Security Assessment

SYNDICATE PROTOCOL

Dec 31st, 2021

Table of Contents

Summary

Overview

- Project Summary
- Audit Summary
- Vulnerability Summary
- Audit Scope

Findings

- SYNDICATE PROTOCOL-01 : Financial Models
- SYNDICATE PROTOCOL-02 : Centralization Risk
- SCS-01 : Lack of validation for `poolToken`
- SCS-02 : The `amount` may not be accurate
- SER-01 : Initial token distribution
- SER-02 : Inconsistent Comments and Code
- SER-03 : Potential Over Mint
- SER-04 : Not standard implement
- SPB-01 : Lack of Input Validation
- SPB-02 : Lack of updates of `user.subYieldRewards`
- SPF-01 : Missing calculate reward

Appendix

Disclaimer

About

Summary

This report has been prepared for SYNDICATE PROTOCOL to discover issues and vulnerabilities in the source code of the SYNDICATE PROTOCOL project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	SYNDICATE PROTOCOL
Platform	ethereum
Language	Solidity
Codebase	https://github.com/superpowerlabs/syndicate
Commit	9ba4867092d02b8ea436798c0e42abd344a47772

Audit Summary

Delivery Date	Dec 31, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

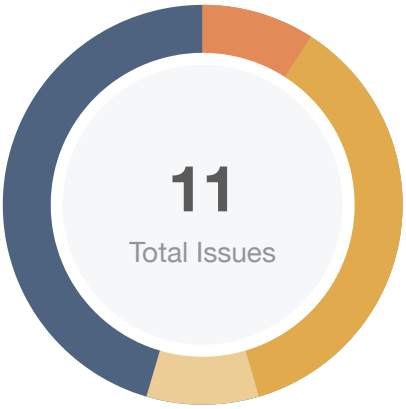
Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	1	0	0	1	0	0
Medium	4	0	0	3	0	1
Minor	1	0	0	0	0	1
Informational	5	0	0	2	0	3
Discussion	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
SAS	pools/SyndicateAware.sol	918297a33a7cbe38f44d1ecb07068bdda088facbab7fa115367e5c410b30e743
SCS	pools/SyndicateCorePool.sol	6ae34aa281b88911bb940babbc83deecab736f880f398bfda0d67080c49361ce
SFP	pools/SyndicateFlashPool.sol	7f23a5dab399097044ba7b039e2e48a47b7de4ece9fcb84b283807a27f34bd58
SPB	pools/SyndicatePoolBase.sol	0131fc7037eab81adca254d02b5f50a424650254111f3a3dbb4bf8ca8617be9b
SPF	pools/SyndicatePoolFactory.sol	944b83a659dd785607e682c86c4097df0a7f67c3421a1b36f53f6b88df598f8e
ERC	token/ERC20Receiver.sol	16cfd09bc6cff4be2877ca80dbac8620a132d0561c60605b82df14a2e4d5894f
ESE	token/EscrowedSyndicateERC20.sol	7d5d4c091d010131f52d9482955173883a863d0213d710342336e49ea2da2faf
SER	token/SyndicateERC20.sol	a9c0b7df7d74d766d480141ffcd0327e1857dcd99cfc27a37d3c41c361a73929

Findings



Critical	**0** (0.00%)
Major	**1** (9.09%)
Medium	**4** (36.36%)
Minor	**1** (9.09%)
Informational	**5** (45.45%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
SYNDICATE PROTOCOL-01	Financial Models	Logical Issue	Informational	Acknowledged
SYNDICATE PROTOCOL-02	Centralization Risk	Volatile Code	Major	Acknowledged
SCS-01	Lack of validation for `poolToken`	Logical Issue	Informational	Resolved
SCS-02	The `amount` may not be accurate	Logical Issue	Informational	Resolved
SER-01	Initial token distribution	**Centralization / Privilege**	**Medium**	Acknowledged
SER-02	Inconsistent Comments and Code	Logical Issue	Minor	Resolved
SER-03	Potential Over Mint	Logical Issue	Medium	Acknowledged
SER-04	Not standard implement	Volatile Code	Informational	Acknowledged
SPB-01	Lack of Input Validation	Control Flow	Informational	Resolved
SPB-02	Lack of updates of `user.subYieldRewards`	Logical Issue	Medium	Acknowledged
SPF-01	Missing calculate reward	Logical Issue	Medium	Resolved

SYNDICATE PROTOCOL-01 | Financial Models

Category	Severity	Location	Status
Logical Issue	● Informational	Global	ⓘ Acknowledged

Description

The main function of `Syndicate` protocol is providing `Syndicate` mining pool.

1. the protocol publishes two `ERC20` token: Syndicate and Escrowed Syndicate.
2. The `SyndicatePoolFactory` can create `SyndicateCorePool` and register it.
3. The `SyndicateFlashPool` can be created by anyone and the `owner` of the `SyndicatePoolFactory` can register it into the `factory`.
4. Each kind of token corresponds to one pool. If one kind of token is registered to a pool, it can not be registered to another pool again.
5. The pool whose `poolToken` is not `Syndicate` will accumulate rewards and stake them to the pool whose `poolToken` is `Syndicate` by the function `stakeAsPool()`. The rewards staked by the function `stakeAsPool()` will participate in the rewards accumulation in the `Syndicate` pool as same as the normal token staked to the pool.
6. Every stake is recorded as the `deposit` in the contract and the `deposit` has an expiry date. User can withdraw their Syndicate from the `deposit`.

