SANCTUM

SMART
CONTRACT
AUDIT

27 JUL, 2021

PASS

Sanctum's Security Team has confirmed that this smart contract passes security audit to be listed on digital asset exchanges.

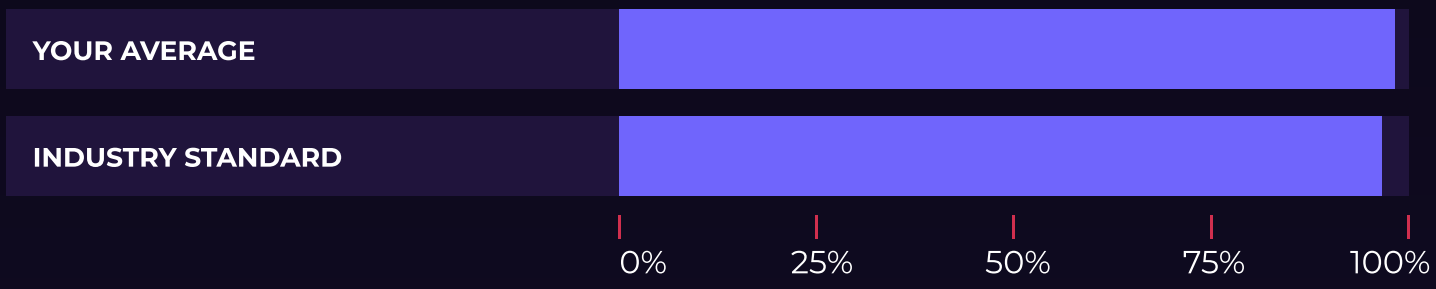
TECHNICAL SUMMARY

This document describes the overall security of the Union Protocol smart contracts, evaluated by Sanctum's Blockchain Security team.

The scope of this audit was to determine quality, security, and correctness of Union Protocol smart contract codebase.

Contract Status

Testable Code



Sanctum Blockchain Security team has evaluated the test coverage of the codebase and has implemented own set of tests to complete the testable code coverage. The testable code is 97%, which is above the industry standard of 95%.

In order to ensure a security of the contract we at Sanctum recommend that the Union Protocol team put in place a bug bounty program to encourage further and active analysis of the smart contract.

TABLE OF CONTENTS

AUDITING STRATEGY AND TECHNIQUES APPLIED

The Union Protocol smart contract's source code was taken from the repository provided by the Union Protocol team.
Repository:
https://github.com/UNIONProtocolFoundation/union-protocol-sc-protection
Commit (initial): 670bc5f9a07d594c50b7cc47750d898da2dadf51
Commit (post-audit): 2a2f1f989bb2874fca9b1b83bcf131cd8880a6e5

Within the scope of this audit Sanctum auditors have reviewed the following contract(s):

- SCPClaims.sol
- UnionSCPool.sol
- UpgradeableBeacon.sol
- SCProtections.sol
- BeaconProxy.sol

The objectives of the audit are as follows:

Determine correct functioning of the contract, in accordance with the specification;
Check the contract code against the standard list of vulnerabilities;
Check the code quality and following best practices;
Determine contract bugs and unexpected behavior;
Check the documentation and verify testable code;
Make recommendations to improve code safety and readability.

Sanctum's Security Team has followed best practices and industry-standard techniques to verify the implementation of Union Protocol smart contracts. To do so, the code is manually reviewed line-by-line by our smart contract developers, together with verification by automated Solidity analysis tools. Audit scope includes writing a unit test suite. In summary, audit strategy consist of several steps tha require manual collaboration between multiple team members at each stage of the review:

1	Assessment of the overall code quality of the codebase.
2	Thorough, manual review of the codebase, line-by-line.
3	Interpretation and verification of the results of static analysis tools.
4	Full test coverage of contract logic, including tests against common and uncommon attack vectors.

EXECUTIVE SUMMARY

There were several criticql, high and medium issues found during the audit. These issues were related to:

- “Deadlock” cycle (successfully verified by the Union team)
- Double token spending
- Potential reentrancy.

At the same time several low and informational issue were also detected during secutiry audit, but they don’t influence smart contract operations in a significant way and may occur only during certain terms appearance.

Nevertheless, most of mentioned findings were successfully fixed by the Union Protocol team. Though, the code is purely documented, has issues with coding style and overall quality. Thus, the entanglement of the code can affect further development and create a circumstance in contracts set usage.

