Security Assessment

Matic

Apr 19th, 2021

Table of Contents

Summary

This report has been prepared for Matic to discover issues and vulnerabilities in the source code of the Matic 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	Matic
Platform	Custom
Language	Solidity
Codebase	https://github.com/maticnetwork/pos-portal
Commit	c810a2400e54f610149435447192 46f0c8b66401 f33407cbecfd0dbbd3da2da849efbc60e6018c7d 8dcc8b4b6476bfdfaa10eff76e1eeed178f252ee

Audit Summary

Delivery Date	Apr 19, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	1	0	0	0	0	1
Major	1	0	0	0	0	1
Medium	0	0	0	0	0	0
Minor	2	0	0	0	0	2
Informational	11	0	0	1	0	10
Discussion	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
CCC	child/ChildChainManager/ChildChainManager.sol	60ed3912290785a263dc28ad45e57419cf86a870f2a4de9b4931d9d533a7e709
CCP	child/ChildChainManager/ChildChainManagerProxy.sol	c4e0df92d8677e2336a37123b2f555b257a141c27f80185f2f3a0b015d4a5dae
ICC	child/ChildChainManager/IChildChainManager.sol	30409cdbef9546aa2b5349957ea570ad828c49379567ac8339619c4e9f452859
CER	child/ChildToken/ChildERC1155.sol	99b3f6b565f84ff18a7d57e74e289227bb7ba4c0cc6bc8c20ebb9e89e6179643
CEC	child/ChildToken/ChildERC20.sol	ce1009850314412559e21cce9765808839cd0b5f1557cb5ca4de036cad54ad47
CET	child/ChildToken/ChildERC721.sol	8ab0435d4088680a453cbb65cb37fb8f78b3cd574a0b6ca97cc17be09730f085
CME	child/ChildToken/ChildMintableERC721.sol	3f7b1545bd0cf9ec5fbcd347dd77029472e15e4e5a3d4bec8b9d5169100a2275
ICT	child/ChildToken/IChildToken.sol	36b7ecdaeeebcf6355ac819f45b2f029ae1e98c59d8bf9b33bb863cc4d7a935b
MWE	child/ChildToken/MaticWETH.sol	da453c7258a9927a0cd857b9a4719e7c241be09d5d191603417ce48623071f4d
IER	common/Proxy/IERCProxy.sol	88bcaa7b8811e3eb498ca32f484c380625df7a4fea1ee78bed148f29bec39ae2
PPK	common/Proxy/Proxy.sol	4e4fdda4ebbaf36bb5677ef53080d045d6741aaa72425176a8530c2851a4b80c
UPP	common/Proxy/UpgradableProxy.sol	57e430ba75d27d315b515d91da8b13638b5cc8dfd75b9b7c2614e4f3ec4f4464
ACM	common/AccessControlMixin.sol	915ef86b166c84378ff8edea739514a263f18a9df705b526946bfa630c3b4aef
CMK	common/ContextMixin.sol	ee8a1937321eb000beb8298d1a61a286b9e42988e7981be0a6f3abfdf80ed5d5
EIP	common/EIP712Base.sol	93a57f09c7404a935d35bcd3467ccd73a217126ded95185d0e3c5a44ae764fb5
INI	common/Initializable.sol	18524e02077162b7f743c17564d6dbaf35ec7aa57978013c06a92aa6bc171784

