



Verida

**ARCHITECTURE & CODEBASE
SECURITY AUDIT**

19.11.2022

PUBLIC VERSION

Made in Germany by Chainsulting.de



Table of contents

1. Disclaimer.....	5
2. About the Project and Company	6
2.1 Project Overview.....	7
3. Vulnerability & Risk Level	8
4. Auditing Strategy and Techniques Applied.....	9
4.1 Methodology	9
4.2 Codebase Overview	10
5. Scope of Work	13
5.1 Findings Overview	14
5.2 Manual and Automated Vulnerability Test.....	17
5.2.1 Verida Vault (iOS & Android) & Verida Protocol	17

CONFIDENTIAL

CONFIDENTIAL

5.3 Focused Areas	73
5.3.1 Keyring (verida-js)	73
5.3.2 EncryptionUtils (verida-js)	73
5.3.3 Messaging inbox and outbox (verida-js)	74
5.3.4 Encrypted database (verida-js)	74
5.3.5 Storage Node Authorization (storage-node)	75
5.3.6 Storage Node Authorization (vault-auth-server)	75
6. Executive Summary	76
8. About the Auditor	77

1. Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only.

The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of Verida Pte Ltd. If you are not the intended receptor of this document, remember that any disclosure, copying or dissemination of it is forbidden.

Major Versions / Date	Description
0.1 (25.02.2022)	Layout
0.2 (02.03.2022)	Test Deployment
0.5 (10.03.2022)	Automated Security Testing Manual Security Testing
0.6 (12.03.2022)	Development Productivity (Code Conventions Check, packages)
0.7 (16.03.2022)	Performance (Connection pooling, caching, Transaction Concurrency)
0.8 (20.03.2022)	Maintainability & Ease of Deployment
0.9 (23.03.2022)	Summary and Recommendation
1.0 (28.03.2022)	Final document

2. About the Project and Company

Company address:

Verida Pte Ltd
Reg.: 202028969C
160 Robinson Road
#14-04 Singapore Business Federation Centre
Singapore 068914



Website: <https://www.verida.io>

Twitter: https://twitter.com/verida_io

Discord: <https://discord.com/invite/qb6vS43>

Medium: <https://news.verida.io>

LinkedIn: <https://www.linkedin.com/company/verida-technology>

Telegram: https://t.me/verida_community

GitHub: <https://github.com/verida>

Blog: <https://news.verida.io>

2.1 Project Overview

Verida is a network of personal data owned and controlled by users. Users are incentivized to unlock their data stored on centralized platforms. Builders access this data for new exciting use cases such as trusted storage, decentralized messaging and single sign on. User's private data can be used as inputs into smart contracts, enabling connectivity to multiple blockchains.

Verida provides a protocol of servers, libraries and SDKs to enable a network of users to create decentralized identities that can access deconcentrated services on the Verida Network. These offerings include database and block storage, messaging, notifications, and blockchain interoperability. The Verida Protocol leverages the rapidly evolving Web3 technology stack, is natively multi-chain, and adds missing decentralized services and technical capabilities where necessary.

Verida provides software SDKs that developers can integrate into their applications to leverage the technical capabilities of the Verida Protocol. Verida provides open-source server nodes that can be operated by anyone on their own infrastructure, to earn tokens on the Verida Network. Verida facilitates decentralized network services as centralized APIs will transition across to smart contracts on the Verida Network once their technical requirements stabilize. Verida provides a reference implementation of a 'Data Wallet' (Verida Vault), a mobile application for end users to create decentralized identities, securely store their private keys, and interact with the Verida Network with its supported blockchains.

3. Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 – 10	A vulnerability that can disrupt the code functioning in a number of scenarios or creates a risk that the code may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a codebase, or provides the opportunity to use an application in an unintended way.	Implementation of corrective actions as soon as possible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executing the code in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the code and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

4. Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pen testers and developers, documenting any issues as there were discovered.

4.1 Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
 - i. Review of the specifications, sources, and instructions provided to Chainsulting to make sure we understand the size, scope, and functionality of the codebase.
 - ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
 - iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Chainsulting describe.
2. Testing and automated analysis that includes the following:
 - i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
 - ii. Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.
3. Best practices review, which is a review of the codebase to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, actionable recommendations to help you take steps to secure your codebase.

