop: (claimStatus,payoutPercent,claimPool,lastStatusUpdate
Timestamp) = claimStorage.getClaimData(_ppID,activeClaimsForPool[_pool][i]) (SCProtections.sol#241)
SCProtections.isClaimLocked(address) (SCProtections.sol#234-251) has external calls inside a loop: lastStatusUpdateTimestamp + claimStorage.protectionPa
youtPeriod() >= uint64(block.timestamp) (SCProtections.sol#243)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
Reentrancy in SCProtections.createTo(uint256[9],bytes,address,uint256) (SCProtections.sol#129-164):
        External calls:
        - IERC20Upgradeable(ISCPool(address(data[4])).getBasicToken()).safeTransferFrom(msg.sender,address(this),data[1]) (SCProtections.sol#139)
        - IERC20Upgradeable(ISCPool(address(data[4])).getBasicToken()).approve(address(data[4]),data[1]) (SCProtections.sol#140)
        - ISCPool(address(data[4])).onProtectionPremium(address(this),(data[0],data[1],data[3],data[2],data[5],data[6],data[7])) (SCProtections.sol#143)
        State variables written after the call(s):
        - protections[data[0]] = SCProtectionData(ISCPool(address(data[4])),timelimits,data[3],data[1],0) (SCProtections.sol#156)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in SCProtections.createTo(uint256[9],bytes,address,uint256) (SCProtections.sol#129-164):
        External calls:
        - IERC20Upgradeable(ISCPool(address(data[4])).getBasicToken()).safeTransferFrom(msg.sender,address(this),data[1]) (SCProtections.sol#139)
        - IERC20Upgradeable(ISCPool(address(data[4])).getBasicToken()).approve(address(data[4]),data[1]) (SCProtections.sol#140)
        - ISCPool(address(data[4])).onProtectionPremium(address(this),(data[0],data[1],data[3],data[2],data[5],data[6],data[7])) (SCProtections.sol#143)
        - uunn.mint(data[0],address(this),erc721Receiver) (SCProtections.sol#157)
        Event emitted after the call(s):
        - SCProtectionCreated(erc721Receiver,data[0],address(data[4]),data[3],now,data[2],data[1]) (SCProtections.sol#160)
Reentrancy in SCProtections.exercise(uint256,uint64) (SCProtections.sol#181-203):
        External calls:
        - protections[_id].pool.onPayoutCoverage(_id,protections[_id].premium,amountToPay,msg.sender) (SCProtections.sol#200)
        Event emitted after the call(s):
        - Exercised(_id,_amount,amountToPay) (SCProtections.sol#202)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

```
INFO:Detectors:
SCProtections.createTo(uint256[9],bytes,address,uint256) (SCProtections.sol#129-164) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp <= deadline,Transaction expired) (SCProtections.sol#130)
        - require(bool,string)(block.timestamp <= data[8],Quotation expired) (SCProtections.sol#136)
SCProtections.withdrawPremium(uint256,uint256) (SCProtections.sol#166-170) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(msg.sender == address(protections[_id].pool) && msg.sender != address(0),Premium can be withdrawn by backed pool only) (S
CProtections.sol#167)
        - require(bool,string)(protections[_id].premium >= _premium,Not enough premium left) (SCProtections.sol#168)
SCProtections.exercise(uint256,uint64) (SCProtections.sol#181-203) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(address(protections[_id].pool) != address(0),Pool not registered) (SCProtections.sol#183)
        - require(bool,string)(now <= validTo,Protection expired) (SCProtections.sol#185)
        - require(bool,string)(claimStatus > 0,Claim not found) (SCProtections.sol#189)
        - require(bool,string)(claimStatus == uint8(ISCPClaims.ClaimStatus.Approved),Can't exercise with unapproved claim) (SCProtections.sol#190)
        - require(bool,string)(claimPool == address(protections[_id].pool),Claim pool reference doesn't match protection pool) (SCProtections.sol#191)
        - require(bool,string)(lastStatusUpdateTimestamp + claimStorage.protectionPayoutPeriod() >= uint64(block.timestamp),Protection payout period exp
ired) (SCProtections.sol#192)
SCProtections.isClaimLocked(address) (SCProtections.sol#234-251) uses timestamp for comparisons
        Dangerous comparisons:
        - lastStatusUpdateTimestamp + claimStorage.protectionPayoutPeriod() >= uint64(block.timestamp) (SCProtections.sol#243)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
AddressUpgradeable.isContract(address) (AddressUpgradeable.sol#26-35) uses assembly
        - INLINE ASM (AddressUpgradeable.sol#33)
AddressUpgradeable._verifyCallResult(bool,bytes,string) (AddressUpgradeable.sol#147-164) uses assembly
        - INLINE ASM (AddressUpgradeable.sol#156-159)
SignLib.splitSignature(bytes) (SignLib.sol#19-40) uses assembly
        - INLINE ASM (SignLib.sol#30-37)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
INFO:Detectors:
Different versions of Solidity is used in :
        - Version used: ['0.6.12', '>=0.6.2<0.8.0', '>=0.6.6', '^0.6.12']
        - ^0.6.12 (AccessControlUpgradeable.sol#3)
        - ^0.6.12 (AddressUpgradeable.sol#3)
        - ^0.6.12 (ContextUpgradeable.sol#3)
        - ^0.6.12 (ERC165Upgradeable.sol#3)
        - ^0.6.12 (IERC165Upgradeable.sol#3)
        - ^0.6.12 (IERC20Upgradeable.sol#3)
        - ^0.6.12 (IERC721EnumerableUpgradeable.sol#3)
        - ^0.6.12 (IERC721Upgradeable.sol#3)
        - ^0.6.12 (IPool.sol#3)
        - 0.6.12 (ISCPClaims.sol#3)
        - >=0.6.2<0.8.0 (ISCPool.sol#3)
        - ^0.6.12 (ISCProtections.sol#3)
        - >=0.6.2<0.8.0 (IUUNNRegistry.sol#3)
        - ^0.6.12 (Initializable.sol#4)
        - ^0.6.12 (PausableUpgradeable.sol#3)
        - 0.6.12 (SCProtections.sol#1)
        - ^0.6.12 (SafeERC20Upgradeable.sol#3)
        - ^0.6.12 (SafeMathUpgradeable.sol#3)
        - >=0.6.6 (SignLib.sol#1)
        - ^0.6.12 (StringsUpgradeable.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Pragma version>=0.6.2<0.8.0 (ISCPool.sol#3) is too complex
Pragma version>=0.6.2<0.8.0 (IUUNNRegistry.sol#3) is too complex
Pragma version>=0.6.6 (SignLib.sol#1) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in AddressUpgradeable.sendValue(address,uint256) (AddressUpgradeable.sol#53-59):
        - (success) = recipient.call{value: amount}() (AddressUpgradeable.sol#57)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (AddressUpgradeable.sol#114-121):
        - (success,returndata) = target.call{value: value}(data) (AddressUpgradeable.sol#119)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (AddressUpgradeable.sol#139-145):
        - (success,returndata) = target.staticcall(data) (AddressUpgradeable.sol#143)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

```
INFO:Detectors:
Function AccessControlUpgradeable.__AccessControl_init() (AccessControlUpgradeable.sol#60-64) is not in mixedCase
Function AccessControlUpgradeable.__AccessControl_init_unchained() (AccessControlUpgradeable.sol#66-67) is not in mixedCase
Variable AccessControlUpgradeable.__gap (AccessControlUpgradeable.sol#252) is not in mixedCase
Function ContextUpgradeable.__Context_init() (ContextUpgradeable.sol#17-19) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (ContextUpgradeable.sol#21-22) is not in mixedCase
Variable ContextUpgradeable.__gap (ContextUpgradeable.sol#31) is not in mixedCase
Function ERC165Upgradeable.__ERC165_init() (ERC165Upgradeable.sol#23-25) is not in mixedCase
Function ERC165Upgradeable.__ERC165_init_unchained() (ERC165Upgradeable.sol#27-28) is not in mixedCase
Variable ERC165Upgradeable.__gap (ERC165Upgradeable.sol#35) is not in mixedCase
Function PausableUpgradeable.__Pausable_init() (PausableUpgradeable.sol#33-36) is not in mixedCase
Function PausableUpgradeable.__Pausable_init_unchained() (PausableUpgradeable.sol#38-40) is not in mixedCase
Variable PausableUpgradeable.__gap (PausableUpgradeable.sol#96) is not in mixedCase
Parameter SCProtections.initialize(address,address,address)._admin (SCProtections.sol#65) is not in mixedCase
Parameter SCProtections.initialize(address,address,address)._uunn (SCProtections.sol#65) is not in mixedCase
Parameter SCProtections.initialize(address,address,address)._claimStorage (SCProtections.sol#65) is not in mixedCase
Parameter SCProtections.withdrawPremium(uint256,uint256)._id (SCProtections.sol#166) is not in mixedCase
Parameter SCProtections.withdrawPremium(uint256,uint256)._premium (SCProtections.sol#166) is not in mixedCase
Parameter SCProtections.setActiveSCProtectionPoolAddress(bytes32,address)._ppID (SCProtections.sol#172) is not in mixedCase
Parameter SCProtections.setActiveSCProtectionPoolAddress(bytes32,address)._value (SCProtections.sol#172) is not in mixedCase
Parameter SCProtections.getActiveSCProtectionPool(bytes32)._ppID (SCProtections.sol#177) is not in mixedCase
Parameter SCProtections.exercise(uint256,uint64)._id (SCProtections.sol#181) is not in mixedCase
Parameter SCProtections.exercise(uint256,uint64)._timestamp (SCProtections.sol#181) is not in mixedCase
Parameter SCProtections.setClaimLock(address,uint64)._pool (SCProtections.sol#205) is not in mixedCase
Parameter SCProtections.setClaimLock(address,uint64)._timestamp (SCProtections.sol#205) is not in mixedCase
Parameter SCProtections.releaseClaimLock(address,uint64)._pool (SCProtections.sol#212) is not in mixedCase
Parameter SCProtections.releaseClaimLock(address,uint64)._timestamp (SCProtections.sol#212) is not in mixedCase
Parameter SCProtections.isClaimLocked(address)._pool (SCProtections.sol#234) is not in mixedCase
Parameter SCProtections.getDocument(uint256,bytes32)._name (SCProtections.sol#326) is not in mixedCase