ID	File	SHA256 Checksum
NAK	common/NetworkAgnostic.sol	28afa94db74c36dd7742f3f29c0d1425ecfb768832daabf946d229bbbb5ae68c
MER	lib/Merkle.sol	e31f4df5a8972b262ad316033b6ce92d41fc37e51d6e794537a49bb3fe612dfe
MPP	lib/MerklePatriciaProof.sol	8553cedb0f8d29719cf7ad8d45803f641a09d682509759d6a2717b768ef7f3a1
RLP	lib/RLPReader.sol	ab90c8c689f835fec966929ccc9b6ed334689806f274458e14d4fce8f2a52772
IRC	root/RootChainManager/IRootChainManager.sol	977d5d57e218e3bd3a4059f0ed70ec74607fc1ba9b14437b3cbd1bf7333a949a
RCR	root/RootChainManager/RootChainManager.sol	1515327ade5149df4f5deefdf3e01afdbfc85e7a1415d917315a8f163fcc8b63
RCP	root/RootChainManager/RootChainManagerProxy.sol	0233005f7e6478ee5e39acf69b9bb4723a57d09f1bb475f8ce4a8fcc039536f9
DER	root/RootToken/DummyERC1155.sol	113900e387ab4c25704831837beb201490bd444025c1e09e8c8e40aeaa7db940
DEC	root/RootToken/DummyERC20.sol	82d4790e43200fb02860a29bc7d69ae83a964ddca5cddc1144f1cebc62786c48
DET	root/RootToken/DummyERC721.sol	f221571024dc25fe457de63383b642badfca7aef64e155ecc92c1b3d8e011b2f
DME	root/RootToken/DummyMintableERC721.sol	72eaf2cdfc0c25bd5de6c69f6b8e7a84e405ed5b77168f54297b126111914e41
IME	root/RootToken/IMintableERC721.sol	a3677d38d4f9be513eec4089f3eb7082ef864cc9abfae430f0e0c3a596b0f764
DSS	root/StateSender/DummyStateSender.sol	1bc1c8c11ba8f043ef8e5682bcee06bc566858dba7f07c61fffda47e3671ddef
ISS	root/StateSender/IStateSender.sol	ce136f7f9ff011e67794317c18096393302ef16a5f221f5d62efd0bbaba2c606
ERC	root/TokenPredicates/ERC1155Predicate.sol	85adbaae773dbd3b7229a58f335d333f16594e8fc435c98dc4f621e8f4bb5bb6
ERP	root/TokenPredicates/ERC1155PredicateProxy.sol	9f440e7116539ac4e9723154d550409cbd21066a3ba129c14d525cf88421dca2
ERT	root/TokenPredicates/ERC20Predicate.sol	780e8bb27f4b1d8efc1eac69a3fbf0017f6dfb1ee8f4d4bb2898e3f0fd1014d9

ID	File	SHA256 Checksum
ECP	root/TokenPredicates/ERC20PredicateProxy.sol	13c8cc863836023ff9ed72bd9d30a01b8c7c16a05b2ece88c3f29aeb29bd7d74
ECT	root/TokenPredicates/ERC721Predicate.sol	2ebe572dba76fa66c0f2536c59fbdd3e8f0ac8bed8d674cb4bb0e4c1eaca7f30
EPP	root/TokenPredicates/ERC721PredicateProxy.sol	1a3b7a5babd62eec54c7d4bda08bdd6d0bfaf954be37b0ac48540a057949916a
EPT	root/TokenPredicates/EtherPredicate.sol	bd97201f0501576af5a72450d30820d689b648462abbd227b854f6600d2990d4
ETP	root/TokenPredicates/EtherPredicateProxy.sol	20e7ea6b3cc6ec89d26c9f3ebfa752621ea6681aa9ec09e7d72d7305eadf276a
ITP	root/TokenPredicates/ITokenPredicate.sol	7eb79792b32a38d53fdf925b42620141035dfecf40e284611495eea723a0bc3f
MEC	root/TokenPredicates/MintableERC721Predicate.sol	b07d253f64800a789cddcd14eed4f64caf5f72d122b1fb9943e8f94b14827ce5
MEP	root/TokenPredicates/MintableERC721PredicateProxy.sol	c9f49b5eed84499a419ea9aeb4487a4c9d36254031f24c0f2f7fbfaa16859063
ICM	root/ICheckpointManager.sol	9875c57e1c54e16c15793a169fcaea0f0dc0ca5b470a5b74593ee1cb305d5f89
MCM	root/MockCheckpointManager.sol	f616c2f2f841370ff6e74ce5b7e0360a02ddde3aeae7aa6ca58e7798589f9782
PTI	test/ProxyTestImpl.sol	94039e0a449ca776dae242a63832705803bd6b6c54c77a8e3c0bc524f092937d
PTS	test/ProxyTestImplStorageLayoutChange.sol	6b2a76401df983526fd5354df75a3111df4b24f9ff0f0aba999f03e095655543
CCK	ChainConstants.sol.template	351e5f849ea95ca2291fe1e3a75f3efe8d5d2d6b895cfda7d72c49815319cf62
MIG	Migrations.sol	6ac6666c6e0e1be693d2a92443a77ec593fefbf508c2365a5670b6f5f443dcce

Review Notes

Overview

A primary focus for the audit is to have a thorough look at the smart contracts that power the PoS (proof-of-stake) based bridge mechanism for Matic Network. Specifically we want to make sure that the exit mechanism is correctly implemented and cannot be exploited to withdraw the tokens deposited on the sidechain more than once per transfer.

