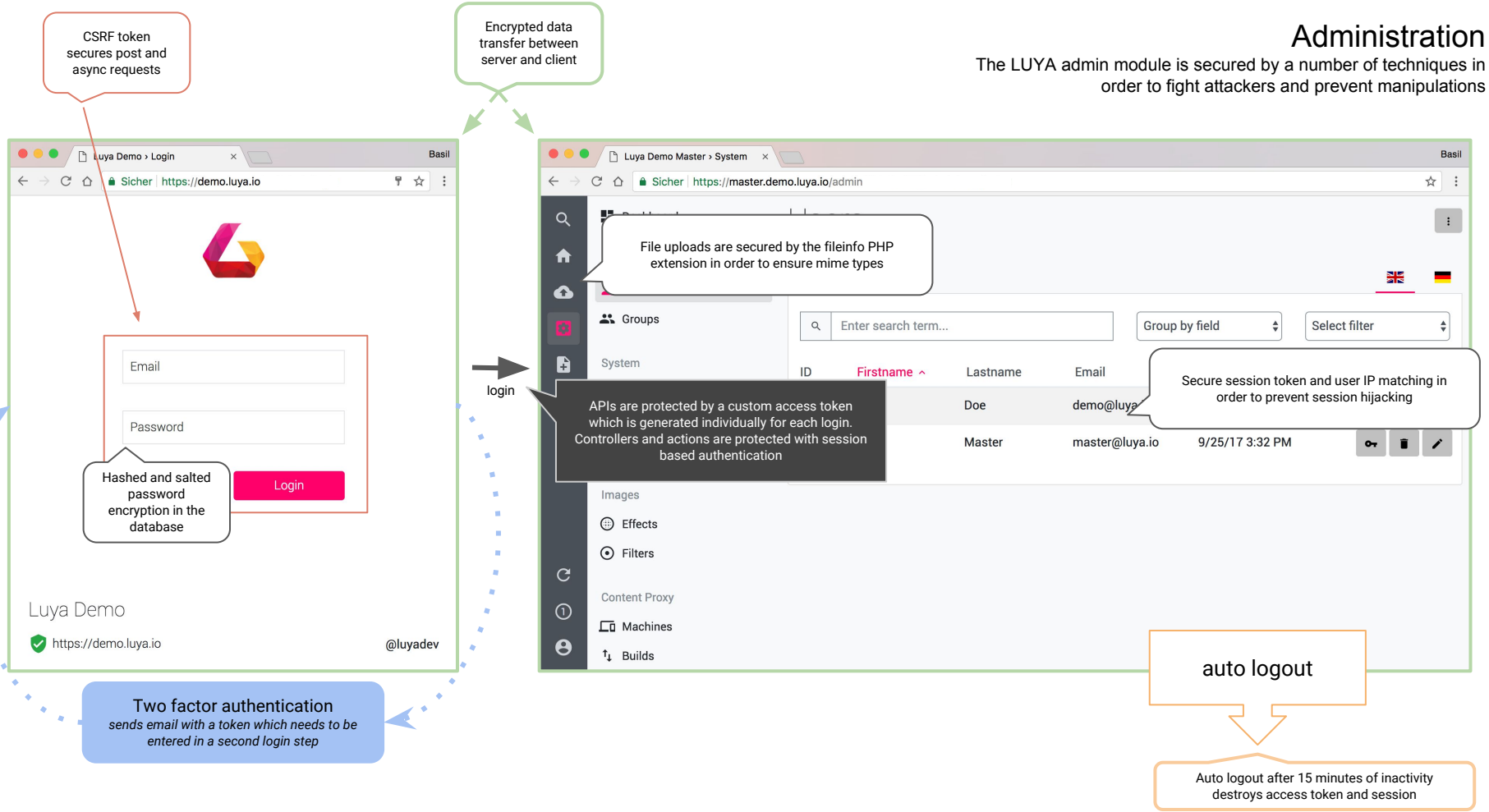
LUYA SECURITY

A brief visual overview of what we do in order to secure your web application

Administration

The LUYA admin module is secured by a number of techniques in order to fight attackers and prevent manipulations



Administration Infos

In addition to security techniques LUYA also keeps track of what is done by whom and when

