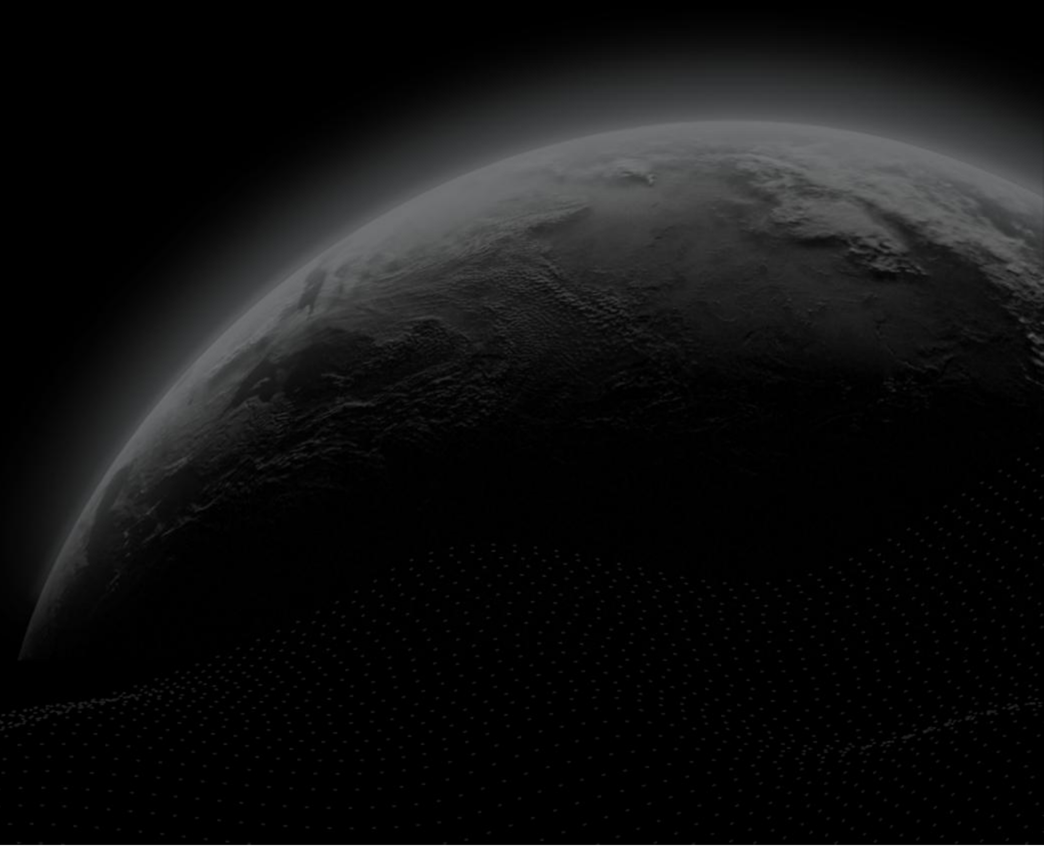
Security Assessment

Musk Identity

Certificate Assessment Date: October 8, 2025



Certificate Assessment Date: August 28, 2025

Musk Identity

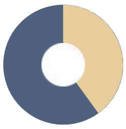
The security assessment was prepared by CertiK, the leader in Web3.0 security.

Executive Summary

TYPES	ECOSYSTEM	METHODS
Smart ontract	Musk Identity	Manual Review, Static Analysis

LANGUAGE	TIMELINE	KEY COMPONENTS
Swift	Delivered on 8/28/2025	N/A

Vulnerability Summary



5	3	0	2	0	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined

0	Critical		Critical risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
0	Major		Major risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
0	Medium		Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
2	Minor	2 Partially Resolved	Minor risks can be any of the above, but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.
3	Informational	3 Resolved	Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