And then, there are some questions:

1. Is the pool with `Syndicate` as the `poolToken` a `SyndicateCorePool`. Are those pools whose `poolToken` are not `Syndicate` `SyndicateFlashPool`?
2. The function `_unstake()` checks the `lockUntil` of `deposit`. However, the contract has the function `updateStakeLock` which can change the `lockUntil` of the specified `deposit`. It makes the checks in the `_unstake()` meaningless.

Recommendation

Financial models of blockchain protocols need to be resilient to attacks. They need to pass simulations and verifications to guarantee the security of the overall protocol.

The financial model of this protocol is not in the scope of this audit.

Alleviation

[SYNDICATE PROTOCOL]:

1. The `CorePool` stakes `SYN` or `SYN` related LP token, e.g `SYN`/`ETH LP token` from Uniswap. The `FlashPool` stakes non directly related tokens. The `CorePool` has a locking period, Flash pool does not have a locking period.
2. The `updateStakeLock` only allows updating to a later unlock date.

SYNDICATE PROTOCOL-02 | Centralization Risk

Category	Severity	Location	Status
Volatile Code	Major	Global	Acknowledged

Description

In the contract `SyndicatePoolFactory`, the role `owner` has the authority over the following function:

- `createPool()`: create a `SyndicateCorePool` and register it.
- `registerPool()`: register the specified pool.
- `changePoolWeight()`: chage the weight of the `pool`.

Any compromise to the `owner` account may allow the hacker to take advantage of this.

In the contract `EscrowedSyndicateERC20`, the role `ROLE_RECEIVERS_MANAGER` has the authority over the following function:

- `updateAllowedReceivers()`: decided the user can whether receive the `Escrowed Syndicate`.

In the contract `Syndicate` and `EscrowedSyndicateERC20`, the role `ROLE_TOKEN_CREATOR` has the authority to mint tokens.

In the contract `Syndicate`, the role `ROLE_TOKEN_DESTROYER` has the authority to burn anyone's `Syndicate`.

In the contract `Syndicate`, the role `FEATURE_DELEGATIONS` has the authority to make someone delegate another user.

Recommendation

We advise the client to carefully manage these accounts' private keys to avoid any potential risks of being hacked.

In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;

- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

[SYNDICATE PROTOCOL]: We will use `openzeppelin`'s defender to manage the multi-sig wallet and time locked upgrading. Once the tokens are sufficiently distributed, DAO will decide on important issues.

SCS-01 | Lack of validation for `poolToken`

Category	Severity	Location	Status
Logical Issue	Informational	projects/Syndicate/contracts/pools/SyndicateCorePool.sol (9ab9f5c): 220	Resolved

Description

The function `_stake` lacks validation for the state variable `poolToken`. The variable `poolReserve` is used to record the balance of `SyndicateERC20` token in the contract.

Recommendation

Add the validation.

Alleviation

[SYNDICATE PROTOCOL]: The `poolToken` is an `immutable` state variable set in the `constructor` and validated then.

SCS-02 | The `amount` may not be accurate

Category	Severity	Location	Status
Logical Issue	Informational	projects/Syndicate/contracts/pools/SyndicateCorePool.sol (9ab9f5c): 216	Resolved

Description

If the `poolToken` is a deflationary token, the `amount` may be less than the input number.

Recommendation

Use the balance after the function `stake()` minus the balance before the function `stake()`.

Alleviation

[SYNDICATE PROTOCOL]: Issue is addressed explicitly in `SyndicatePoolBase` L444-450.

SER-01 | Initial token distribution

Category	Severity	Location	Status
Centralization / Privilege	**Medium**	projects/Syndicate/contracts/token/SyndicateERC20.sol (9ab9f5c): 402	Acknowledged

Description

All of the `SyndicateERC20` tokens are sent to the contract deployer when deploying the contract. This could be a centralization risk as the deployer can distribute `SyndicateERC20` tokens without obtaining the consensus of the community.

Recommendation

We recommend the team to be transparent regarding the initial token distribution process, and the team shall make enough efforts to restrict the access of the private key.

Alleviation

[SYNDICATE PROTOCOL]: The contract owner will distribute the token strictly according to schedule and once the tokens are sufficiently distributed, DAO will decide on important issues.

SER-02 | Inconsistent Comments and Code

Category	Severity	Location	Status
Logical Issue	Minor	projects/Syndicate/contracts/token/SyndicateERC20.sol (9ab9f5c): 480	Resolved

Description

```
473 // depending on `FEATURE_UNSAFE_TRANSFERS` we execute either safe (default)
474 // or unsafe transfer
475 // if `FEATURE_UNSAFE_TRANSFERS` is enabled
476 // or receiver has `ROLE_ERC20_RECEIVER` permission
477 // or sender has `ROLE_ERC20_SENDER` permission
478 if(isFeatureEnabled(FEATURE_UNSAFE_TRANSFERS)
479   || isOperatorInRole(_to, ROLE_ERC20_RECEIVER)
480   || isSenderInRole(ROLE_ERC20_SENDER)) {
```

Referring to the L477 comments, the `if` condition should verify the permission of `sender` instead of `msg.sender`.

