CERTIK

Code Security Assessment

Pika Crypto

Jan 26th, 2022

Table of Contents

Summary

This report has been prepared for Pika Crypto to discover issues and vulnerabilities in the source code of the Pika Crypto 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	Pika Crypto
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/GameFiLtd/Pika-core/
Commit	8148d4d99a9df390b43fd7635f06aab33dbaad06 d987a33065bc6b912e3f0db2645c00a8ebe09c92 28f68bdfd36762ce1d43b1312a024a00bd5287aa

Audit Summary

Delivery Date	Jan 26, 2022
Audit Methodology	Static Analysis, Manual Review
Key Components	Base, Pika, Staking, Liquidity

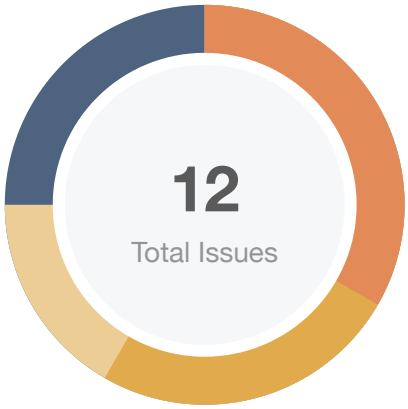
Vulnerability Summary

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

Audit Scope

ID	File	SHA256 Checksum
SCK	projects/pika/contracts/Staking.sol	baa34d00fba9ae807c7a4d0258ae15a81c8f58de20f81cafea6f0296457d5a0d
LCK	projects/pika/contracts/ecosystem/Liquidity.sol	1937444841693 61a47165c19061f3927ca7997f3f6d3bea5061dd6d0aecca5c1
SCP	projects/pika/contracts/ecosystem/Staking.sol	8710deded936184844a7687be33697ba8c7d503d5b5bd783946aa8c8e04020fc
PCK	projects/pika/contracts/tokens/pika/Pika.sol	b82a82d41c0126630e14d708ea183f1af9a1fb27f0ffc297819ea21a08173f58
PPC	projects/pika/contracts/tokens/pika/PikaPolygon.sol	9ba8202067e4bd7d46c02062ef523e24d83343f361bcbf49e19dfca0e99bc0f6
BCK	projects/pika/contracts/tokens/Base.sol	20bd3cf23f4c88ca935b1606ebfa5853783e0c348fae9acaaec43ff1b0573022
BPC	projects/pika/contracts/tokens/BasePolygon.sol	97f33a52a0d1183a189b3c49361fe502286baa1a072f8a04815b313162acb2f4
OIC	projects/pika/contracts/OwnedInitializable.sol	d1101adb59b6bfc048778054dedd88829b21f419dfb09ed0689e142d7ecf3466
LCP	projects/pika/contracts/ecosystem/Liquidity.sol	2b60885cad04fafe46188edc8a33bff4da28f7f42e6be3a353f06d2248026ebd
SKP	projects/pika/contracts/ecosystem/Staking.sol	6a29494dd9ccc395cadae55107d44904c15b597d63f0636494b243dc52cc7a78
PCP	projects/pika/contracts/tokens/pika/Pika.sol	e706e11fab17a7d9dd0c4445423001fd6f012ab57c2efb0fe7b07cf8fae1cd21
PPK	projects/pika/contracts/tokens/pika/PikaPolygon.sol	d36e313f67f093c5134276c0419ae78464dc12b8122f5046b2b2f958ef9feb78
BCP	projects/pika/contracts/tokens/Base.sol	8dc012add7dc537535641f711c0b6a300826ba070134dd986c7e6a5e3c3442ad
BPK	projects/pika/contracts/tokens/BasePolygon.sol	fbd9d97501fdcd7c3e22440f92719408fc446a904bef009f0aaf8855e381eebe
OIK	projects/pika/contracts/OwnedInitializable.sol	3bebdc4a1ec8c40e15e8ee87761a7f2bd1b59daf40d7d35d63d74636fa00a99a