STRUCTURE AND ORGANIZATION OF DOCUMENT

For ease of navigation, sections are arranged from most critical to least critical. Issues are tagged “Resolved” or “Unresolved” depending on whether they have been fixed or addressed. Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

Critical

The issue is dangerous for users, leads to significant lost of funds or stops proper work of the contract. Needs immediate improvements and further check.

High

The issue is easy to exploit, may provide an attacker with full control of the affected systems, also may lead to significant data loss or downtime.

Medium

The issue affects the ability of the contract to operate correctly or influences its behavior. Is much harder to exploit, or has low chance of the exploit.

Low

The issue has minimal impact on the contract’s ability to operate.

Informational

The issue has no impact on the contract’s ability to operate and is connected with code style or optimization.

CRITICAL VERIFIED FUNCTIONALITY

“Deadlock” cycle.
When you call createTo () line 132 (SCProtections.sol), _updateMCR () is called which cannot be executed due to totalCap = 0 and cannot be changed until createTo () is executed. Vicious circle.

Recommendation:
Review the functionality, fix the “deadlock” and/or add separate setter on the protections pool (protections mapping).

Post-audit:
The issue was verified by Union team with provided test script. Though, due to not obvious solution it is recommended to provide proper documentation.

HIGH RESOLVED

Double token spending.
SCPClaims.sol, claimFeeRefund()
The contract does not clear the storage and does not change the status for;
claims[_ppID][_timestamp].claimPayments[msg.sender]
So, claim can performed multiple times (especially with Reentrancy mode)

Recommendation:
Fix the storage and provide reentrancyGuard measures.

MEDIUM RESOLVED

Potential reentrancy.
SCPClaims.sol
fillClaim()
fillClaimForPool()
claimFeeRefund()
distributeClaimFee()
setClaimAppeal()
These functions contain calls to external contract - the pool basic token. Since the token is not validated in the system, there is a risk of reentrancy. Consider providing reentrancy measures.

Recommendation:
Provide all storage changes before the external call, check the correct clearing of the storage, use ReentrancyGuard.

MEDIUM UNRESOLVED

Fix solidity version.
Use the single fixed version for all contracts.
Currently several contract use diapason version instead of the single fixed one. It is recommended to use one of the latest stable versions like 0.7.6 or 0.8.4. Incorrect solc versioning is included into the standard auditor's vulnerabilities list.
Contracts in:
Interfaces directory (version >= 0.6.2 < 0.8.0)
Upgradeable directory (version >=0.6.0 < 0.8.0)
Core directory (version >=0.6.12)

Recommendation:
Consider usage of a single fixed version.

LOW RESOLVED

Unused parameters.
SCProtections.sol:249
Local variables payoutPercent and claimPool are not used and can be removed for gas usage optimization and code quality increasement.

Recommendation:
Remove unused variables.

LOW RESOLVED

Useless return interface.
SCPProtections.sol, setClaimLock(), releaseClaimLock()
Functions always return true value, thus the “return” interface is useless and can be removed.

Recommendation:
Remove unused Remove “return” interface..

LOW UNRESOLVED

Functions to be declared external.
- SCPClaims.version() (SCPClaims.sol#45-48)
- SCPClaims.initialize(address,address) (SCPClaims.sol#50-61)
- SCPClaims.setClaimingParams(uint64,uint64,uint64) (SCPClaims.sol#79-83)
- SCPClaims.setProtectionPayoutPeriod(uint64) (SCPClaims.sol#86-88)
- SCPClaims.setChallengePeriod(uint64) (SCPClaims.sol#90-92)

Functions to be declared external (continue)