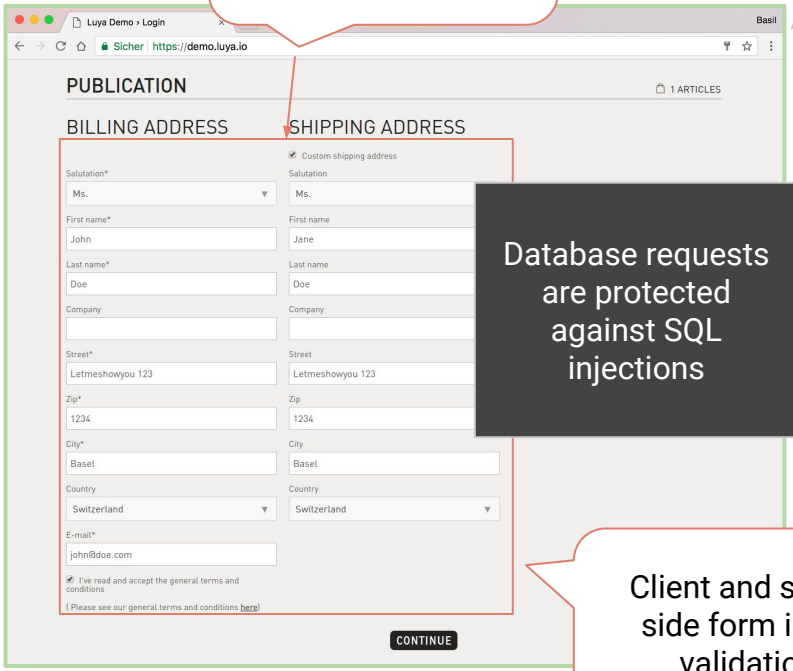
- Any data change in the administration is logged with information about who changed what and when.
- Any user login is documented, with the user's IP address and access token.
- A user can only be logged in once. Concurrent logins by the same user are avoided by terminating the previous session.
- The two-way factor authentication sends a custom token to the user's email which has to be entered to complete login, therefore brute forcing is hindered and insecure passwords become less of a risk.
- We do not provide extension installation via a web interface – all extensions and modules are implemented via composer, therefore versioning and bug fixing is enforced.
- LUYA stores configurations and data in files so they can be tracked via VCS systems like GIT and a full change history is provided.

Frontend

Security measures of the website's frontend

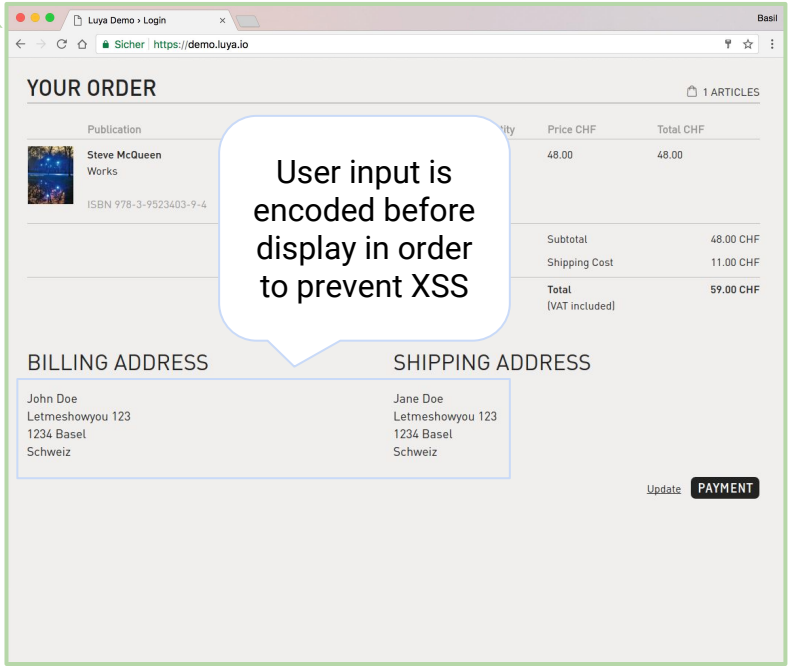
CSRF token secures requests

Encrypted data transfer between server and client



Database requests are protected against SQL injections

save



User input is encoded before display in order to prevent XSS

Client and server side form input validation

Prevention technics in depth

The frontend technics used to protect the application.

- All database requests are protected against SQL injections by the Yii 2 database abstraction layer, **data binding (filtering out malicious inputs) is used for all SQL statements**.
- CSRF: Cross-site request forgery (CSRF) is a typical web application vulnerability. It is based on the assumption that a user is authenticated at a legitimate website. Then he's visiting an attacker's website which issues requests to the legitimate website using JavaScript code, a form, <img src=""> tag or other means. This way, attackers could, for example, reset a victim's password or transfer funds from his bank account (in case the bank's website isn't secure, of course). In order to prevent such request forgery, **we use an encrypted token which is stored on the server and client side and is compared on each request**.
- MITM: Man in the middle attacks are prevented by using encrypted data transfers between client and server as **we use SSL**.
- XSS: Cross site scripting is commonly a problem when user data is returned, therefore **we use an encoding and HTML purifying technique**.
- To prevent attackers from stealing the cookie used to authenticate the user on the remote server and create a false identity to take over the user's session, **we store the access token in combination with the IP address and compare those values on each request.**
- File uploads to the storage system can contain dangerous files which then expose system informations. **We prevent this with a secure file upload which uses the PHP fileinfo extension to deep check mime types.**