Constant StringsUpgradeable.alphabet (StringsUpgradeable.sol#9) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (ContextUpgradeable.sol#28)" inContextUpgradeable (ContextUpgradeable.sol#16-32)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
```

```
INFO:Detectors:
SCProtections.version() (SCProtections.sol#60-63) uses literals with too many digits:
        - uint32(1000001) (SCProtections.sol#62)
SCProtections.exercise(uint256,uint64) (SCProtections.sol#181-203) uses literals with too many digits:
        - validTo = protections[_id].timelimits & 0x0000000000000000000000000000000000000000000000000000FFFFFFFFFFFF (SCProtections.sol#184)
SCProtections.getProtectionData(uint256) (SCProtections.sol#265-274) uses literals with too many digits:
        - (address(protections[id].pool),protections[id].pool.ppID(),protections[id].amount,protections[id].premium,(protections[id].timelimits >> 64) &
 0x0000000000000000000000000000000000000000000000000000FFFFFFFFFFFF,protections[id].timelimits & 0x0000000000000000000000000000000000000000000000000000FFFF
FFFFFFFFFF) (SCProtections.sol#266-273)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
PausableUpgradeable.__gap (PausableUpgradeable.sol#96) is never used in SCProtections (SCProtections.sol#17-343)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
grantRole(bytes32,address) should be declared external:
        - AccessControlUpgradeable.grantRole(bytes32,address) (AccessControlUpgradeable.sol#172-174)
revokeRole(bytes32,address) should be declared external:
        - AccessControlUpgradeable.revokeRole(bytes32,address) (AccessControlUpgradeable.sol#185-187)
renounceRole(bytes32,address) should be declared external:
        - AccessControlUpgradeable.renounceRole(bytes32,address) (AccessControlUpgradeable.sol#203-207)
initialize(address,address,address) should be declared external:
        - SCProtections.initialize(address,address,address) (SCProtections.sol#65-72)
pause() should be declared external:
        - SCProtections.pause() (SCProtections.sol#85-87)
unpause() should be declared external:
        - SCProtections.unpause() (SCProtections.sol#91-93)
create(uint256[9],bytes,uint256) should be declared external:
        - SCProtections.create(uint256[9],bytes,uint256) (SCProtections.sol#109-111)
setActiveSCProtectionPoolAddress(bytes32,address) should be declared external:
        - SCProtections.setActiveSCProtectionPoolAddress(bytes32,address) (SCProtections.sol#172-175)
exercise(uint256,uint64) should be declared external:
        - SCProtections.exercise(uint256,uint64) (SCProtections.sol#181-203)
setClaimLock(address,uint64) should be declared external:
        - SCProtections.setClaimLock(address,uint64) (SCProtections.sol#205-210)
releaseClaimLock(address,uint64) should be declared external:
        - SCProtections.releaseClaimLock(address,uint64) (SCProtections.sol#212-232)
isClaimLocked(address) should be declared external:
        - SCProtections.isClaimLocked(address) (SCProtections.sol#234-251)
getProtectionData(uint256) should be declared external:
        - SCProtections.getProtectionData(uint256) (SCProtections.sol#265-274)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:SCProtections.sol analyzed (22 contracts with 72 detectors), 74 result(s) found
```

Pic 3. **UnionSCPool** Slither automated report:

```
INFO:Detectors:
AccessControlUpgradeable.__gap (AccessControlUpgradeable.sol#252) shadows:
        - ERC165Upgradeable.__gap (ERC165Upgradeable.sol#35)
        - ContextUpgradeable.__gap (ContextUpgradeable.sol#31)
ERC20Upgradeable.__gap (ERC20Upgradeable.sol#310) shadows:
        - ContextUpgradeable.__gap (ContextUpgradeable.sol#31)
PausableUpgradeable.__gap (PausableUpgradeable.sol#96) shadows:
        - ContextUpgradeable.__gap (ContextUpgradeable.sol#31)
PoolUpgradable.__gap (PoolUpgradable.sol#109) shadows:
        - ERC20Upgradeable.__gap (ERC20Upgradeable.sol#310)
        - ContextUpgradeable.__gap (ContextUpgradeable.sol#31)
UnionERC20Pool.__gap (UnionERC20Pool.sol#300) shadows:
        - PoolUpgradable.__gap (PoolUpgradable.sol#109)
        - ERC20Upgradeable.__gap (ERC20Upgradeable.sol#310)
        - PausableUpgradeable.__gap (PausableUpgradeable.sol#96)
        - AccessControlUpgradeable.__gap (AccessControlUpgradeable.sol#252)
        - ERC165Upgradeable.__gap (ERC165Upgradeable.sol#35)
        - ContextUpgradeable.__gap (ContextUpgradeable.sol#31)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing
INFO:Detectors:
UnionERC20Pool._updateMCR(uint256,uint256,uint256) (UnionERC20Pool.sol#248-268) ignores return value by mcrPendingsList.pushBack(item) (UnionERC20Pool.s
ol#263)
UnionERC20Pool._unloadMCRPendingQueue(uint256,uint256) (UnionERC20Pool.sol#270-283) ignores return value by mcrPendingsList.remove(item) (UnionERC20Pool