TABLE OF CONTENTS | Musk Identity

AUDIT SCOPE | Musk Identity

0 files audited

ID	Repo	File	SHA256 Checksum

APPROACH & METHODS | Musk Identity

Introduction

This report has been prepared for Musk Identity to identify issues and vulnerabilities in the supplied source code snippets of the front- end, mobile (iOS/Android), and SDK components. The code snippets are supplied to us in individual zipped files, each having the following `sha1` hash:

```
0b02b2b5c25d7b00f4d52a4c90fa89f9ed736f37  FE.zip

1188830aa5af47af8c148dccc755cc5571b288c9  Mobile.zip

094d5a1d0d39cdd287a4a874d41706978bf321e0  sdk.zip
```

The primary goal of this engagement was to scrutinise the source code to gauge the application's resilience against various logic issues and vulnerabilities targeting its controls and functionalities. This process was essential for identifying vulnerabilities and weaknesses within the codebase, thereby enabling the provision of tailored recommendations to enhance and fortify its security posture.

This security assessment is an extension of the previous Musk Identity's Blackbox penetration test performed on the Musk Identity wallet. The findings of the prior assessment are documented separately and are not included in this report.

Scope of the Auditing

During this audit review process, CertiK security team has reviewed the following components.

A portion of Mobile App Source Code: The mobile source code contains both the iOS and Android components responsible for wallet creation/import, password management, and data backup to cloud platforms.

A Front End Module: The front-end source code contains the ReactJS UI components for the wallet functionalities, as well as the JS controllers used for interacting with the keyring.

Multiple Modules of Wallet SDK: The SDK contains the following components:

- **Bitcoin SDK:** written in Golang, it is used to interact with the Bitcoin Mainnet or Testnet.
- **okwallet-core:** mnemonic generator and wallet management.
- **src:** Javascript/Typescript component containing BIP32 and BIP39 support and various mathematical modules, ranging from elliptic curve calculation to hashing algorithms.

Summary of Findings

Overall we found the modules being reviewed are secure against the risks being considered in this audit process. The exact risk being reviewed and our evaluation results are presented below in separate sections.

Through this static review process, we found 5 security issues. Three of them are low risk and informational findings, and two of them are with undetermined risk levels. They are likely low risk, but we can not make a final determination due to the scope limit.

Limitations

The following limitations were encountered during the audit, which significantly restricted our ability to perform a comprehensive and thorough security assessment.

The constraints outlined below limit our ability to holistically evaluate the codebase and assess the overall security of the applications and the SDKs:

- Only certain parts of the codebase were shared, it is NOT possible to establish how/where certain functions are used and if memory cleanup is done correctly.
- Given that only partial code was shared, we cannot perform dynamic analysis. Dynamic analysis would clarify if there are any signs of PII leakage, either through application memory or filesystem.
- For the third party dependencies used, the versions could not be established. Some of the libraries used have security vulnerabilities in older versions and it is not possible to determine if the client is at risk or not.
- Mobile app configurations were not shared. (AndroidManifest.xml / Info.plist)
- The shared codebase makes extensive use of internal dependencies which could not be audited. This makes it impossible to determine if the client is exposed to dependency confusion or supply chain attacks.

Mobile Application Audit Summary

Two mobile applications, an iOS app written in Swift + Objective-C and an Android app written in Kotlin, were provided by the client. The application code mostly concerned wallet operations, the entire source code was not provided but sensitive operations regarding wallets and backups could be analyzed.

Threat Vectors Being Reviewed

Our testing was focused on the following attack vectors:

- **Dependency Confusion / Supply Chain attacks**
    - Mobile apps mostly use Musk Identity's internal libraries, most of which were not
    - provided. We check whether the application depends on any known risky third-party packages.

- **Insecure Data Storage**
  - The application allows users to backup their wallets on Huawei Drive, Google Drive or iCloud. We check whether the backup process is implemented following good coding practices.
- **Insecure Cryptographic Primitives**
  - Check whether the cryptographic primitives in use adhere to the standards.
- **Memory Manipulation**
  - Check whether the applications adopt good memory operation practice.
- **Hard-coded Secrets**
  - We check whether the applications contain or make use of hard-coded secrets.