Findings



Critical	**0** (0.00%)
Major	**4** (33.33%)
Medium	**3** (25.00%)
Minor	**2** (16.67%)
Informational	**3** (25.00%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
BCK-01	Centralization Risk	**Centralization / Privilege**	**Major**	Acknowledged
BCK-02	Initial Token Distribution	**Centralization / Privilege**	**Major**	Acknowledged
BCK-03	Should Ensure The Final `amountWithFee` Not 0 Before Transfer Fees	Volatile Code	Medium	Resolved
BCK-04	Did Not Remove Current Address From ExcludeFromFee When Set New Address	Control Flow	Minor	Resolved
BCK-05	Need To Check Fee Number Not Overflow uint96	Mathematical Operations	Informational	Resolved
BCK-06	Potential Sandwich Attack	Logical Issue	Informational	Acknowledged
LCK-01	Centralization Risk	**Centralization / Privilege**	**Major**	Acknowledged
LCK-02	Unchecked Value of ERC-20 `transfer()`/`transferFrom()` Call	Volatile Code	Minor	Resolved
LCK-03	Storage Manipulation In `view` Functions	Gas Optimization	Informational	Resolved
OIC-01	Centralization Risk	**Centralization / Privilege**	**Major**	Acknowledged
SCK-01	Centralization Risk	**Centralization / Privilege**	**Medium**	Acknowledged

ID	Title	Category	Severity	Status
SCP-01	Potential Sandwich Attacks	Logical Issue	Medium	Acknowledged

BCK-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	Major	projects/pika/contracts/tokens/Base.sol (all contracts): 202, 192, 178, 158, 138, 122, 86	Acknowledged

Description

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

- setExcludeFromFee
- setMinSupply
- setBeneficiary
- setStaking
- setLiquidity
- setFeesEnabled
- setSwapEnabled

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

Recommendation

We advise the client to carefully manage the `owner` account's private key 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

[Pika Team]: _owner accounts will be managed by a multisig wallet which is controlled by multiple hardware wallets. Even if someone could take control of the ownership of the contract, all they could do is take snapshots, which doesn't pose any risk.

BCK-02 | Initial Token Distribution

Category	Severity	Location	Status
Centralization / Privilege	Major	projects/pika/contracts/tokens/Base.sol (all contracts): 51	Acknowledged

Description

All of the tokens inherited from Base Token are sent to the contract deployer when deploying the contract. This could be a centralization risk as the deployer can distribute 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

[Pika Team]: Token is already live and distributed.

BCK-03 | Should Ensure The Final `amountWithFee` Not 0 Before Transfer Fees

Category	Severity	Location	Status
Volatile Code	Medium	projects/pika/contracts/tokens/Base.sol (all contracts): 230~249	Resolved

Description

The Contract should precheck the amount before any fee transfer to avoid the loss of the user's asset. But Within the transfer function, `amountWithFee` is calculated after fee transfer, and launch the ERC20 transfer without ensuring the final `amountWithFee` > 0. The transfer would not make any token transfer to recipient but paid the all fees.

```
230         if (transferFee > 0) {
231             transferFee = handleFeeTransfer(sender, amount, beneficiaryAddress,
transferFee);
232         }
233         uint256 amountWithFee = amount - transferFee;
234         // burn tokens if min supply not reached yet
235         uint256 burnedFee = _calculateFee(amount, 25);
236         if (totalSupply() - burnedFee >= minSupply) {
237             _burn(sender, burnedFee);
238             amountWithFee -= burnedFee;
239         }
240         if (stakingFee > 0) {
241             stakingFee = _calculateFee(amount, stakingFee);
242             ERC20Upgradeable._transfer(sender, stakingContract, stakingFee);
243             amountWithFee -= stakingFee;
244         }
245         if (liquidityFee > 0) {
246             liquidityFee = _calculateFee(amount, liquidityFee);
247             ERC20Upgradeable._transfer(sender, liquidityContract, liquidityFee);
248             amountWithFee -= liquidityFee;
249         }
```

Recommendation

It's recommended to check the final amountWithFee before any fee transfer.

Alleviation

[Pika Team]: Fixed in commit d6730b0.