.sol#278)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
UnionSCPool.setLockupPeriod(uint256) (UnionSCPool.sol#48-50) should emit an event for:
        - lockupPeriod = value (UnionSCPool.sol#49)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
StructuredLinkedList.getSortedSpot(StructuredLinkedList.List,address,uint256) (StructuredLinkedList.sol#124-135) has external calls inside a loop: (next
 != 0) && ((_value < IStructureInterface(_structure).getValue(next)) != _NEXT) (StructuredLinkedList.sol#131)
UnionSCPool.unlockPremium(uint256[]) (UnionSCPool.sol#72-97) has external calls inside a loop: (premium,validTo) = scProtectionStorage.getProtectionData
(_ids[i]) (UnionSCPool.sol#84)
UnionSCPool.unlockPremium(uint256[]) (UnionSCPool.sol#72-97) has external calls inside a loop: scProtectionStorage.withdrawPremium(_ids[i],premium) (Union
SCPool.sol#86)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
```

```
INFO:Detectors:
Reentrancy in PoolUpgradable._withdraw(uint256,address) (PoolUpgradable.sol#92-102):
        External calls:
        - basicToken.safeTransfer(to,revenue) (PoolUpgradable.sol#98)
        State variables written after the call(s):
        - withdrawals[msg.sender] = withdrawals[msg.sender].add(revenue) (PoolUpgradable.sol#99)
Reentrancy in PoolUpgradable.deposit(uint256) (PoolUpgradable.sol#52-57):
        External calls:
        - basicToken.safeTransferFrom(msg.sender,address(this),_amount) (PoolUpgradable.sol#55)
        State variables written after the call(s):
        - _deposit(_amount,msg.sender) (PoolUpgradable.sol#56)
                - _balances[account] += amount (ERC20Upgradeable.sol#246)
        - _deposit(_amount,msg.sender) (PoolUpgradable.sol#56)
                - _totalSupply += amount (ERC20Upgradeable.sol#245)
        - _deposit(_amount,msg.sender) (PoolUpgradable.sol#56)
                - deposits[to] = deposits[to].add(amount) (PoolUpgradable.sol#87)
        - _deposit(_amount,msg.sender) (PoolUpgradable.sol#56)
                - totalCap = totalCap.add(amount) (PoolUpgradable.sol#86)
Reentrancy in PoolUpgradable.depositTo(uint256,address) (PoolUpgradable.sol#40-46):
        External calls:
        - basicToken.safeTransferFrom(msg.sender,address(this),_amount) (PoolUpgradable.sol#44)
        State variables written after the call(s):
        - _deposit(_amount,_to) (PoolUpgradable.sol#45)
                - _balances[account] += amount (ERC20Upgradeable.sol#246)
        - _deposit(_amount,_to) (PoolUpgradable.sol#45)
                - _totalSupply += amount (ERC20Upgradeable.sol#245)
        - _deposit(_amount,_to) (PoolUpgradable.sol#45)
                - deposits[to] = deposits[to].add(amount) (PoolUpgradable.sol#87)
        - _deposit(_amount,_to) (PoolUpgradable.sol#45)
                - totalCap = totalCap.add(amount) (PoolUpgradable.sol#86)
Reentrancy in UnionSCPool.onProtectionPremium(address,uint256[7]) (UnionSCPool.sol#58-70):
        External calls:
        - basicToken.safeTransferFrom(buyer,address(this),data[1]) (UnionSCPool.sol#67)
        State variables written after the call(s):
        - lockedPremium = lockedPremium.add(data[1]) (UnionSCPool.sol#68)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

```
INFO:Detectors:
Reentrancy in PoolUpgradable._withdraw(uint256,address) (PoolUpgradable.sol#92-102):
        External calls:
        - basicToken.safeTransfer(to,revenue) (PoolUpgradable.sol#98)
        Event emitted after the call(s):
        - Withdraw(msg.sender,revenue) (PoolUpgradable.sol#100)
Reentrancy in PoolUpgradable.deposit(uint256) (PoolUpgradable.sol#52-57):
        External calls:
        - basicToken.safeTransferFrom(msg.sender,address(this),_amount) (PoolUpgradable.sol#55)
        Event emitted after the call(s):
        - Deposit(to,amount) (PoolUpgradable.sol#88)
                - _deposit(_amount,msg.sender) (PoolUpgradable.sol#56)
        - Transfer(address(0),account,amount) (ERC20Upgradeable.sol#247)
                - _deposit(_amount,msg.sender) (PoolUpgradable.sol#56)
Reentrancy in PoolUpgradable.depositTo(uint256,address) (PoolUpgradable.sol#40-46):
        External calls:
        - basicToken.safeTransferFrom(msg.sender,address(this),_amount) (PoolUpgradable.sol#44)
        Event emitted after the call(s):
        - Deposit(to,amount) (PoolUpgradable.sol#88)
                - _deposit(_amount,_to) (PoolUpgradable.sol#45)
        - Transfer(address(0),account,amount) (ERC20Upgradeable.sol#247)
                - _deposit(_amount,_to) (PoolUpgradable.sol#45)
Reentrancy in UnionSCPool.onPayoutCoverage(uint256,uint256,uint256,address) (UnionSCPool.sol#99-109):
        External calls:
        - basicToken.safeTransfer(_beneficiary,_coverageToPay) (UnionSCPool.sol#105)