Summary Review Results

Below is our summary of the code security status

- **Dependency Confusion / Supply Chain attacks**
  - Mobile apps mostly use Musk Identity's internal libraries, most of which were not provided. The other third-party dependencies that are used by the applications are either static or primitives of the respective platform which are unlikely to contain malicious code.
- **Insecure Data Storage**
  - The sensitive information corresponding to each wallet is encrypted throughout the lifecycle of the application. The password is encrypted with an AES key that is generated at runtime. The sensitive information is also not logged or stored unencrypted on the filesystem.
  - We confirm these backups are encrypted with a symmetric key that is derived from a user-supplied password. A password is required to enable backups.
- **Insecure Cryptographic Primitives**
  - We confirm that the cryptographic primitives in use adhere to the standards. The apps make use of

strong symmetric and asymmetric encryption algorithms.

- **Memory Manipulation**
  - The applications generally manage memory well. An information recommendation is provided in the finding section.
  - No UAF risks were identified in the client's source code.

- **Hard-coded Secrets**
  - The applications do not contain or make use of hard-coded secrets.

SDK Audit Summary

The following components were provided by the client for the presented engagement:

- **Bitcoin SDK:** written in Golang, used to interact with the Bitcoin Mainnet or Testnet.
- **okwallet-core:** written in Golang, used for mnemonic generation and wallet management.
- **src:** Javascript/Typescript component containing BIP32 and BIP39 support and various mathematical modules, ranging from elliptic curve calculation to hashing algorithms.

Threat Vectors Being Reviewed

Our testing was focused on the following attack vectors:

- **Dependency Confusion / Supply Chain attacks**
  - The SDK uses multiple sources for the module importing within both the JS and Golang components of the SDK. The sources range from local files to libraries located in Musk Identity's internal repositories, as well as third-party modules.
  - We check whether the application depends on any known risky third-party packages.

- **Data Storage**
  - We check whether the data access to the storage is limited and secure.

- **Cryptographic Primitives**

- Check whether the cryptographic primitives in use adhere to the standards.

- **Hard-coded Secrets**
  - We check whether the applications contain or make use of hard-coded secrets.

- **Data Handling**
  - We check whether the implementation adopt good security practices on handling data processing.

Summary Review Results

Our testing was focused on the following attack vectors:

- **Dependency Confusion / Supply Chain attacks**
  - We found some component use modules that are no longer under active development. Please find detailed information in the finding section.
  - The version of `crypto-js` used in the client's SDK could not be established based on the shared source code.
  - he internal repositories used within the imports are locatedat `okinc.com(64.98.135.50)` and `gitlab.okg.com (10.254.3.52)`.
  - Making use of internal dependencies is recommended but it is not possible to establish if the referenced internal dependencies could be vulnerable to supply chain attacks, as they were not shared in the scope of the audit. It's strongly recommended to review the configuration and contents of the third-party modules imported within the applications.

- **Data Storage**
  - The okwallet-core library is used for mnemonic generation and wallet management. The data is transmitted within the function calls, mostly through the `storage` object.

- **Cryptographic Primitives**
  - The cryptographic primitives in use adhere to the standards. The apps make use of strong symmetric and asymmetric encryption algorithms. Hard-coded Secrets
  - We found a hard-coded secret in the code, however, no function calls towards these functions have been observed within the provided codebase. Details are provided in the finding section.

- **Data Handling**
  - We identified a snippet of code that could allow an integer overflow if a sufficiently large integer is cast as a uint32. This function is actively used in the codebase but it is not possible to determine if there is a real risk of an integer overflow in the client's codebase, as we cannot establish the runtime values of the provided parameters.