BCK-04 | Did Not Remove Current Address From ExcludeFromFee When Set New Address

Category	Severity	Location	Status
Control Flow	Minor	projects/pika/contracts/tokens/Base.sol (all contracts): 180, 160, 140	Resolved

Description

Within Functions:

- setBeneficiary
- setStaking
- setLiquidity

We could know the address is put into excludeFromFee when it's set. But as the new address is set, the origin address is still in excludeFromFee. It's needed to remove the currentAddress from ExcludeFee.

```
setExcludeFromFee(_contractAddress, true);
...
setExcludeFromFee(currentAddress, false);
```

Recommendation

It's recommended to remove current address from excludeFromFee when new address is set excludeFromFee.

Alleviation

[Pika Team]: Fixed in commit 0277afe.

BCK-05 | Need To Check Fee Number Not Overflow Uint96

Category	Severity	Location	Status
Mathematical Operations	Informational	projects/pika/contracts/tokens/Base.sol (all contracts): 96	Resolved

Description

As the linked code, uint256 variable `fee` need to shift to the higher 96 bits and combine with an uint160 to uin256.

```
95          uint256 storedBeneficiary = uint256(uint160(_beneficiary));
96          storedBeneficiary |= _fee << 160;
```

But it did not ensure `fee` is not larger than the max uint96.

Recommendation

It is recommended to ensure the fee(uint256) would not overflow from uint96.

Alleviation

[Pika Team]: Fixed in commit 1a63cc4: When storing the fee it's ensured it does not exceed 10000.

BCK-06 | Potential Sandwich Attack

Category	Severity	Location	Status
Logical Issue	Informational	projects/pika/contracts/tokens/Base.sol (all contracts): 278	Acknowledged

Description

Potential sandwich attacks could happen if calling `uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens` and `uniswapV2Router.addLiquidityETH` without setting restrictions on slippage.

For example, when we want to make a transaction of swapping 100 AToken for 1 ETH, an attacker could raise the price of ETH by adding AToken into the pool before the transaction so we might only get 0.1 ETH. After the transaction, the attacker would be able to withdraw more than he deposited because the total value of the pool increases by 0.9 ETH.

Recommendation

We recommend using Oracle to get an estimation of prices and setting minimum amounts based on the prices when calling the aforementioned functions.

Alleviation

[Pika Team]: The potential attack vector is acknowledged, but because the tokens are swapped automatically on every sell, the swapped fees are too low to make such an attack feasible.

LCK-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	Major	projects/pika/contracts/ecosystem/Liquidity.sol (all contracts) : 92, 84	Acknowledged

Description

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

- setLockPeriod
- setVestingPeriod

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

Recommendation

We advise the client to carefully manage the `owner` account's private key 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

[Pika Team]: _owner accounts will be managed by a multisig wallet which is controlled by multiple hardware wallets. Even if someone could take control of the ownership of the contract, all they could do is take snapshots, which doesn't pose any risk.

LCK-02 | Unchecked Value Of ERC-20 `transfer()` / `transferFrom()` Call

Category	Severity	Location	Status
Volatile Code	Minor	projects/pika/contracts/ecosystem/Liquidity.sol (all contracts): 77, 118	Resolved

Description

The linked `transfer()`/`transferFrom()` invocations do not check the return value of the function call which should yield a `true` result in case of proper ERC-20 implementation.

Recommendation

As many tokens do not follow the ERC-20 standard faithfully, they may not return a `bool` variable in this function's execution meaning that simply expecting it can cause incompatibility with these types of tokens. Instead, we advise that OpenZeppelin's `SafeERC20.sol` implementation is utilized for interacting with the `transfer()` and `transferFrom()` functions of ERC-20 tokens. The OZ implementation optionally checks for a return value rendering compatible with all ERC-20 token implementations.

Alleviation

[Pika Team]: Fixed in commit 9f4b99c. Although not necessary because the UniSwap V2 pair ERC20 tokens revert when a transfer fails.