4.2 Codebase Overview

Verida Vault

Source: <https://github.com/verida/vault-mobile>

Commit: 02b7fa610aeb6356b5865ba07f35194a12721ff3

Total : 277 files, 15701 codes, 282 comments, 1677 blanks, all 17660 lines

language	files	code	comment	blank	total
JavaScript	140	9,770	162	1,009	10,941
TypeScript React	42	3,801	38	418	4,257
XML	78	1,113	0	78	1,191
TypeScript	14	1,014	82	172	1,268
JSON	3	3	0	0	3

Verida Protocol

1. Name: verida-js

Source: <https://github.com/verida/verida-js>

Commit: a39619b438af073a51a3c5c7d253dd3205891297

Total : 138 files, 7489 codes, 1520 comments, 1817 blanks, all 10826 lines

language	files	code	comment	blank	total
TypeScript	93	6,621	1,499	1,657	9,777
JSON	11	351	21	10	382
Markdown	22	294	0	149	443
JSON with Comments	11	192	0	0	192
XML	1	31	0	1	32

2. Name: storage-node

Source: <https://github.com/verida/storage-node>

Commit: f515b5cfb8e7c69782cd5163c70c496519293d37

Total : 7 files, 502 codes, 62 comments, 101 blanks, all 665 lines

language	files	code	comment	blank	total
JavaScript	7	502	62	101	665

3. Name: vault-auth-server

Source: <https://github.com/verida/vault-auth-server>

Commit: 584352561f55f390b3bf64f5af2d1ed51bbbbeac0

Total : 5 files, 309 codes, 18 comments, 68 blanks, all 395 lines

language	files	code	comment	blank	total
JavaScript	5	309	18	68	395

4. Name: wallet-utils

Source: <https://github.com/verida/wallet-utils>

Commit: ed20ffc504c83800cb3ac472f821d4a85d3ff5e4

Total : 7 files, 339 codes, 76 comments, 77 blanks, all 492 lines

language	files	code	comment	blank	total
TypeScript	7	339	76	77	492

5. Scope of Work

The Verida Team provided us with the files that needs to be tested. The scope of the audit is the Verida architecture and codebase.

1. Automated Vulnerability Test (OWASP, Acunetix, Sonarsource, Snyk, jsfuzz..)
2. Manual Security Testing (Line by line, Overflow, CVE, MAST etc.)
3. Test environment deployment
4. Evaluating and testing software architecture
 - Development Productivity (Code Conventions Check, packages)
 - Functions & Logic Testing
 - Performance (Connection pooling, caching, Transaction Concurrency)
 - Reliability & Availability
 - Maintainability & Ease of Deployment

Particular areas are focused on the audit of the protocol

Keyring: <https://github.com/verida/verida-js/blob/main/packages/keyring/src/keyring.ts>

AutoAccount of Keyring: <https://github.com/verida/verida-js/blob/7bffc4e4d4a1c7ab86838cf6a55bb4da9d65ac46/packages/account-node/src/auto.ts#L33>

EncryptionUtils: <https://github.com/verida/verida-js/tree/main/packages/encryption-utils>

Messaging inbox and outbox: <https://github.com/verida/verida-js/tree/main/packages/client-ts/src/context/engines/verida/messaging>

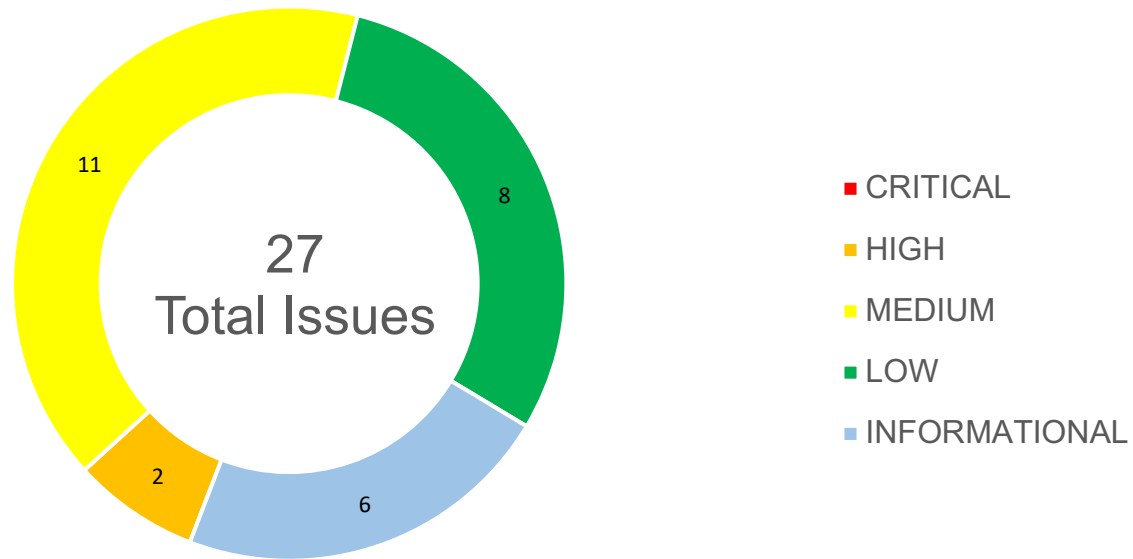
Encrypted database: <https://github.com/verida/verida-js/blob/7bffc4e4d4a1c7ab86838cf6a55bb4da9d65ac46/packages/client-ts/src/context/engines/verida/database/db-encrypted.ts#L60>

