venclave security whitepaper

Last modified: 14/03/2018.

This document describes the various security measures in place on the “venclave.com” online platform.
All of this concerns any customer input data and the data derived from manual or automatic actions on that input data.
Altough this will not describe everything in technical detail, it should be enough to illustrate the implementation.

1. Web frontend

1.1 HTML standard

We try to use as much HTML5-compliant code as possible, taking into account current browser shortcomings.
This means applicable entities will describe the required charset, possible format and minimum & maximum length.

1.2 Links, Images, CSS, Javascript

Javascript is only used where strictly possible, and offered via separate javascript files that are loaded asynchronously without any credentials supplied.
Images are hosted on the platform under the venclave domain.
Links are inserted with the rel=noopener attribute, so redirected pages have no javascript DOM access to the previous web page.
CSS is hosted on separate files, loaded asynchronously without any credentials supplied.
All the assets are protected by a Content-Security-Policy header, only allowing assets to be loaded from the venclave.com domain and from separate files.
An exception to this is the learning.venclave.com and handler.venclave.com domains, which can host custom web pages and custom learning campaigns.

1.3 Security headers

We score an A+ score on our security headers, which you can check at securityheaders.io/venclave (https://securityheaders.io/?q=venclave.com&followRedirects=on).

Clickjacking is remediated by using the X-FRAME-OPTIONS header.
Content-Type sniffing is remediated by using the X-Content-Type-Options header.
Browser XSS protection is activated by supplying a X-XSS-Protection header.
venclave.com is forced to only run over a protected TLS connection by suppling a Strict-Transport-Security header and appearing on the hsts preload list (https://hstspreload.org/?domain=venclave.com).
We are not leaking your usage of our platform to external links by supplying a Referrer-Policy header.
The latest Expect-CT header is supplied as wel in reporting mode.
As mentioned in section 1.2, Content-Security-Policy header is fully enforced to prevent injection of malicious Javascript on our platform.
We use three different headers for maximum browser compatibility: Content-Security-Policy, X-Content-Security-Policy and X-WebKit-CSP.

1.4 TLS configuration

Our servers score an A+ score on our TLS configuration, which you can view on the Qualys SSL Labs report (https://www.ssllabs.com/ssltest/analyze.html?d=venclave.com&s=104.18.37.1).
To reach an optimal audience, we use a custom cipherlist without any known security vulnerabilities. (TLS 1.0 and up)
This should prevent anyone from decrypting the TLS stream.
Minimum supported client versions are: Android 4.0.4, Windows 7, Java 7, OSX 10, iOS 6.

1.5 Local visitor Storage

We try to limit the amount of data that is stored on customer's end for security and efficiency reasons.
Two cookies are the only thing stored in the browser: one for tracking the application session, and one for tracking the visitor on our DDoS perimeter.
Both cookies carry the HttpOnly and Secure attributes, which prevent them from being leaked or tampered with

1.6 Denial of Service protection

We have a high-bandwidth perimeter between us and our visitors for a number of reasons, all over the world.
Our external perimeter ensures quick delivery of our platform by compressing our web responses and delivering from the nearest Point-Of-Presence.
This also delivers protection against Distributed Denial of Service attacks, which can block or offer captchas to the offending users.

1.7 Database Storage

Every sensitive piece of user data is encrypted using an industry-standard symmetric encryption scheme before being stored in our databases.
These encryption keys are only available to our application servers, allowing for zero impact in case our database gets leaked.
Passwords are stored using a salted password hash using the argon2i algorhytm which won several competitions.

2. Backend infrastructure

2.1 Hardware

Our services run on virtual hardware with dedicated resources, running on an encrypted filesystem to protect against cold-boot attacks or theft.
Virtual instances were chosen instead of dedicated hardware because of disaster recovery reasons.
On this hardware every service runs in a slimmed-down dedicated container. These containers talk to each other over a separate, encrypted network.
Daily backups allow our services to be backed up in their complete form to an off-shore server.

2.2 Containerisation

We run all of our infrastructure on unprivileged containers that are remapped to a non-privileged user on the host system.
This ensures that even breaking out of a container still results in zero privilege.
The use of containers also force that the test environment is exactly the same as our production environment.

2.3 Operating system

All of our container images use the same minified image that relies on a stable release of an industry-standard operating system.
This allows for micro builds of our containers, thus reducing the attack surface because non-used packages are removed.

2.4 Software packages

We try to follow stable releases of all of our software as close as possible, applying bugfix releases whenever possible.
Our development builds automatically check for any newer releases.

2.5 Storage

We use SAN storage offered by our infrastructure provider to be able to offer high availibility of our platform.
Data that is stored on disk via the database is encrypted using AES-256 symmetric encryption keys.
Session storage is only stored in memory and does not contain any sensitive information, safe for the session key.

2.6 Backend communication

Backend communication happens over the JSON protocol, which carries an assymetricly encrypted, authenticated payload. Every node bears his own public/private keypair, with the public keys of our infrastructure pinned on every node. Different encryption keys are used over environments.

2.7 Security by Design

The platform has been designed in a way that completely makes Sql Injection, Cross-Site Scripting and Cross-Site Request Forgery impossible.
Security measures have been added from the ground up.

3. Development

3.1 Revision Control

Our hardware & application code is committed to a central code repository over a secure protocol.
This allows for a full audit log with signed commits and to roll back to previous versions.

3.2 Integrated Development Environment

During development our IDE will alert against any functional and security problems that might arise in our code.

3.3 Manual Code Review

Our code is periodically peer-reviewed to ensure good quality of code and no security vulnerabilities.

3.4 Static Code Analysis

Every couple releases, our code is scanned by tools to detect any possible code bugs or security issues.
This is done as wel during development.

4. Data Retrieval (GDPR)

Users always have the possibility to retrieve their data in question or get it deleted within 2 business days when filing a request at help@venclave.com (mailto:help@venclave.com)