Scope of Work

- The audit work was scoped to a specific commit c810a2400e54f61014943544719246f0c8b66401 of the source code per the agreement
- The codebase are divided into modules of smart contracts based on their functionalities:

Lib

Contains implementation of the RLP encoding, Merkle tree proof verification, Merkle Patricia Tree proof verification.

Common

Contains standard solidity libraries for smart contract upgradability and access control.

Audit Summary

The codebase of the project was identified to be carefully designed and detailed, as well as properly documented. In total we found one critical issue in the exit mechanism (Exhibit 1) that enables malicious attackers repeated withdrawal from the childchain. All other issues were of negligible importance and mostly referred to coding standards and inefficiencies.

In the second round we have found one major issue that enables replay attacks, one between the child chain and main chain, second between different contracts on the child chain.

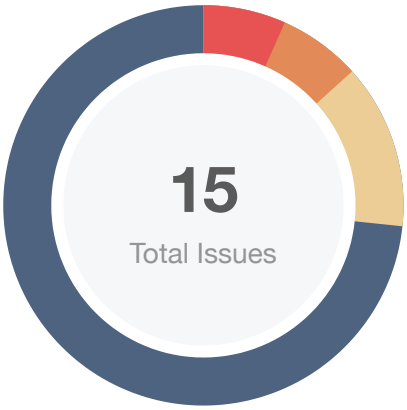
Audit Revisions

On 11th August 2020 the pull request preliminary-audit-fixes with commit f33407cbecfd0dbbd3da2da849efbc60e6018c7d was submitted. This pull request fixed almost all listed issues except 5, 11, 15. The changes were approved by the Certik audit team on the same day.

On 19th August 2020 the pull request feature/meta-transaction with commit 8dcc8b4b6476bfdfaa10eff76e1eeed178f252ee was submitted. The pull request has fixed all the remaining issues. The changes were approved by the Certik audit team on the same day.

On 20th August 2020 the Matic team discovered some issues in the library contracts through extra testing. The Certik audit team has verified and approved the fixes in commit 165acf14f70ff272883600ce1a233c129c584398.

Findings



Critical	1 (6.67%)
Major	1 (6.67%)
Medium	0 (0.00%)
Minor	2 (13.33%)
Informational	11 (73.33%)
Discussion	0 (0.00%)

ID	Title	Category	Severity	Status
GLOBAL-01	Unlocked compiler version	Compiler version	Informational	Resolved
GLOBAL-02	Depositor role	Function logics	Informational	Resolved
ICT-01	Interface structure	Coding Style	Informational	Resolved
MPP-01	Repeated exit	Security	Critical	Resolved
MPP-02	Redundant return statement	Code optimization	Informational	Resolved
NAK-01	Replay attack between child chain and root chain	Implementation	Major	Resolved
RCR-01	Integer underflow	Mathematical Operations	Informational	Resolved
RCR-02	Multiple lookups from storage	Gas Optimization	Informational	Resolved
RLP-01	`require` error message	Coding Style	Informational	Resolved
RLP-02	`isList()` verification	Logics	Informational	Resolved
RLP-03	Unnecessary if clause	Logics	Informational	Resolved

ID	Title	Category	Severity	Status
RLP-04	Inefficient comparison	Logics	Informational	Acknowledged
RLP-05	Exclusion of empty bytes input	Implementation	Informational	Resolved
RLP-06	Non unique uint encoding	Implementation	Minor	Resolved
RLP-07	Incorrect encoding of value “false”	Implementation	Minor	Resolved

GLOBAL-01 | Unlocked compiler version

Category	Severity	Location	Status
Compiler version	Informational	Global	Resolved

Description

An unlocked compiler version in the source code of the contract permits the user to compile it at or above a particular version. This, in turn, leads to differences in the generated bytecode between compilations due to differing compiler version numbers. This can lead to an ambiguity when debugging as compiler specific bugs may occur in the codebase that would be hard to identify over a span of multiple compiler versions rather than a specific one.

Recommendation

We advise that the compiler version is instead locked at a specific version possible that the full project can be compiled at.

Alleviation

Compiler version has been locked in commit f33407c.