Storage Node Authorization: <https://github.com/verida/storage-node/blob/main/src/middleware/requestValidator.js>

Vault Auth Server SessionManager: <https://github.com/verida/vault-auth-server/blob/main/src/SessionManager.js>

The main goal of this audit was to make sure the infrastructure is built according to newest standards and securely developed. The auditors can provide additional feedback on the code upon the client's request.

5.1 Findings Overview



No	Title	Severity	Status
CONFIDENTIAL			

CONFIDENTIAL



CONFIDENTIAL

5.2 Manual and Automated Vulnerability Test

5.2.1 Verida Vault (iOS & Android) & Verida Protocol

Details (iOS)

Application Name	Verida Vault
Platform	iOS
Application Namespace	io.verida.vault.internal
Version	0.0.14
Version Code	2
Application SHA1 Hash	a17a9562e10ad0810373f41aa4ea1c75119ea4ce
Application MD5 Hash	41998bffe79d729574e94894da350cd3
Scans	Static, Dynamic, API, Manual
Language	React Native

Details (Android)

Application Name	Verida Vault Internal
Platform	Android
Application Namespace	io.verida.vault.internal
Version	1.0.1
Version Code	2
Application SHA1 Hash	1123a796bf5ac85df949557ea9ee62f7f126ec44
Application MD5 Hash	d4655c5f1cec36e43b5315a5fc52f573
Scans	Static, Dynamic, API, Manual
Language	React Native

Test Summary (App)

Vulnerability Title	Test Type	Status
TLS Protocol Downgrade Attack	API	✗
General Server Vulnerabilities	API	✗
Buffer Overflow Vulnerabilities in HTTP Requests	API	✓
Command Injection Vulnerabilities in HTTP Requests	API	✓
Integer Overflow Vulnerabilities in HTTP Requests	API	✓
JSON Depth Overflow in HTTP Requests	API	✓
LDAP Injection Vulnerabilities in HTTP Requests	API	✓
Regex DoS Vulnerabilities in HTTP Requests	API	✓
SQL Injection Vulnerabilities in HTTP Requests	API	✓
String Validation Vulnerabilities in HTTP Requests	API	✓
XML-external-entity Injection Vulnerabilities in HTTP Body	API	✓
Cross-site-scripting Vulnerabilities in HTTP Body	API	✓
CORS Wild Character Vulnerabilities in HTTP Headers	API	✓
Cross Site Tracing Vulnerabilities	API	✓
Response Body Contains Non-HTTPS Links	API	✓
HTTP TRACE method is enabled	API	✓
HTTP Host Header Injection	API	✓
OpenSSL CCS Injection Vulnerability	API	✓
Heartbleed Vulnerability	API	✓

TLS ROBOT Attack	API	✓
TLS/SSL CRIME Attack	API	✓
SSL/TLS Renegotiation Vulnerability	API	✓
Unsecured Keychain Data	Manual	✗
Exposed Pasteboard Data	Manual	✗
Improper Session Management	Manual	✓
Sensitive Information Disclosure	Manual	✗
Business Logic	Manual	✓
Buffer Overflows and Underflows	Manual	✓
One Time Password Bypass	Manual	✓
Insecure Direct Object Reference	Manual	✓
Misconfigured AWS S3 Buckets	Manual	✓
Insecure Cookie Attributes	Manual	✓
Stack Trace Enabled	Manual	✓
Insecure Biometric Authentication	Manual	✓
Debug Logging with NSLog	Dynamic	✓
Storing Information in Shared Preferences	Dynamic	✓
Application Logs	Dynamic	✗
Sensitive information in Sqlite database	Dynamic	✓
Derived Crypto Keys	Dynamic	✓
Sensitive Information in Property Lists	Dynamic	✓
Sensitive Data in NSUserDefaults	Dynamic	✓