        Event emitted after the call(s):
        - PoolProtectionCoveragePaid(_id,_coverageToPay,_beneficiary) (UnionSCPool.sol#106)
Reentrancy in UnionSCPool.onProtectionPremium(address,uint256[7]) (UnionSCPool.sol#58-70):
        External calls:
        - basicToken.safeTransferFrom(buyer,address(this),data[1]) (UnionSCPool.sol#67)
        Event emitted after the call(s):
        - PoolProtectionIssued(_id,data[1],data[2],uint64(data[3])) (UnionSCPool.sol#69)
Reentrancy in UnionSCPool.unlockPremium(uint256[]) (UnionSCPool.sol#72-97):
        External calls:
        - scProtectionStorage.withdrawPremium(_ids[i],premium) (UnionSCPool.sol#86)
        Event emitted after the call(s):
        - PoolProtectionPremiumUnlocked(_ids[i],premium) (UnionSCPool.sol#88)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

```
INFO:Detectors:
UnionSCPool.onProtectionPremium(address,uint256[7]) (UnionSCPool.sol#58-70) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(data[3] > block.timestamp,Invalid _validto parameter) (UnionSCPool.sol#63)
UnionSCPool.unlockPremium(uint256[]) (UnionSCPool.sol#72-97) uses timestamp for comparisons
        Dangerous comparisons:
        - validTo < block.timestamp && premium > 0 (UnionSCPool.sol#85)
UnionSCPool.withdrawWithData(uint256,uint256,uint256[5],bytes) (UnionSCPool.sol#128-155) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(lastProvideTimestamp[msg.sender].add(lockupPeriod) <= now,Withdrawal is locked up) (UnionSCPool.sol#129)
        - require(bool,string)(block.timestamp <= _data[4],quotation expired) (UnionSCPool.sol#145)
UnionSCPool._beforeTokenTransfer(address,address,uint256) (UnionSCPool.sol#157-160) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(lastProvideTimestamp[from].add(lockupPeriod) <= now,Withdrawal is locked up) (UnionSCPool.sol#158)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
AddressUpgradeable.isContract(address) (AddressUpgradeable.sol#26-35) uses assembly
        - INLINE ASM (AddressUpgradeable.sol#33)
AddressUpgradeable._verifyCallResult(bool,bytes,string) (AddressUpgradeable.sol#147-164) uses assembly
        - INLINE ASM (AddressUpgradeable.sol#156-159)
SignLib.splitSignature(bytes) (SignLib.sol#19-40) uses assembly
        - INLINE ASM (SignLib.sol#30-37)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
INFO:Detectors:
Different versions of Solidity is used in :
        - Version used: ['>=0.6.12', '>=0.6.2<0.8.0', '>=0.6.6', '^0.6.12']
        - ^0.6.12 (AccessControlUpgradeable.sol#3)
        - ^0.6.12 (AddressUpgradeable.sol#3)
        - ^0.6.12 (ContextUpgradeable.sol#3)
        - ^0.6.12 (ERC165Upgradeable.sol#3)
        - ^0.6.12 (ERC20Upgradeable.sol#3)
        - ^0.6.12 (IERC165Upgradeable.sol#3)
        - ^0.6.12 (IERC20MetadataUpgradeable.sol#3)
        - >=0.6.2<0.8.0 (IERC20Token.sol#4)
        - ^0.6.12 (IERC20Upgradeable.sol#3)
        - ^0.6.12 (IPool.sol#3)
        - >=0.6.2<0.8.0 (ISCPool.sol#3)
        - ^0.6.12 (ISCProtections.sol#3)
        - ^0.6.12 (Initializable.sol#4)
        - ^0.6.12 (PausableUpgradeable.sol#3)
        - >=0.6.12 (PoolUpgradable.sol#1)
        - ^0.6.12 (SafeERC20Upgradeable.sol#3)
        - ^0.6.12 (SafeMath.sol#1)
        - ^0.6.12 (SafeMathUpgradeable.sol#3)
        - >=0.6.6 (SignLib.sol#1)
        - ^0.6.12 (StringsUpgradeable.sol#3)
        - >=0.6.12 (StructuredLinkedList.sol#1)
        - >=0.6.12 (UnionERC20Pool.sol#1)
        - >=0.6.12 (UnionSCPool.sol#1)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Pragma version>=0.6.2<0.8.0 (IERC20Token.sol#4) is too complex
Pragma version>=0.6.2<0.8.0 (ISCPool.sol#3) is too complex
Pragma version>=0.6.6 (SignLib.sol#1) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

```
INFO:Detectors:
Low level call in AddressUpgradeable.sendValue(address,uint256) (AddressUpgradeable.sol#53-59):
        - (success) = recipient.call{value: amount}() (AddressUpgradeable.sol#57)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (AddressUpgradeable.sol#114-121):
        - (success,returndata) = target.call{value: value}(data) (AddressUpgradeable.sol#119)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (AddressUpgradeable.sol#139-145):
        - (success,returndata) = target.staticcall(data) (AddressUpgradeable.sol#143)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

```
INFO:Detectors:
Function AccessControlUpgradeable.__AccessControl_init() (AccessControlUpgradeable.sol#60-64) is not in mixedCase
Function AccessControlUpgradeable.__AccessControl_init_unchained() (AccessControlUpgradeable.sol#66-67) is not in mixedCase
Variable AccessControlUpgradeable.__gap (AccessControlUpgradeable.sol#252) is not in mixedCase