- SCPClaims.setFeesCollectorAddress(address) (SCPClaims.sol#94-96)
- SCPClaims.pause() (SCPClaims.sol#102-104)
- SCPClaims.unpause() (SCPClaims.sol#108-110)
- SCPClaims.fillClaim(bytes32,uint64) (SCPClaims.sol#112-117)
- SCPClaims.fillClaimForPool(address,uint64) (SCPClaims.sol#119-123)
- SCPClaims.claimFeeRefund(bytes32,uint64) (SCPClaims.sol#125-132)
- SCPClaims.distributeClaimFee(bytes32,uint64) (SCPClaims.sol#134-148)
- SCPClaims.setClaimInReview(bytes32,uint64) (SCPClaims.sol#150-157)
- SCPClaims.setClaimApproved(bytes32,uint64,uint8) (SCPClaims.sol#159-169)
- SCPClaims.setClaimRejected(bytes32,uint64) (SCPClaims.sol#171-179)
- SCPClaims.setClaimAppeal(bytes32,uint64) (SCPClaims.sol#181-196)
- SCPClaims.setClaimAppealRejected(bytes32,uint64) (SCPClaims.sol#198-206)
- SCPClaims.releaseLockOnExpiredClaim(bytes32,uint64) (SCPClaims.sol#208-214)
- SCPClaims.getNextClaimFillPayment(bytes32,uint64) (SCPClaims.sol#216-222)
- SCPClaims.getClaimData(bytes32,uint64) (SCPClaims.sol#271-273)
- SCPClaims.getClaiingParams() (SCPClaims.sol#280-282)
- SCProtections.initialize(address,address,address) (SCProtections.sol#68-76)
- SCProtections.pause() (SCProtections.sol#89-91)
- SCProtections.unpause() (SCProtections.sol#95-97)
- SCProtections.create(uint256[9],bytes,uint256) (SCProtections.sol#113-115)
- SCProtections.setActiveSCProtectionPoolAddress(bytes32,address) (SCProtections.sol#176-179)
- SCProtections.setMaxActiveClaimsPerPool(uint8) (SCProtections.sol#181-183)
- SCProtections.exercise(uint256,uint64) (SCProtections.sol#189-211)
- SCProtections.setClaimLock(address,uint64) (SCProtections.sol#213-218)
- SCProtections.releaseClaimLock(address,uint64) (SCProtections.sol#220-240)
- SCProtections.isClaimLocked(address) (SCProtections.sol#242-259)
- SCProtections.getProtectionData(uint256) (SCProtections.sol#273-282)

Recommendation:

Declare functions as external for gas usage optimization and code quality increasement.

Import of external files.
SCProtections.sol, line 9, 11,
import "../../union-protocol-oc-protection/contracts/libraries/SignLib.sol";
import
"../../union-protocol-oc-protection/contracts/interfaces/IUUNNRegistry.sol"
;
UnionSCPool.sol line 6
import "../../union-protocol-oc-protection/contracts/UnionERC20Pool.sol";
uUNNToken.sol line 5
import
"../../union-protocol-oc-protection/contracts/interfaces/IUUNNRegistry.sol"
;
ISCPool.sol, line 5
import "../../../union-protocol-oc-protection/contracts/interfaces/IPool.sol";
The project is importing files that are provided in the external source. Such approach decreases the readability and overall quality of the code. It affects further development and may lead to type or manual errors.

Recommendation:
Check if the missing files are imported and processed correctly. Use interfaces, or add external contract as a separate package or module. Or add missing contracts to the directory.

LOW RESOLVED

Unused Missing check in protections getter..
Calling getProtectionData () line 282 (SCProtections.sol) before createTo () is executed, an error is thrown, trying to access a pool that has not yet been added.

Recommendation:
Add correct return for the non-existent pool or add “require” statement for the non-existent pool.

LOW VERIFIED FUNCTIONALITY

Imports non-node_moduls libraries.
In all project
import
../../union-protocol-oc-protection/openzeppelin-contracts-upgradeable.

Recommendation:
Download standard libraries from "npm" or trusted sources. Code from other sources may contain malicious code. Verify, that third party contract are imported from standard OpenZeppelin library.

LOW

RESOLVED

No check for 0 address.
SCPClaims.sol, line 49, initialize(address _admin, address _scProtections)
SCPClaims.sol, line 86, setFeesCollectorAddress(address _target)
SCProtections.sol line65 initialize(address _admin, address _uunn, address _claimStorage)
UnionRouter.sol line 17 initialize(address admin)
UnionSCPool.sol line 40 initialize()

0 address can be specified as values.

Recommendation:
Add check to 0 address.

INFORMATIONAL

RESOLVED

Missing getter.
SCPClaims.sol, line 25.
Variable claimAppealAmount has no getter and cannot be read outside of the contract.

Recommendation:
Consider updating the solidity version.

INFORMATIONAL RESOLVED

Functions can be restricted to pure.
SCPClaims.sol:45, version()
uUNNToken.sol:32, version()

Recommendation:
Restrict

INFORMATIONAL UNRESOLVED

Upgrade solidity version.
The solidity version may be updated.
Currently used version (0.6.12) is stable and may be used in production.
Though, it is recommended to use one of the latest version like 0.7.6 or 0.8.4. Also, Solidity 0.8.x has built-in support for reverts for overflows and underflows, which will allow it to omit the SafeMath library.

Recommendation:
Consider updating the solidity version.

