Encircle

151 Charles St. West, Suite 100
Kitchener, Ontario
Canada, N2G 1H6

Security Overview

Encircle's customer information is stored on, or transits through, the following components:

- A PostgreSQL database (containing most of the user data).
- An object store containing binary blob data. Uploaded pictures, videos and documents live here, with only references to them stored in the PostgreSQL database.
- A set of webservers (which serve the website and provide API access for mobile devices).
- The end user's own mobile phone (which keeps a copy of the user's data to work offline), or browser (which has access to the data when running the web application).

Access Controls

The database runs on a Microsoft Azure virtual machine instance. Access to it is limited to port 22 (SSH) from the outside world, and to port 5432 (PostgreSQL) from the web servers only. This connection is encrypted to stop passive listeners from eavesdropping on the traffic and authenticated to stop man-in-the-middle attacks.

The webservers are only accessible on ports 22 (SSH), 80 (HTTP), and 443 (HTTPS). SSH access, whether to the webservers or databases is only possible through authorized keys, which only 2 employees in the company have. All webservers run a regularly updated installation of Ubuntu.

Transport Security

Communications between the user's device and our servers is always done over HTTPS. All attempts to connect to HTTP, which is unencrypted, using a browser, are logged and redirected to a secure site instead. To ensure greater security, we use HTTP Strict Transport Security[1] to disallow any insecure connections to our webservers in modern browsers. We've also been added to the list of hardcoded HTTPS-only websites[2], securing connections for all Chrome, Firefox, Safari, and future IE users.

We limit the set of encryption algorithms and protocols to those considered strong and secure. This means avoiding obsolete and insecure tools like SSLv2/v3, RC4, DES, and MD5. Our security level is thus highly rated by SSLLabs' tool[3].

Authentication

User authentication does not rely exclusively on HTTP cookies, which are vulnerable to a host of cross-site-request-forgery risks. Instead the API uses a secure bearer token which must be included as a header in every request as an additional protection.

(519) 573-1186 team@encircleapp.com

All actions that are especially destructive are further protected by re-requesting the user's password to authorize the action.

Every session uses a distinct identifier, which can be individually disabled. This allows access to be remotely revoked in case the user's device is stolen or otherwise compromised.

The password reset process is tied exclusively to the email address that was used to register for Encircle.

Privacy

All database queries use a system called the ACL Graph, which guarantees that database queries can only access information that the user is allowed to see. This is used when reading, updating or creating items. That way, even if an API endpoint mistakenly tries to return all pictures in the system, only pictures that the user is allowed to see will be made available.

The API never makes ad-hoc privacy checks; Encircle has been built from the ground up to incorporate privacy as part of the core system.

Password Storage

Encircle hashes all passwords with an industry standard secure password hash (bcrypt, cost 14). The plaintext password is encrypted in transit over HTTPS and only the hashed copy is ever stored in the database. It is computationally infeasible for an attacker to reverse this hash into the original password.

Many other password breaches mentioned in recent news arise from poor choice in hashing algorithm (e.g. using a cryptographic hash instead of a password hash) or poor attention to secure methodology (e.g. not salting the hashes).

Mitigated Attacks

The Encircle API server is built atop a small set of core abstractions that are easy to audit and ensure the security of the entire system. These core abstractions mitigate and nullify a number of common attacks many of which make the OWASP top 10 web attacks[4] year after year.

SQL Injection

Encircle

151 Charles St. West, Suite 100
Kitchener, Ontario
Canada, N2G 1H6

Encircle uses an Object Relational Mapper called SQLAlchemy to formulate all database queries. SQLAlchemy automatically separates query parameters from the query itself. It is thus impossible for user provided input to escape the query parameter and perform a SQL Injection.

Cross Site Scripting (XSS)

Encircle uses tools for the server (Tornado Templates) and the clients (React) that automatically sanitize user input from the structure of the web page, preventing XSS entirely. As an additional layer of protection, Content Security Policy headers are given, disabling the loading of scripts or content from any domain not specifically whitelisted.

Timing Attacks

Timing attacks are a specific breed of side channel attacks where an attacker exploits the time taken by the server to respond to reason about various code paths.

All authentication for the Encircle API is handled by a single core layer that is timing independent. Each code path is the same length and thus no sensitive information can be inferred to attack the authentication layer.

CRIME and BREACH

The BREACH (and its predecessor CRIME) attacks exploit compression artifacts in the core connection protocols of the internet to reveal sensitive information. Announced in August 2013, it is comparable in scope to Heartbleed, though it is not as easy to exploit. A large number of services continue to be vulnerable to this however.

BREACH relies on sensitive information being in the document for every request an active attacker makes. Encircle generates a random nonce for each request and only responds with the secure token exclusively OR'd with the nonce. This way, the token is represented differently for each request, which mitigates BREACH but still allows the clients to reconstruct the actual token.

---

[1] https://tools.ietf.org/html/rfc6797
[2] https://code.google.com/p/chromium/codesearch#chromium/src/net/http/transport_security_state_static.json&l=782
[3] https://www.ssllabs.com/ssltest/analyze.html?d=encircleapp.com
[4] https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project