Function ContextUpgradeable.__Context_init() (ContextUpgradeable.sol#17-19) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (ContextUpgradeable.sol#21-22) is not in mixedCase
Variable ContextUpgradeable.__gap (ContextUpgradeable.sol#31) is not in mixedCase
Function ERC165Upgradeable.__ERC165_init() (ERC165Upgradeable.sol#23-25) is not in mixedCase
Function ERC165Upgradeable.__ERC165_init_unchained() (ERC165Upgradeable.sol#27-28) is not in mixedCase
Variable ERC165Upgradeable.__gap (ERC165Upgradeable.sol#35) is not in mixedCase
Function ERC20Upgradeable.__ERC20_init(string,string) (ERC20Upgradeable.sol#53-56) is not in mixedCase
Function ERC20Upgradeable.__ERC20_init_unchained(string,string) (ERC20Upgradeable.sol#58-61) is not in mixedCase
Variable ERC20Upgradeable.__gap (ERC20Upgradeable.sol#310) is not in mixedCase
Function PausableUpgradeable.__Pausable_init() (PausableUpgradeable.sol#33-36) is not in mixedCase
Function PausableUpgradeable.__Pausable_init_unchained() (PausableUpgradeable.sol#38-40) is not in mixedCase
Variable PausableUpgradeable.__gap (PausableUpgradeable.sol#96) is not in mixedCase
Function PoolUpgradable.__Pool_init(address,string) (PoolUpgradable.sol#25-28) is not in mixedCase
Parameter PoolUpgradable.__Pool_init(address,string)._basicToken (PoolUpgradable.sol#25) is not in mixedCase
Parameter PoolUpgradable.__Pool_init(address,string)._description (PoolUpgradable.sol#25) is not in mixedCase
Function PoolUpgradable.__Pool_init_unchained(address) (PoolUpgradable.sol#30-32) is not in mixedCase
Parameter PoolUpgradable.__Pool_init_unchained(address)._basicToken (PoolUpgradable.sol#30) is not in mixedCase
Parameter PoolUpgradable.depositTo(uint256,address)._amount (PoolUpgradable.sol#41) is not in mixedCase
Parameter PoolUpgradable.depositTo(uint256,address)._to (PoolUpgradable.sol#42) is not in mixedCase
Parameter PoolUpgradable.deposit(uint256)._amount (PoolUpgradable.sol#53) is not in mixedCase
Parameter PoolUpgradable.withdraw(uint256)._amount (PoolUpgradable.sol#65) is not in mixedCase
Parameter PoolUpgradable.withdrawTo(uint256,address)._amount (PoolUpgradable.sol#76) is not in mixedCase
Parameter PoolUpgradable.wlthdrawTo(uint256,address)._to (PoolUpgradable.sol#77) is not in mixedCase
Variable PoolUpgradable.__gap (PoolUpgradable.sol#109) is not in mixedCase
Constant StringsUpgradeable.alphabet (StringsUpgradeable.sol#9) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter StructuredLinkedList.nodeExists(StructuredLinkedList.List,uint256)._node (StructuredLinkedList.sol#45) is not in mixedCase
Parameter StructuredLinkedList.getNode(StructuredLinkedList.List,uint256)._node (StructuredLinkedList.sol#72) is not in mixedCase
Parameter StructuredLinkedList.getAdjacent(StructuredLinkedList.List,uint256,bool)._node (StructuredLinkedList.sol#87) is not in mixedCase
Parameter StructuredLinkedList.getAdjacent(StructuredLinkedList.List,uint256,bool)._direction (StructuredLinkedList.sol#87) is not in mixedCase
Parameter StructuredLinkedList.getNextNode(StructuredLinkedList.List,uint256)._node (StructuredLinkedList.sol#101) is not in mixedCase
Parameter StructuredLinkedList.getPreviousNode(StructuredLinkedList.List,uint256)._node (StructuredLinkedList.sol#111) is not in mixedCase
Parameter StructuredLinkedList.getSortedSpot(StructuredLinkedList.List,address,uint256)._structure (StructuredLinkedList.sol#124) is not in mixedCase
Parameter StructuredLinkedList.getSortedSpot(StructuredLinkedList.List,address,uint256)._value (StructuredLinkedList.sol#124) is not in mixedCase
Parameter StructuredLinkedList.insertAfter(StructuredLinkedList.List,uint256,uint256)._node (StructuredLinkedList.sol#150) is not in mixedCase
Parameter StructuredLinkedList.insertAfter(StructuredLinkedList.List,uint256,uint256)._new (StructuredLinkedList.sol#150) is not in mixedCase
Parameter StructuredLinkedList.insertBefore(StructuredLinkedList.List,uint256,uint256)._node (StructuredLinkedList.sol#161) is not in mixedCase
Parameter StructuredLinkedList.insertBefore(StructuredLinkedList.List,uint256,uint256)._new (StructuredLinkedList.sol#161) is not in mixedCase
Parameter StructuredLinkedList.remove(StructuredLinkedList.List,uint256)._node (StructuredLinkedList.sol#171) is not in mixedCase
Parameter StructuredLinkedList.pushFront(StructuredLinkedList.List,uint256)._node (StructuredLinkedList.sol#190) is not in mixedCase
Parameter StructuredLinkedList.pushBack(StructuredLinkedList.List,uint256)._node (StructuredLinkedList.sol#200) is not in mixedCase
Function UnionERC20Pool.__UnionERC20Pool_init(address,address,string) (UnionERC20Pool.sol#60-65) is not in mixedCase
Parameter UnionERC20Pool.__UnionERC20Pool_init(address,address,string)._basicToken (UnionERC20Pool.sol#60) is not in mixedCase