Unsecured Data in CoreData	Dynamic	✓
Unsecured Data in CouchDB	Dynamic	✓
Unsecured Data in RealmDB	Dynamic	✓
Unsecured Data in YapDB	Dynamic	✓
Deprecated NSURLConnection	Dynamic	✓
Insecure Cryptographic Keys	Dynamic	✓
iOS SecKeyEncrypt implementation	Dynamic	✓
Insecure Peer Connections	Dynamic	✓
UIWebView Exploits	Dynamic	✓
Insufficient Transport Layer Protection	Dynamic	✓
Short HMAC Keys	Dynamic	✓
Vulnerable Hash Algorithms	Dynamic	✓
Zipperdown Vulnerability leading to Remote Code Execution Attack	Dynamic	✓
App Transport Security	Static	✗
Disabled SSL CA Validation and Certificate Pinning	Static	✗
MediaProjection: Android Service Allows Recording of Audio, Screen Activity	Static	✗
Unused Permissions	Static	✗
Unprotected Exported Activities	Static	✗
Bytecode Obfuscation	Static	✗
Unprotected Services	Static	✓
Improper Content Provider Permissions	Static	✓
Improper Custom Permissions	Static	✓

Broken SSL Trust Manager	Static	✓
Broken HostnameVerifier for SSL	Static	✓
Insecure SSLSocketFactories	Static	✓
HostnameVerifier Allowing All Hostnames	Static	✓
App Extending WebViewClient	Static	✓
JavascriptInterface Remote Code Execution	Static	✓
Remote URL Redirection Vulnerability	Static	✓
Content Provider File Traversal Vulnerability	Static	✓
Android Fragment Injection	Static	✓
Android Component Hijacking via Intent	Static	✓
Network Security Misconfiguration	Static	✓
PhoneGap Debug Logging	Static	✓
PhoneGap Whitelist Open Access	Static	✓
PhoneGap Whitelist RegEx Bypass	Static	✓
iOS Binary having ASLR Protection	Static	✓

CRITICAL ISSUES

During the audit, Chainsulting's experts found **no Critical issues** in the codebase and architecture

CONFIDENTIAL

CONFIDENTIAL

CONFIDENTIAL

CONFIDENTIAL

CONFIDENTIAL

MEDIUM ISSUES

CONFIDENTIAL

CONFIDENTIAL

CONFIDENTIAL

CONFIDENTIAL

CONFIDENTIAL

CONFIDENTIAL

CONFIDENTIAL

CONFIDENTIAL

CONFIDENTIAL

CONFIDENTIAL

CONFIDENTIAL

The application was seen using a general pasteboard for copying sensitive data like uid and

CONFIDENTIAL

CONFIDENTIAL

CONFIDENTIAL

CONFIDENTIAL

CONFIDENTIAL

CONFIDENTIAL

applications can execute the logcat command.

CONFIDENTIAL

CONFIDENTIAL

CONFIDENTIAL

CONFIDENTIAL

CONFIDENTIAL

CONFIDENTIAL

CONFIDENTIAL

CONFIDENTIAL

CONFIDENTIAL

CONFIDENTIAL

LOW ISSUES

CONFIDENTIAL

CONFIDENTIAL

CONFIDENTIAL

CONFIDENTIAL

CONFIDENTIAL

CONFIDENTIAL

CONFIDENTIAL

CONFIDENTIAL

intended. If the duplication was intentional, but may be confusing to maintainers.

CONFIDENTIAL

CONFIDENTIAL

CONFIDENTIAL

Vendor: Vendor, components: Components, Vendor data server: Data server,

CONFIDENTIAL

INFORMATIONAL ISSUES

CONFIDENTIAL

CONFIDENTIAL

CONFIDENTIAL

CONFIDENTIAL

CONFIDENTIAL

// @todo Cache databases so we don't open the same one more than once



CONFIDENTIAL

CONFIDENTIAL

Line 04 (repeated)


CONFIDENTIAL

CONFIDENTIAL


CONFIDENTIAL

5.3 Focused Areas


5.3.1 Keyring (verida-js)

Source	Keyring: https://github.com/verida/verida-js/blob/main/packages/keyring/src/keyring.ts AutoAccount of Keyring: https://github.com/verida/verida-js/blob/7bffc4e4d4a1c7ab86838cf6a55bb4da9d65ac46/packages/account-node/src/auto.ts#L33
Description	Class that takes a signature (generated from a signed consent message) and generates a collection of asymmetric keys, symmetric key and signing key for a given secure storage context.
Test	Encryption:  Cryptographic vulnerabilities: None Issues: 5.2.1.21


