Union Protocol

Security Assessment

December 10th, 2020

For :
Union Protocol

By :
Alex Papageorgiou @ CertiK
alex.papageorgiou@certik.org

Georgios Delkos @ CertiK
georgios.delkos@certik.io

Disclaimer

CertiK reports are not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts CertiK to perform a security review.

CertiK Reports do not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

CertiK Reports should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

CertiK Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK’s position is that each company and individual are responsible for their own due diligence and continuous security. CertiK’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

What is a CertiK report?

- A document describing in detail an in depth analysis of a particular piece(s) of source code provided to CertiK by a Client.
- An organized collection of testing results, analysis and inferences made about the structure, implementation and overall best practices of a particular piece of source code.
- Representation that a Client of CertiK has indeed completed a round of auditing with the intention to increase the quality of the company/product's IT infrastructure and or source code.

Overview

Project Summary

Project Name	Union Protocol
Description	An open Platform lowering DeFi Access, Cost, and Risk Barriers.
Platform	Ethereum; Solidity, Yul
Codebase	GitHub Repository
Commits	1. c1f5aaea93d550e153ad8238102d6c2bee5011ae

Audit Summary

Delivery Date	December 10th, 2020
Method of Audit	Static Analysis, Manual Review
Consultants Engaged	2
Timeline	November 23rd, 2020 - December 10th, 2020

Vulnerability Summary

Total Issues	21
Total Critical	0
Total Major	2
Total Medium	2
Total Minor	6
Total Informational	11

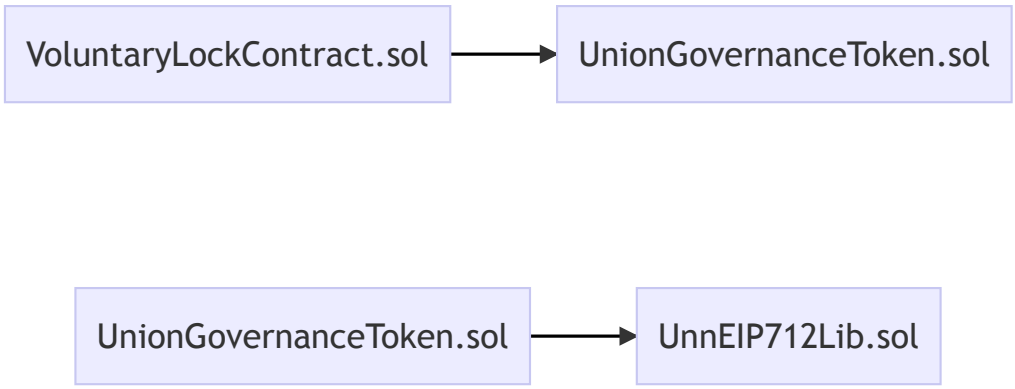
Executive Summary

The code examined by the team contained certain design decisions that redundantly convoluted the system as well as certain behavioral discrepancies that were disclosed to the Union team. The Union team proceeded to partially remediate the action items we listed, however, the codebase contains duplicated statements in a lot of situations and redundant conditionals on top of the utilization of safe libraries that only increase the illegibility of the codebase rather than increase its security. When we approached the Union team with regards to this issue, they maintained that the codebase conforms to a security practice whereby the behavior of libraries, i.e. SafeMath, is unknown and unreliable, and as such, additional conditionals are imposed on the inputs of any particular function to ensure consistency in the outputs regardless of the library or even implementation utilized. All security-related exhibits were promptly dealt with and any informational exhibits that were raised during the audit merely serve as notices and can optionally be ignored as they do not affect the overall security of the project.

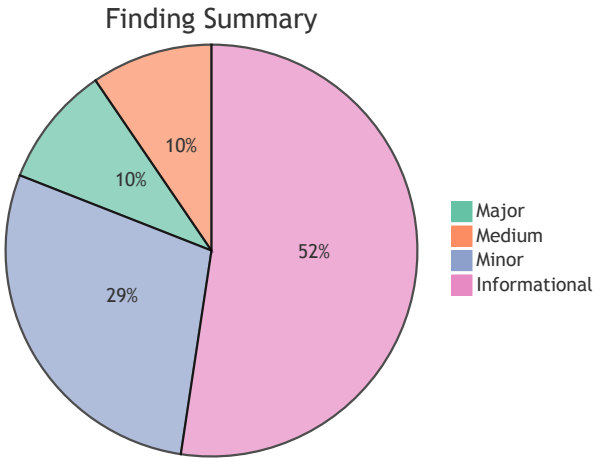
Files In Scope

ID	Contract	Location
UGT	UnionGovernanceToken.sol	contracts/UnionGovernanceToken.sol
UEI	UnnEIP712Lib.sol	contracts/UnnEIP712Lib.sol
VLC	VoluntaryLockContract.sol	contracts/VoluntaryLockContract.sol
DTE	DateTime.sol	contracts/util/DateTime.sol