GLOBAL-02 | Depositor role

Category	Severity	Location	Status
Function logics	Informational	Global	Resolved

Description

In the constructors of `ChildToken` contracts the depositor role is assigned to the contract's creator. The depositor is an important role as it is the only address that is authorized to deposit the tokens from root contract to child contract. From the natspec it is evident that only the `ChildChainManager` contract should possess this role and this contract does not deploy the `ChildToken` contracts so one would needs some transactions to pass over the depositor rights.

Recommendation

Assign the depositor role directly to the `ChildChainManager` in the constructor.

Alleviation

In commit f33407c the depositor role has been initiated to `childChainManager`.

ICT-01 | Interface structure

Category	Severity	Location	Status
Coding Style	Informational	child/ChildToken/IChildToken.sol: 3	Resolved

Description

The IChildToken interface contains only the deposit function and is inherited in every Child Token contract. Every Child Token contains an additional withdraw function which is essential because it enables token withdrawal from the child chain to the main chain so we believe this function belongs to the general pattern as well, i.e. IChildToken interface.

Recommendation

Add withdraw to the IChildToken interface.

Alleviation

All child tokens are not expected to have the same interface for withdraw. For eg. ERC20 tokens will have single param for amount and ERC1155 will have 2 params for amount and tokenId. Due to this reason, not adding withdraw function to IChildToken interface.

MPP-01 | Repeated exit

Category	Severity	Location	Status
Security	Critical	lib/MerklePatriciaProof.sol: 150~153	Resolved

Description

RootChainManager.sol lines 250-255 the exitHash is determined by three factors, one of which is inputDataRLPList[8], the branch mask, i.e. the path in the receipt merkle patricia trie from the root to the corresponding burning transaction receipt.

The path is in bytes and translated to hex by _getNibbleArray() in MerklePatriciaProof.sol. There are 2 cases depending whether the path in hex has odd or even length. In case on line 150 we accept every beginning nibble except 1,3 and erase the first two nibble in the hex array, which means two arrays 2055 and 4055 would produce the same _getNibbleArray() to bypass `verify`, but they produce different exitHash so one can exit more than once.

Recommendation

One needs to check in the else case that the first two nibbles are 20 (we don't allow paths ending in extension nodes here).

Alleviation

In commit f33407c the computation of exitHash the component inputDataRLPList[8] is changed to MerklePatriciaProof._getNibbleArray(inputDataRLPList[8].toBytes()), which makes the branchMask unique.

MPP-02 | Redundant return statement

Category	Severity	Location	Status
Code optimization	Informational	lib/MerklePatriciaProof.sol: 102, 106	Resolved

Description

The default return value is false so the return statements on lines 107 and 102 are not needed and the else case on line 101 can be omitted.

Recommendation

Omit the unnecessary code.

Alleviation

The recommendation has been assimilated in commit f33407c.

NAK-01 | Replay attack between child chain and root chain

Category	Severity	Location	Status
Implementation	Major	common/NetworkAgnostic.sol: 37~87	Resolved

Description

Matic wants to enable users to send transactions to matic network without changing the network in their wallet so the chainId is the same for both child chain and main chain. This opens an attack vector when a user wants to send a transaction only to the child chain, then the attacker can submit the signed transaction to the other chain against that user's will. The requires the same nonce, which can happen, and the same contract address on both chains to have an exploitable function, for example when the two contracts are deployed from the same address.

Recommendation

Change the chainID and check the chainID before transaction execution.

Alleviation

Using same chainId for child token contracts is exploitable. Changed NetworkAgnostic feature to native meta transaction in commit 8dcc8b4. The opcode chainid is used so it is always the native chain id. This chain id is used as salt in EIP712 domain separator so that metamask allows signing tx with different chain id than currently selected network.

RCR-01 | Integer underflow

Category	Severity	Location	Status
Mathematical Operations	Informational	root/RootChainManager/RootChainManager.sol: 342	Resolved

Description

The subtraction `blockNumber - startBlock` is unsafe and can cause integer underflow, but we believe this cannot be exploited in any way as it would certainly cause the Merkle proof verification to fail.

Recommendation

Use SafeMath for arithmetic operations.

Alleviation

SafeMath usage has been added in commit f33407c.