Front-End Audit Summary

The supplied codebase contains the ReactJS UI components for the wallet functionalities, as well as the JS controllers used for interacting with the keyring. The keyring controllers offer support for multiple environments: BTC, ETH, Ronin, Harmony, and Cardano.

Threat Vectors Being Reviewed

Our testing was focused on the following attack vectors:

- **Dependency Confusion / Supply Chain attacks**
  - We check whether the application depends on any known risky third-party packages.
- **Data Storage**
  - We check whether the data access to the storage is limited and secure.
- **Cryptographic Primitives**
  - Check whether the cryptographic primitives in use adhere to the standards.
- **Hard-coded Secrets**
  - We check whether the applications contain or make use of hard-coded secrets.
- **Data Handling**
  - We check whether the implementation adopt good security practices on handling data processing.

Summary Review Results

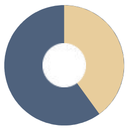
Below is a summary of our evaluation of front-end code against the above risks:

- **Dependency Confusion / Supply Chain attacks**
  - The application mostly uses Musk Identity's shared libraries, most of which were not provided. The other third- party dependencies that are used by the applications are either static or primitives of the respective platform which are unlikely to contain malicious code.
- **Data Storage**
  - The sensitive information corresponding to each keychain is encrypted throughout the lifecycle of the application within the Musk Identitywallet/app/scripts/controllers/keyrings/browser-passworder.ts file.
- **Insecure Cryptographic Primitives**
  - The cryptographic primitives in use mostly adhere to the standards, with the following exception:
  - The random number generator is a custom implementation, which has a potential risk. Detail is provided in the finding section.
- **Code Injection/Cross-Site Scripting**
  - The application sanitizes the input before injecting it into the UI components through ReactJS's sanitization mechanism. Furthermore, vulnerable calls such as dangerouslySetInnerHTML were not observed within the codebase.
- **Hard-coded Secrets**
  - The application does not contain or make use of hard-coded secrets.

REVIEW NOTES | Musk Identity

This security assessment is an extension of the previous Musk Identity's Blackbox penetration test performed on the Musk Identity wallet. The findings of the prior assessment are documented separately and are not included in this report.

FINDINGS | Musk Identity



5	0	0	0	2	3
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for Musk Identity. Through this audit, we have uncovered 5 issues ranging from different severity levels. Utilizing the techniques of Manual Review & Static Analysis to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
GLOBAL-01	SDK Uses Crypto Modules ThatAre No Longer Active	Coding Issue	Minor	Partially Resolved
GLOBAL-02	FrontEnd Use Insecure Cryptographic Primitives	Coding Issue	Minor	Partially Resolved
GLOBAL-03	Risky Implementation On Memory Free Operation	Coding Issue	Informational	Resolved
GLOBAL-04	Hard-Coded Secrets Found In SDK Code	Coding Issue	Informational	Resolved
GLOBAL-05	Potential Integer Overflow Risk In Data Handling.	Coding Issue	Informational	Resolved

GLOBAL-01 | SDK USES CRYPTO MODULES THAT ARE NO LONGER ACTIVE

Category	Severity	Location	Status
Coding Issue	Minor		Partially Resolved

Description

The SDK uses multiple sources for the module importing within both the JS and Golang components of the SDK. The JS component (audit/src/index.ts) uses the `crypto-js` module, which is no longer under active development.

The Golang components (audit/src/index.ts) use the `crypto-js` module, which is no longer under active development. It's recommended to replace the import with the native `Crypto` library.

Recommendation

It's recommended to replace the import with the native `Crypto` library.

Alleviation

The developers will replace the `crypto-js` module in a future release.

GLOBAL-02 | FRONTEND USE INSECURE CRYPTOGRAPHIC PRIMITIVES

Category	Severity	Location	Status
Coding Issue	Minor		Partially Resolved

Description

We found the cryptographic primitives used a custom random number generation implementation. It's using the `Math.random()` JS function, which is cryptographically insecure.