File Dependency Graph (BETA)



Findings



ID	Title	Type	Severity	Resolved
DTE-01	Inproper Implementation	Inconsistency	Informational	✓
UEI-01	Incompatibility with EIP2612	Language Specific	Major	✓
UEI-02	Oz Library Code Difference	Language Specific	Informational	✓
UGT-01	EIP712 Domain Separator Vulnerable	Language Specific	Medium	✓
UGT-02	DoS Due To Loop Iteration	Language Specific	Medium	✓
UGT-03	Unsafe Addition	Mathematical Operations	Major	✓

UGT-04	Check Missing	Logical Issue	Minor	✓
UGT-05	Ilegal Transfer	Control Flow	Minor	✓
UGT-06	Non Complying Implementation	Inconsistency	Minor	✓
UGT-07	Undocumented Functionality	Inconsistency	Minor	✓
UGT-08	Unnecessary Check	Control Flow	Informational	✓
UGT-09	Redundant Code	Control Flow	Informational	✓
UGT-10	Redundant Code	Control Flow	Informational	✓
UGT-11	Improper Nonce Use	Language Specific	Minor	✓
UGT-12	Inconsistant Implementation	Inconsistency	Informational	✓
UGT-13	Redundant Checks	Mathematical Operations	Informational	✓
UGT-14	Require Missing Error Messages	Language Specific	Informational	✓
UGT-15	Action Order	Coding Style	Informational	✓
VLC-01	Logic Error	Logical Issue	Minor	✓
VLC-02	Numbers	Language Specific	Informational	✓
VLC-03	Set Variables to Imutable and lock version 0.6.12	Language Specific	Informational	✓

DTE-01: Inproper Implementation

Type	Severity	Location
Inconsistency	Informational	DateTime.sol L1

Description:

The contract states that it is inspired by ethereum-datetime, while it is a 1 to 1 copy from the original with the addition of a linter applied and a improper conversion from `contract` to `library`.

Recommendation:

Refactor the code to the original implementation.

Alleviation:

Ignored Initially developed using the ethereum-datetime library as a reference, it's not a direct c/p. The team declared it as a library to ease import, but this approach is not believed to have an impact.

UEI-01: Incompatibility with EIP2612

Type	Severity	Location
Language Specific	Major	UnnEIP712Lib.sol L38, L62

Description:

The structure of `PERMIT_TYPEHASH` differs from the one expected for no apparent reason. This will cause compatibility issues.

Recommendation:

Refactor the code with regards to the expected structure of the standard as described. `PERMIT_TYPEHASH` https://eips.ethereum.org/EIPS/eip-2612.

Alleviation:

Fixed by the team in commit 29f4374b67feb5de6c5cd660fabb4e59cc628c95.

UEI-02: Oz Library Code Difference

Type	Severity	Location
Language Specific	Informational	UnnEIP712Lib.sol L118

Description:

Function `_recoverSigner` differs from the OZ implementation for no apparent reason

Recommendation:

Refactor to the OZ implementation or document a rationale about the reason of this modification.

Alleviation:

Ignored We borrowed our impl from oz, we explicitly check to make sure that the signer is not 0x0. This could arguably be done outside the function, but we also took time to understand the implementation here, and didn't want to look like we were assembling without thought. We make heavy use of oz conventions elsewhere.

UGT-01: EIP712 Domain Separator Vulnerable

Type	Severity	Location
Language Specific	Medium	UnionGovernanceToken.sol L114

Description:

The EIP712 Domain Separator is susceptible to fork replay attacks

Recommendation:

Refactor the code with respect to the chain id.

Alleviation:

Fixed by the team in commit 29f4374b67feb5de6c5cd660fabb4e59cc628c95.

UGT-02: DoS Due To Loop Iteration

Type	Severity	Location
Language Specific	Medium	UnionGovernanceToken.sol L343, L365, L405, L1008

Description:

Function `_calculateLockedBalance`, `_calculateReleasedBalance`, `_moveReleasedBalance`, `_calculateVotingPower` are susceptible to DoS due to loop iteration

Recommendation:

We advise these pieces of code are re-evaluated and refactored to not require loop iterations as the current implementation is prone to out-of-gas errors.