Parameter UnionERC20Pool.__UnionERC20Pool_init(address,address,string)._descRiption (UnionERC20Pool.sol#60) is not in mixedCase
Function UnionERC20Pool.__UnionERC20Pool_init_unchained(address) (UnionERC20Pool.sol#67-91) is not in mixedCase
Parameter UnionERC20Pool.setPoolReserveAddress(address)._poolReserveAddress (UnionERC20Pool.sol#110) is not in mixedCase
Parameter UnionERC20Pool.setFoundationReserveAddress(address)._foundationReserveAddress (UnionERC20Pool.sol#115) is not in mixedCase
Parameter UnionERC20Pool.setPoolReservePremiumCommission(uint8,uint8)._nom (UnionERC20Pool.sol#120) is not in mixedCase
Parameter UnionERC20Pool.setPoolReservePremiumCommission(uint8,uint8)._denom (UnionERC20Pool.sol#120) is not in mixedCase
Parameter UnionERC20Pool.setFoundationReservePremiumCommission(uint8,uint8)._nom (UnionERC20Pool.sol#126) is not in mixedCase
Parameter UnionERC20Pool.setFoundationReservePremiumCommission(uint8,uint8)._denom (UnionERC20Pool.sol#126) is not in mixedCase
Parameter UnionERC20Pool.setPoolReserveExcessLiquidityCommission(uint8,uint8)._nom (UnionERC20Pool.sol#132) is not in mixedCase
Parameter UnionERC20Pool.setPoolReserveExcessLiquidityCommission(uint8,uint8)._denom (UnionERC20Pool.sol#132) is not in mixedCase
Parameter UnionERC20Pool.setFoundationReserveExcessLiquidityCommission(uint8,uint8)._nom (UnionERC20Pool.sol#138) is not in mixedCase
Parameter UnionERC20Pool.setFoundationReserveExcessLiquidityCommission(uint8,uint8)._denom (UnionERC20Pool.sol#138) is not in mixedCase
Parameter UnionERC20Pool.getWriterData(address)._writer (UnionERC20Pool.sol#224) is not in mixedCase
Variable UnionERC20Pool.__gap (UnionERC20Pool.sol#300) is not in mixedCase
Variable UnionERC20Pool.OPERATOR_ROLE (UnionERC20Pool.sol#26) is not in mixedCase
Variable UnionERC20Pool.MCR_PROVIDER (UnionERC20Pool.sol#27) is not in mixedCase
```

```
Parameter UnionSCPool.getWriterDataExtended(address)._writer (UnionSCPool.sol#36) is not in mixedCase
Parameter UnionSCPool.initialize(address,address,bytes32,address,string)._basicToken (UnionSCPool.sol#40) is not in mixedCase
Parameter UnionSCPool.initialize(address,address,bytes32,address,string)._ppID (UnionSCPool.sol#40) is not in mixedCase
Parameter UnionSCPool.initialize(address,address,bytes32,address,string)._scProtectionStorage (UnionSCPool.sol#40) is not in mixedCase
Parameter UnionSCPool.initialize(address,address,bytes32,address,string)._description (UnionSCPool.sol#40) is not in mixedCase
Parameter UnionSCPool.unlockPremium(uint256[])._lds (UnionSCPool.sol#72) is not in mixedCase
Parameter UnionSCPool.onPayoutCoverage(uint256,uint256,uint256,address)._id (UnionSCPool.sol#99) is not in mixedCase
Parameter UnionSCPool.onPayoutCoverage(uint256,uint256,uint256,address)._premiumToUnlock (UnionSCPool.sol#99) is not in mixedCase
Parameter UnionSCPool.onPayoutCoverage(uint256,uint256,uint256,address)._coverageToPay (UnionSCPool.sol#99) is not in mixedCase
Parameter UnionSCPool.onPayoutCoverage(uint256,uint256,uint256,address)._beneficiary (UnionSCPool.sol#99) is not in mixedCase
Parameter UnionSCPool.withdrawWithData(uint256,uint256,uint256[5],bytes)._requestID (UnionSCPool.sol#128) is not in mixedCase
Parameter UnionSCPool.withdrawWithData(uint256,uint256,uint256[5],bytes)._amount (UnionSCPool.sol#128) is not in mixedCase
Parameter UnionSCPool.withdrawWithData(uint256,uint256,uint256[5],bytes)._data (UnionSCPool.sol#128) is not in mixedCase
Parameter UnionSCPool.withdrawWithData(uint256,uint256,uint256[5],bytes)._signature (UnionSCPool.sol#128) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (ContextUpgradeable.sol#28)" inContextUpgradeable (ContextUpgradeable.sol#16-32)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
UnionERC20Pool._updateMCR(uint256,uint256,uint256) (UnionERC20Pool.sol#248-268) uses literals with too many digits:
        - item = (mcrIncrement << 64).add(block.number & 0x0000000000000000000000000000000000000000FFFFFFFFFFFFFFFF) (UnionERC20Pool.sol#262)
UnionERC20Pool._unloadMCRPendingQueue(uint256,uint256) (UnionERC20Pool.sol#270-283) uses literals with too many digits:
        - blockNumber = uint64(item & 0x0000000000000000000000000000000000000000FFFFFFFFFFFFFFFF) (UnionERC20Pool.sol#273)
UnionSCPool.version() (UnionSCPool.sol#18-21) uses literals with too many digits:
        - uint32(1000001) (UnionSCPool.sol#20)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
UnionERC20Pool.__gap (UnionERC20Pool.sol#300) is never used in UnionSCPool (UnionSCPool.sol#9-167)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
UnionERC20Pool.excessLiquidityManagerAddress (UnionERC20Pool.sol#41) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
```

```
INFO:Detectors:
grantRole(bytes32,address) should be declared external:
        - AccessControlUpgradeable.grantRole(bytes32,address) (AccessControlUpgradeable.sol#172-174)
revokeRole(bytes32,address) should be declared external:
        - AccessControlUpgradeable.revokeRole(bytes32,address) (AccessControlUpgradeable.sol#185-187)
renounceRole(bytes32,address) should be declared external:
        - AccessControlUpgradeable.renounceRole(bytes32,address) (AccessControlUpgradeable.sol#203-207)
name() should be declared external:
        - ERC20Upgradeable.name() (ERC20Upgradeable.sol#66-68)
symbol() should be declared external:
        - ERC20Upgradeable.symbol() (ERC20Upgradeable.sol#74-76)
decimals() should be declared external:
        - ERC20Upgradeable.decimals() (ERC20Upgradeable.sol#91-93)
transfer(address,uint256) should be declared external:
        - ERC20Upgradeable.transfer(address,uint256) (ERC20Upgradeable.sol#117-120)
allowance(address,address) should be declared external:
        - ERC20Upgradeable.allowance(address,address) (ERC20Upgradeable.sol#125-127)
approve(address,uint256) should be declared external:
        - ERC20Upgradeable.approve(address,uint256) (ERC20Upgradeable.sol#136-139)
transferFrom(address,address,uint256) should be declared external:
        - ERC20Upgradeable.transferFrom(address,address,uint256) (ERC20Upgradeable.sol#154-162)
increaseAllowance(address,uint256) should be declared external:
        - ERC20Upgradeable.increaseAllowance(address,uint256) (ERC20Upgradeable.sol#176-179)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20Upgradeable.decreaseAllowance(address,uint256) (ERC20Upgradeable.sol#195-201)
setPoolReserveAddress(address) should be declared external:
        - UnionERC20Pool.setPoolReserveAddress(address) (UnionERC20Pool.sol#110-113)
setFoundationReserveAddress(address) should be declared external:
        - UnionERC20Pool.setFoundationReserveAddress(address) (UnionERC20Pool.sol#115-118)
setPoolReservePremiumCommission(uint8,uint8) should be declared external:
        - UnionERC20Pool.setPoolReservePremiumCommission(uint8,uint8) (UnionERC20Pool.sol#120-124)
setFoundationReservePremiumCommission(uint8,uint8) should be declared external:
        - UnionERC20Pool.setFoundationReservePremiumCommission(uint8,uint8) (UnionERC20Pool.sol#126-130)
setPoolReserveExcessLiquidityCommission(uint8,uint8) should be declared external:
        - UnionERC20Pool.setPoolReserveExcessLiquidityCommission(uint8,uint8) (UnionERC20Pool.sol#132-136)
setFoundationReserveExcessLiquidityCommission(uint8,uint8) should be declared external:
        - UnionERC20Pool.setFoundationReserveExcessLiquidityCommission(uint8,uint8) (UnionERC20Pool.sol#138-142)
pause() should be declared external:
        - UnionERC20Pool.pause() (UnionERC20Pool.sol#147-149)
unpause() should be declared external:
        - UnionERC20Pool.unpause() (UnionERC20Pool.sol#153-155)
withdrawPoolReserveCommission(uint256) should be declared external:
        - UnionERC20Pool.withdrawPoolReserveCommission(uint256) (UnionERC20Pool.sol#158-163)
withdrawFoundationReserveCommission(uint256) should be declared external:
        - UnionERC20Pool.withdrawFoundationReserveCommission(uint256) (UnionERC20Pool.sol#165-170)
flushMCRPendingQueue(uint256,uint256[2],bytes) should be declared external:
        - UnionERC20Pool.flushMCRPendingQueue(uint256,uint256[2],bytes) (UnionERC20Pool.sol#190-205)
getBasicToken() should be declared external:
        - UnionERC20Pool.getBasicToken() (UnionERC20Pool.sol#207-209)
getBasicTokenDecimals() should be declared external:
        - UnionERC20Pool.getBasicTokenDecimals() (UnionERC20Pool.sol#212-214)
getWriterData(address) should be declared external:
        - UnionERC20Pool.getWriterData(address) (UnionERC20Pool.sol#224-226)
getTotalValueLocked() should be declared external:
        - UnionERC20Pool.getTotalValueLocked() (UnionERC20Pool.sol#234-236)
getPoolStat() should be declared external:
        - UnionERC20Pool.getPoolStat() (UnionERC20Pool.sol#238-246)
```

```
getWriterDataExtended(address) should be declared external:
        - UnionSCPool.getWriterDataExtended(address) (UnionSCPool.sol#36-38)
initialize(address,address,bytes32,address,string) should be declared external:
        - UnionSCPool.initialize(address,address,bytes32,address,string) (UnionSCPool.sol#40-46)
onProtectionPremium(address,uint256[7]) should be declared external:
        - UnionSCPool.onProtectionPremium(address,uint256[7]) (UnionSCPool.sol#58-70)
unlockPremium(uint256[]) should be declared external:
        - UnionSCPool.unlockPremium(uint256[]) (UnionSCPool.sol#72-97)
ppID() should be declared external:
        - UnionSCPool.ppID() (UnionSCPool.sol#111-113)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:UnionSCPool.sol analyzed (26 contracts with 72 detectors), 150 result(s) found
```