5.3.2 EncryptionUtils (verida-js)

Source	EncryptionUtils: https://github.com/verida/verida-js/tree/main/packages/encryption-utils
Description	Encryption utilities to make using tweetnacl a bit easier. Utilizes tweetnacl for symmetric and asymmetric encryption. Utilizes keccak256 algorithm to hash signed data and secp256k1 signature algorithm for the resulting signature. TweetNaCl is an audited encryption library, which uses the xsalsa20-poly1305 algorithm, it also got audited https://cure53.de/tweetnacl.pdf .
Test	Encryption:  Cryptographic vulnerabilities: None Issues: 5.2.1.25


5.3.3 Messaging inbox and outbox (verida-js)

Source	Messaging inbox and outbox: https://github.com/verida/verida-js/tree/main/packages/client-ts/src/context/engines/verida/messaging
Description	Every application has a built-in inbox for receiving messages and outbox for sending messages. This allows users and applications to send data between each other knowing nothing than the other user's DID and application name.
Test	Encryption:  Cryptographic vulnerabilities: None Issues: 5.2.1.21, 5.2.1.24, 5.2.1.25


5.3.4 Encrypted database (verida-js)

Source	Encrypted database: https://github.com/verida/verida-js/blob/7bffc4e4d4a1c7ab86838cf6a55bb4da9d65ac46/packages/client-ts/src/context/engines/verida/database/db-encrypted.ts#L60
Description	Encrypted sync of PouchDB
Test	Encryption:  Cryptographic vulnerabilities: None Issues: 5.2.1.21, 5.2.1.25

5.3.5 Storage Node Authorization (storage-node)

Source	Storage Node Authorization: https://github.com/verida/storage-node/blob/main/src/middleware/requestValidator.js
Description	Allow access to any user who provides a valid signed message for the given application
Test	Encryption:  Cryptographic vulnerabilities: None Issues: 5.2.1.24

5.3.6 Storage Node Authorization (vault-auth-server)

Source	Vault Auth Server SessionManager: https://github.com/verida/vault-auth-server/blob/main/src/SessionManager.js
Description	Session Management is handling the socket connections, generation of JWT and garbage collection.
Test	Encryption:  Cryptographic vulnerabilities: None Issues: None

6. Executive Summary

Three (3) independent Chainsulting experts performed an unbiased and isolated audit of the architecture and codebase. The final debriefs took place on the March 25, 2022.

The main goal of this audit was to make sure the infrastructure is built according to newest standards and securely developed. During the audit, no critical issues were found, after the manual and automated security testing. Overall, the code quality and architecture had a high grad of professionalism.

We recommend the following things to apply to the next dev ops:

1. Check our issues and get them fixed
2. Make sure newest packages are used, wherever possible
3. Use linter for static code analysis to flag programming errors, bugs, stylistic errors and suspicious constructs

8. About the Auditor

Chainsulting is a professional software development firm, founded in 2017 and based in Germany. They show ways, opportunities, risks and offer comprehensive web3 solutions. Their services include web3 development, security and consulting.

Chainsulting conducts code audits on market-leading blockchains such as Solana, Tezos, Ethereum, Binance Smart Chain, and Polygon to mitigate risk and instil trust and transparency into the vibrant crypto community. They have also reviewed and secured the smart contracts of 1Inch, POA Network, Uniswap, LUKSO among numerous other top DeFi projects.

Chainsulting currently secures [\\$100 billion](#) in user funds locked in multiple DeFi protocols. The team behind the leading audit firm relies on their robust technical know-how in the web3 sector to deliver top-notch smart contract audit solutions, tailored to the clients' evolving business needs.

Check our website for further information: <https://chainsulting.de>

How We Work



1 -----

PREPARATION

Supply our team with audit ready code and additional materials



2 -----

COMMUNICATION

We setup a real-time communication tool of your choice or communicate via e-mails.



3 -----

AUDIT

We conduct the audit, suggesting fixes to all vulnerabilities and help you to improve.



4 -----

FIXES

Your development team applies fixes while consulting with our auditors on their safety.



5 -----

REPORT

We check the applied fixes and deliver a full report on all steps done.