Alleviation:

The Union Protocol development team is fully aware of the issue raised and is experimenting for a solution, however any feasible solution would require an overhaul of the logic of the system and as such the team has decided to stick with the current implementation until further notice.

UGT-03: Unsafe Addition

Type	Severity	Location
Mathematical Operations	Major	UnionGovernanceToken.sol L343

Description:

The code uses unsafe addition while safe math is available.

Recommendation:

Refactor the code using the safe math addition.

Alleviation:

Fixed by the team in commit 29f4374b67feb5de6c5cd660fabb4e59cc628c95.

UGT-04: Check Missing

Type	Severity	Location
Logical Issue	Minor	UnionGovernanceToken.sol L468

Description:

The code does not properly implement a race condition protection.

Recommendation:

Function `_approveUNN` should ensure that either `m_allowances[_owner][_spender] == 0` or `_value == 0` as that's how the race condition protection is meant to be implemented.

Alleviation:

Fixed by the team in commit 29f4374b67feb5de6c5cd660fabb4e59cc628c95.

UGT-05: Ilegal Transfer

Type	Severity	Location
Control Flow	Minor	UnionGovernanceToken.sol L609, L705

Description:

Function `_transferUNN` and `_transferFromUNN` allow transfers that have been frozen to a locked destination even though the funds aren't `locked`.

Recommendation:

Refactor the code and remove it.

Alleviation:

The Union Protocol development team has informed us that this is desired functionality and as such, the codebase should remain as is.

UGT-06: Non Complying Implementation

Type	Severity	Location
Inconsistency	Minor	UnionGovernanceToken.sol L1154

Description:

Function `_writeVotingCheckpoint` is not complying with Compound implementation and is using full `block.number` instead of a casted `uint32`.

Recommendation:

Refactor to match the specifications of the Compound implementation.

Alleviation:

Ignored As far as we know EVM is casting each uint/int type to uint256/int256 anyway, so we don't grasp the need to use int32. Perhaps for optimization, but code working in current form.

UGT-07: Undocumented Functionality

Type	Severity	Location
Inconsistency	Minor	UnionGovernanceToken.sol L1247

Description:

Function `_delegateVote` undocumented functionality of removing votes.

Recommendation:

Document this functionality as it is crucial to the overall system.

Alleviation:

The Union Protocol development team has defined that this feature will be closely integrated with their front end systems and as such no documentation is necessary within the codebase

UGT-08: Unnecessary Check

Type	Severity	Location
Control Flow	Informational	UnionGovernanceToken.sol L496

Description:

Check `m_allowances[_owner][_spender] >= _subtractedValue` is not needed due to safe math usage in line 498.

Recommendation:

Refactor the code and remove this check.

Alleviation:

Ignored Belt and suspenders never hurt. Explicit condition checking is ameliorated by the use of SafeMath, but expectation of the condition should set a reasonable boundary to prevent the issue apart from use of the SafeMath library.

UGT-09: Redundant Code

Type	Severity	Location
Control Flow	Informational	UnionGovernanceToken.sol L614

Description:

Check (_value >= 0) will never yield false so its redundant.

Recommendation:

Refactor the code and remove this check.

Alleviation:

Ignored Belt and suspenders never hurt. Explicit condition checking. Given use of a uint value, this will never be 0. Believe it to be legacy from before the use of uint in the function. No penalty to keeping an explicit reminder that we only address positive values.

UGT-10: Redundant Code

Type	Severity	Location
Control Flow	Informational	UnionGovernanceToken.sol L613, L619, L621, L670, L672, L716, L718, L768, L770

Description:

The code assigns `true` while it could use a variable.

Recommendation:

Refactor the code and just assign `_moveVotingDelegates`.

Alleviation:

Ignored The construct used of retval = retval && _moveVotingDelegates separates the native return value of the function from the return value of the subordinate function. This is the logically correct way to handle a boolean return where the subordinate function is not the complete determinator of boolean status.

UGT-11: Improper Nonce Use

Type	Severity	Location
Language Specific	Minor	UnionGovernanceToken.sol L1331, L1374

Description:

Functions `permitAllocationBySignature` and `delegateVoteBySignature` increment the `m_nonces` before using it, being incompatible with the expected behavior of every cryptographic system including ethereum.

Recommendation:

Refactor the code and make proper use of the nonce.

Alleviation:

Fixed by the team in commit 29f4374b67feb5de6c5cd660fabb4e59cc628c95.