	SCPClaims	SCProtections	UnionSCPool
Re-entrancy	Pass	Pass	Pass
Access Management Hierarchy	Pass	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass	Pass
Delegatecall Unexpected Ether	Pass	Pass	Pass
Default Public Visibility	Pass	Pass	Pass
Hidden Malicious Code	Pass	Pass	Pass
Entropy Illusion (Lack of Randomness)	Pass	Pass	Pass
External Contract Referencing	Pass	Pass	Pass
Short Address/ Parameter Attack	Pass	Pass	Pass
Unchecked CALL Return Values	Pass	Pass	Pass
Race Conditions / Front Running	Pass	Pass	Pass
General Denial Of Service (DOS)	Pass	Pass	Pass
Uninitialized Storage Pointers	Pass	Pass	Pass
Floating Points and Precision	Pass	Pass	Pass
Tx.Origin Authentication	Pass	Pass	Pass
Signatures Replay	Pass	Pass	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass	Pass

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests written by Sanctum Secured team

As part of audit scope for Union Protocol, our team was responsible for the completness of unit test coverage and writing integration tests using Truffle testing framework. Tests were based on the reviewed functionality of the code and available requirements.

Contract: SCPClaims

Modifier testing.

- Should throw an error if the sender role is missing

Setter testing.

- Should set Claiming Params
- Should set Protection Payout Period
- Should set Challenge Period
- Should set Fees Collector Address

Pause testing

- Should pause/unpause contract

Claim testing

- Should fill Claim
- Should fill Claim For Pool
- Should claim Fee Refund
- Should claim Fee Refund with other parameters
- Should distribute Claim Fee
- Should set Claim In Review
- Should set Claim Approved
- Should set Claim Rejected
- Should set Claim Appeal
- Should set Claim Appeal Rejected
- Should claim Fee Refund with other parameters
- Should release Lock On Expired Claim
- Should fill Claim with other parameters
- Should fill Claim with other parameters

Getter testing

- Should get version
- Should get Next Claim Fill Payment
- Should get Claim Data
- Should get Claiming Params

Contract: SCProtections

- Initialize testing
  - ✓ Should initialize contract
- Pause testing
  - ✓ Should pause/unpause contract
- Create testing
  - ✓ Should create protection or give an error if the parameters are incorrect
  - ✓ Should withdrawPremium or give an error if the parameters are incorrect
  - ✓ Should set ActiveSCProtection Pool Address
  - ✓ Should set Max Active Claims Per Pool
  - ✓ Should get Active SCProtection Pool
  - ✓ Should exercise or give an error if the parameters are incorrect
  - ✓ Should set Claim Lock or give an error if the parameters are incorrect
  - ✓ Should check isClaimLocked or give an error if the parameters are incorrect
  - ✓ Should set Document or give an error if the parameters are incorrect
  - ✓ Should remove Document or give an error if the parameters are incorrect
  - ✓ Should get Document or give an error if the parameters are incorrect
  - ✓ Should get All Documents or give an error if the parameters are incorrect

Contract: UnionSCPool

- ✓ Initialization test
- Get data test
  - ✓ Should be able to get version
  - ✓ Should be able to get pool type
  - ✓ Should be able to get writer data
  - ✓ Should be able to get pp id
- Set lockup period test
  - ✓ Admin should be able to set lockup period
  - ✓ Third party shouldn't be able to set lockup period
- Premium test
  - ✓ SC protection should be able to use protection premium
  - ✓ SC protection shouldn't be able to use protection premium with invalid premium
  - ✓ SC protection shouldn't be able to use protection premium with invalid validTo
  - ✓ SC protection shouldn't be able to use protection premium with invalid amount
  - ✓ Third party shouldn't be able to use protection premium
  - ✓ Should be able to unlock premium
  - ✓ Shouldn't be able to unlock too much
- Payout coverage test
  - ✓ Should be able to use payout coverage

Withdraw with data test

-
- Shouldn't be able to withdraw with locked up withdrawal
- Shouldn't be able to withdraw with invalid data signature
- Shouldn't be able to withdraw with incorrect data package (_requestID)
- Shouldn't be able to withdraw with incorrect data package (_amount)
- Shouldn't be able to withdraw with quotation expired

Before token transfer test

- shouldn't be able to transfer with locked up withdrawal
- Shouldn't be able to transfer with currently locked pool by active claim

After deposit test

- Should be able to verify deposit

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	UNCOVERED LINES
SCPClaims.sol	98.00	95.00	98.00	100.00	
SCProtections.sol	98.00	95.00	98.00	100.00	
UnionSCPool.sol	96.00	95.00	98.00	100.00	
All files	**97**	**95**	**98**	**100**	

We are grateful to have been given the opportunity to work with the Union Protocol team.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them.

Sanctum's Security Team recommends that the Unioт Protocol team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

SANCTUM