Recommendation

We recommend verifying the input parameter `_from`.

Alleviation

[SYNDICATE PROTOCOL]: The function is `SendFrom`, which is a third party. It is expected that we check the action executer. Eg `msg.sender`, not the source of fund `_from`.

SER-03 | Potential Over Mint

Category	Severity	Location	Status
Logical Issue	Medium	projects/Syndicate/contracts/token/SyndicateERC20.sol (9ab9f5c): 781 , 20	Acknowledged

Description

```
20  *       - Maximum final token supply: 10,000,000 SYN
```

```
 769  function mint(address _to, uint256 _value) public {
 770    // check if caller has sufficient permissions to mint tokens
 771    require(isSenderInRole(ROLE_TOKEN_CREATOR), "insufficient privileges
(ROLE_TOKEN_CREATOR required)");
 772
 773    // non-zero recipient address check
 774    require(_to != address(0), "ERC20: mint to the zero address"); // Zeppelin msg
 775
 776    // non-zero _value and arithmetic overflow check on the total supply
 777    // this check automatically secures arithmetic overflow on the individual balance
 778    require(totalSupply + _value > totalSupply, "zero value mint or arithmetic
overflow");
 779
 780    // uint192 overflow check (required by voting delegation)
 781    require(totalSupply + _value <= type(uint192).max, "total supply overflow
(uint192)");
```

Refer to the L20 comment `Maximum final token supply: 10,000,000 SYN`. The function `mint` may issue more tokens than the issuance limit.

Recommendation

We recommend that the annotations and the code logic are consistent, and the information about the upper limit of tokens should be public in the white paper.

Alleviation

[SYNDICATE PROTOCOL]: 2^192 is much larger than 10,000,000. We will leave it as an extra sanity check.

[CertiK]: Refer to the function `mint` source code, the actual minting limit is 2^129.

SER-04 | Not standard implement

Category	Severity	Location	Status
Volatile Code	Informational	projects/Syndicate/contracts/token/SyndicateERC20.sol (9ab9f5c): 289~297	Acknowledged

Description

The `DOMAIN_TYPEHASH` is different from the implement in: https://eips.ethereum.org/EIPS/eip-712#rationale-for-typehash.

- `string name` - the user-readable name of signing domain, i.e. the name of the DApp or the protocol.
- `string version` - the current major version of the signing domain. Signatures from different versions are not compatible.
- `uint256 chainId` - the EIP-155 chain id. The user agent should refuse signing if it does not match the currently active chain.
- `address verifyingContract` - the address of the contract that will verify the signature. The user-agent may do contract-specific phishing prevention.
- `bytes32 salt` - a disambiguating salt for the protocol. This can be used as a domain separator of last resort.

Recommendation

Review the code and do more testing.

Alleviation

[SYNDICATE PROTOCOL]: We will address it as soon as possible.

SPB-01 | Lack of Input Validation

Category	Severity	Location	Status
Control Flow	Informational	projects/Syndicate/contracts/pools/SyndicatePoolBase.sol (9ab9f5c): 200	Resolved

Description

The state variable `minLockTime` has no function to modify. If it is set a wrong number, the only way to change it will be deploying a new contract. So, it is necessary to add basic checks on this variable.

Recommendation

Consider adding validation check `minLockTime`.

Alleviation

[SYNDICATE PROTOCOL]: The variable `minLockTime` has been removed.

SPB-02 | Lack of updates of `user.subYieldRewards`

Category	Severity	Location	Status
Logical Issue	Medium	projects/Syndicate/contracts/pools/SyndicatePoolBase.sol (9ab9f5c): 331~333	Acknowledged

Description

The function `updateStakeLock()` calls the function `_processRewards()` and passes `false` as the variable `_withUpdate`. So, the function `_processRewards()` will accumulate the user's rewards without updating `user.subYieldRewards`. It may let the user can get the rewards repeatedly.

Recommendation

Add the updates of `user.subYieldRewards`.

Alleviation

[SYNDICATE PROTOCOL]: We acknowledged this finding, and we will change the `_withUpdate` to `true`.

SPF-01 | Missing calculate reward

Category	Severity	Location	Status
Logical Issue	Medium	projects/Syndicate/contracts/pools/SyndicatePoolFactory.sol (9ab9f5c): 308	Resolved

Description

The function `changePoolWeight(address, uint32)` is used to change the `weight` of the `pool` but the previous rewards were ignored.

Recommendation

We recommend calculating the previous rewards before modifying the `weight` of the `pool`.

Alleviation

[SYNDICATE PROTOCOL]: Adjust of pool weight is usually done when the pol is just created and thus no need to adjust.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Logical Issue

Logical Issue findings 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.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is 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 assessment. This report does 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.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing 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.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS

AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING

MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.