UGT-12: Inconsistant Implementation

Type	Severity	Location
Inconsistency	Informational	UnionGovernanceToken.sol L1060

Description:

Function `_moveVotingDelegates` differs from Compound implementation for no reason

Recommendation:

Refactor the code to the original implementation.

Alleviation:

Ignored UNN followed the Compound implementation for guidance in its own approach, but we don't specifically believe deviating from reference is any reason for claims of 'inconsistent implementation'. The Compound implementation is not a cannonical in the same sense as OZ. This code has been implemented and tested on its own merit.

UGT-13: Redundant Checks

Type	Severity	Location
Mathematical Operations	Informational	UnionGovernanceToken.sol L1

Description:

Almost all checks are redundant as SafeMath is being utilized

Recommendation:

The team needs to refactor the code based on the securities that safe math provides.

Alleviation:

Ignored Implicit contracts between functions are sloppy at best and hard to manage over successive generations of code, at worst. Explicit condition checking is ameliorated by the use of SafeMath, but expectation of the condition should set a reasonable boundary to prevent the issue apart from use of the SafeMath library. A function's parameters should form an explicit contract with its operators. A function receiving parameters from another function should never assume their fitness. A function calling another function should call that function defensively, in not assuming the fitness of results.

UGT-14: Require Missing Error Messages

Type	Severity	Location
Language Specific	Informational	UnionGovernanceToken.sol L1

Description:

Require checks are missing error messages.

Recommendation:

Refactor and inlcude a error message with every require check.

Alleviation:

Fixed by the team in commit 29f4374b67feb5de6c5cd660fabb4e59cc628c95.

UGT-15: Action Order

Type	Severity	Location
Coding Style	Informational	UnionGovernanceToken.sol L480

Description:

Code emit and changes a state.

Recommendation:

Refactor and first make the state change and then emit the event.

Alleviation:

Ignored retval=true, emit or emit retval=true have no explicit impact on return result of the code or the emitted value of the event, as the retval being sent is not part of the emitted event.

VLC-01: Logic Error

Type	Severity	Location
Logical Issue	Minor	VoluntaryLockContract.sol L56

Description:

Check `require(_amount <= unnToken.balanceOf(interestWallet), 'amount too big');` is incorrect as line 68 `unnToken.transferFromAndLock(interestWallet, msg.sender, interest, releaseTime, false);` would throw if there are insufficient interest funds. It also prevents legitimate deposits.

Recommendation:

Refactor the functionality as the correct conditional would be `interest <= unnToken.balanceOf(interestWallet)`.

Alleviation:

Fixed by the team in commit 29f4374b67feb5de6c5cd660fabb4e59cc628c95.

VLC-02: Numbers

Type	Severity	Location
Language Specific	Informational	VoluntaryLockContract.sol L34, L36

Description:

Numbers specified up to 32 decimal precision when the benefit is negligible.

Recommendation:

Refactor and add documentation regarding the numbers.

Alleviation:

Ignored Higher precision than required, but working in testing, so not expected to have negative impact.

VLC-03: Set Variables to Imutable and lock version 0.6.12

Type	Severity	Location
Language Specific	Informational	VoluntaryLockContract.sol L1

Description:

Variables can be set to `immutable` as project uses `0.6.12` sol version, which should also be locked at for all contracts.

Recommendation:

Refactor the code and take advantage of the versions optimisations.

Alleviation:

Fixed by the team in commit 29f4374b67feb5de6c5cd660fabb4e59cc628c95.

Appendix

Finding Categories

Gas Optimization

Gas Optimization findings refer to exhibits that do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Mathematical Operations

Mathematical Operation exhibits entail findings that relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

Logical Issue findings are exhibits that detail a fault in the logic of the linked code, such as an incorrect notion on how `block.timestamp` works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Data Flow

Data Flow findings describe faults in the way data is handled at rest and in memory, such as the result of a `struct` assignment operation affecting an in-memory `struct` rather than an in-storage one.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of `private` or `delete`.

Coding Style

Coding Style findings usually do not affect the generated byte-code and comment on how to make the codebase more legible and as a result easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a `constructor` assignment imposing different `require` statements on the input variables than a setter function.

Magic Numbers

Magic Number findings refer to numeric literals that are expressed in the codebase in their raw format and should otherwise be specified as `constant` contract variables aiding in their legibility and maintainability.

Compiler Error

Compiler Error findings refer to an error in the structure of the code that renders it impossible to compile using the specified version of the project.

Dead Code

Code that otherwise does not affect the functionality of the codebase and can be safely omitted.