RCR-02 | Multiple lookups from storage

Category	Severity	Location	Status
Gas Optimization	Informational	root/RootChainManager/RootChainManager.sol: 275, 281	Resolved

Description

The value in storage `childToRootToken[childToken]` is looked up twice in exit function. Each lookup costs 200 gas each. It would be better to assign this value to a memory variable, because both memory assignment and look up cost 3 gas each.

Recommendation

Use memory assignment to avoid multiple storage lookups.

Alleviation

The recommendation has been assimilated in commit f33407c.

RLP-01 | `require` error message

Category	Severity	Location	Status
Coding Style	Informational	lib/RLPReader.sol: 104, 168, 208	Resolved

Description

`require` can be used to check for conditions and throw an exception if the condition is not met, in which case the error message provided by the developer will appear. This is why a very descriptive error message is needed.

Recommendation

Adding an error message describing the failed condition.

Alleviation

In commit f33407c error messages have been added.

RLP-02 | `isList()` verification

Category	Severity	Location	Status
Logics	Informational	lib/RLPReader.sol: 166~191	Resolved

Description

The function `numItems()` calculates the number of elements in an `RLPItem` representing a list, so it should check whether the input indeed represents a list. For example it would return the length of a string. On the other hand adding a `require` check is not necessary and would only cost more gas, since the only place this function is used is in `toList()` and the aforementioned condition is already checked. The exhibit is here just for completeness in case the `RLPReader.sol` library is expanded and the function `numItems()` is used in additional functions.

Recommendation

Add `require` check if necessary.

Alleviation

In commit f33407c the check and issue description have been added to the comments of `numItems()` function

RLP-03 | Unnecessary if clause

Category	Severity	Location	Status
Logics	Informational	lib/RLPReader.sol: 229	Resolved

Description

Checking in the function `numItems()` whether `item.len == 0` to return zero is not needed since if `item.len == 0` then it does not encode a list, because the RLP encoding of an empty list is `0xC0`. Moreover the function `numItems()` is only used in `toList()` and right before calling `numItems()` the condition `isList()` is checked which excludes the possibility of an empty item.

Recommendation

Omit the if clause.

Alleviation

In commit f33407c the recommendation has been assimilated.

RLP-04 | Inefficient comparison

Category	Severity	Location	Status
Logics	Informational	lib/RLPReader.sol: 234	Acknowledged

Description

In function `numItems()` by the definition of `RLPItem` it is clear the the value of `currPtr` can never exceed `endPtr` and the while loop on line 234 would stop when `currPtr` and `endPtr` are equal, hence instead of `currPtr < endPtr` we can use `CurrPtr != endPtr` as each not equal comparison costs 3 less gas (negligible for short `RLPItem`, this exhibit is just here for completeness) than less than comparison

Recommendation

Use not equal instead of less than.

Alleviation

The improvement is negligible so the Exhibit was not applied.

RLP-05 | Exclusion of empty bytes input

Category	Severity	Location	Status
Implementation	Informational	lib/RLPReader.sol: 53~64	Resolved

Description

In several functions of `RLPReader` library we check that `item.len` is not zero. Indeed any RLP encoding (even of empty string or empty array) is non-empty, this means we should check this condition in the function `toRlpItem()` to exclude this case before hand.

Recommendation

Check that the input bytes are not empty in the function `toRlpItem()`.

Alleviation

In commit f33407c the length check has been added.

RLP-06 | Non unique uint encoding

Category	Severity	Location	Status
Implementation	Minor	lib/RLPReader.sol: 173~205	Resolved

Description

The functions `toUint()` and `toUintStrict()` only take `RLPItem` representing uint as input. To get the data bytes only the length of the payload offset is calculated, whereas the content of this prefix is not considered, which means invalid `RLPItem` with prefixes of incorrect length or short list would still get through.

Recommendation

Check the payload offset content in toUint() and toUintStrict().

Alleviation

In commit f33407c the prefix check has been added.

RLP-07 | Incorrect encoding of value “false”

Category	Severity	Location	Status
Implementation	Minor	lib/RLPReader.sol: 155~164	Resolved

Description

According to RLP encoding specification the special value “false” is encoded as “0x80”, whereas in current implementation of the function “toBoolean” it is “0x00”.

Recommendation

Change the implementation to follow the specification.

Alleviation

toBoolean() is not being used anywhere in the system, the function is removed completely.

Appendix

Finding Categories

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.

Coding Style

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

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.