`Musk`

Recommendation

It's strongly recommended to use the `Crypto.getRandomValues()` function.

Alleviation

The developers will replace the `Math.random()` function with a more cryptographically secure generation method.

GLOBAL-03 | RISKY IMPLEMENTATION ON MEMORY FREE OPERATION

Category	Severity	Location	Status
Coding Issue	Informational		Resolved

Description

In the iOS app, we identified a snippet of code that potentially could free previously allocated memory. No concrete memory manipulation risks (UAF) were identified in the client's source code.

The memory being free here (ptr) could be already freed in other calls, and ptr is not set to NULL (which is not a must in free).

```
/** 释放某个指针 */
void tryToFreePtr(void *ptr){
    if(ptr == NULL){
        return;
    }
    free(ptr);
}
```

Recommendation

This snippet of code could be improved by null-ing the pointer values after free-ing, to prevent potential Use-after-Free (UAF) vulnerabilities by protecting against a dangling pointer:

```
void tryToFreePtr(void **ptr)
  { if (ptr && *ptr) {
    free(*ptr);      // Free the allocated memory
    *ptr = NULL;     // Set the original pointer to NULL
  }
}
```

It's also worth noting that if the pointer describes a nested structure, other nested pointers should also be freed beforehand, to avoid memory leaks.

Alleviation

The Client fixed the issue in the code snippet that was sent after the engagement.

GLOBAL-04 | HARD-CODED SECRETS FOUND IN SDK CODE

Category	Severity	Location	Status
Coding Issue	Informational		Resolved

Description

The okwallet-core component contains the AesEncrypt and AesDecrypt functions which use the encryption key that is hardcoded at `audit/okwallet-core/util/util.go:267` :

```
var AesHardKey = common.MD5("1697773335922")


// AesEncrypt AES 加密数据
func AesEncrypt(message, key string) (string, error)
    { if len(message) == 0 {
        return "", fmt.Errorf("message is empty")
    }
    if len(key) == 0
        { key =
        AesHardKey
    }
    return common.Encrypt(message, key)
```

No function calls towards these functions have been observed within the provided codebase. If this code snippet is not actively used, there are no security issues. **However, if this code is being used in other parts of the client's software, this potentially poses a serious security problem.**

Note: The reason this issue is undetermined is that we have only partial code. Although no reference to this hard coded secret was found in the code given to us, we can not conclude that code not shared with us did not use this hard-coded secret As such, the Severity is set as Informational, however it could be higher.

Recommendation

Remove this hard-coded secret from code, and make sure no other place in the software use it.

Alleviation

The hardcoded secret was removed from the codebase.

GLOBAL-05 | POTENTIAL INTEGER OVERFLOW RISK IN DATA HANDLING.

Category	Severity	Location	Status
Coding Issue	Informational		Resolved

Description

We identified a snippet of code that could allow an integer overflow if a sufficiently large integer is cast as a `uint32`. The snippet of code is foundhere: `bitcoin/brc20/util.go:15`

```
func ConvertToUint32(v string) uint32
    { i, _ := strconv.Atoi(v)
    return uint32(i)
}
```

Note: This function is actively used in the codebase but it is not possible to determine if there is a real risk of an integer overflow in the client's codebase, as we cannot establish the runtime values of the provided parameters. As such, the Severity is set as Informational, however it could be higher.

Recommendation

The snippet of code could be improved by using the `strconv.ParseInt` function.

Alleviation

The new version of the codeuses the `strconv.ParseInt` function in order toconvert the supplied string.

APPENDIX | Musk Identity

Finding Categories

Categories	Description
Coding Issue	Coding Issue findings are about general code quality including, but not limited to, coding mistakes, compile errors, and performance issues.

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 | CERTIK

This report is subject to the terms and conditions (including without limitation, description of services, con dentiality, 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.

CertiK | **Securing** the **Web3** World

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.