LCK-03 | Storage Manipulation In `view` Functions

Category	Severity	Location	Status
Gas Optimization	Informational	projects/pika/contracts/ecosystem/Liquidity.sol (all contracts): 107	Resolved

Description

There should not be any storage variable manipulation in the `view` function.

Recommendation

We advise the client to consider changing `storage` into `memory`.

Alleviation

[Pika Team]: Fixed in commit 98fd174.

OIC-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	Major	projects/pika/contracts/OwnedInitializable.sol (all contracts): 33	Acknowledged

Description

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

- proposeOwner

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

Recommendation

We advise the client to carefully manage the `owner` account's private key 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

[Pika Team]: _owner accounts will be managed by a multisig wallet which is controlled by multiple hardware wallets. Even if someone could take control of the ownership of the contract, all they could do is take snapshots, which doesn't pose any risk.

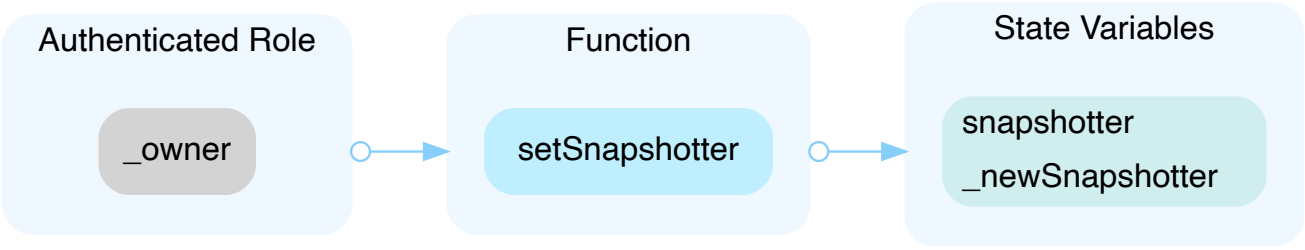
SCK-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	Medium	projects/pikacrypto/contracts/Staking.sol (f2cbe78): 57~60	Acknowledged

Description

In the contract, `Staking`, the role, `_owner`, has the authority over the functions shown in the diagram below.

Any compromise to the privileged account which has access to `_owner` may allow the hacker to take advantage of this and take snapshot of the token.



Recommendation

We advise the client to carefully manage the privileged account's private key 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

[Pika Team]: _owner accounts will be managed by a multisig wallet which is controlled by multiple hardware wallets. Even if someone could take control of the ownership of the contract, all they could do is take snapshots, which doesn't pose any risk.

SCP-01 | Potential Sandwich Attacks

Category	Severity	Location	Status
Logical Issue	Medium	projects/pika/contracts/ecosystem/Staking.sol (all contracts): 89~102, 48~56	Acknowledged

Description

The reward calculation in `staking.sol` is not related to how long has a user staked. It only depends on `magnifiedRewardPerShare`, which is updated whenever the contract receives ETH. Therefore, it is possible for an attacker to perform the following action:

- When he observes a tx that sends ETH to the contract, he makes a huge deposit(Potentially via flashloan) with high gas fee, so that his deposit will be processed before the tx.
- He immediately calls withdraw, to gain the majority of the ETH as profit

Recommendation

We recommend making the staking reward calculation related to how long has a user staked.

Alleviation

[Pika Team]: I deem this attack vector as a non-issue. The rewards in ETH paid out to the contract will be very tiny, because the autoswap feature just takes a fraction of a users transaction, autoconverts it to ETH and sends it to the staking contract. Executing such an attack would cost way more than it would benefit. Additionally, a lot of tokens were paid into the contract (with our previous staking contract held over 25% of the total supply if I remember correctly). Even if emptying out the entire uniswap pool with a huge buy, the attacker would only gain less than 17% of the staking pool, further minimizing potential profits. Lastly, I don't see the possibility of a flashloan attack here, as the user would buy a lot of tokens for example to deposit into the staking pool, but the deposit of the large amount of ETH would only happen in the next transaction, leaving the attacker with no funds to pay back the loan.

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.

Gas Optimization

Gas Optimization findings 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 findings relate to mishandling of math formulas, such as overflows, incorrect operations etc.

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.