

Dokeos 1.8.5 Security Quick Reference Guide

General web security info

1. Security problems are due, most of the time, to lazy or uninformed developers considering that one piece of code is not **vulnerable** because it is only accessible to reliable people
2. Security flaws for PHP can come from two main sources:
 - mistakes
 - intentional attacks...but they always come from the outside: users
3. Security flaws can be avoided, most of the time, by filtering user input

Risks for the Dokeos system

Database (SQL injection)

How: by sending specially-modified parameters in Dokeos URL or forms

Why: to obtain or destroy the Dokeos data

How to avoid: Use `Database::escape_string()` or (int) casting on each variable just before any SQL query

Remote file execution

How: upload a script to the server OR abuse an dynamic include parameter (like `$includePath` in `index.php`) to load a PHP script on another server

Why: to gain complete access to your server and steal or destroy not only your database but all the files on your server

How to avoid: Filter every and each parameter to make sure including an external resource, or uploading a PHP script to the server is impossible. This can be done by using the `filter_extension()` and the `disable_dangerous_file()` functions (see `main/inc/lib/fileUpload.lib.php`) and through the `Security::check_abs_path()` and `Security::check_rel_path()` functions

Risks for the Dokeos users

XSS attacks (Cross-Site Scripting)

How: use JavaScript to direct the user's browser to another site for one or more requests

Why: to either steal user information or force him to go to a specific site, to make him believe he is on a reliable site and make him do things out of trust.

How to avoid: Filter **any** text that is later (not only directly) shown to the same or another user. This is done by using `Security::remove_XSS()`

CSRF (Cross Site Request Forgery)

How: use JavaScript to make the user's browser execute simple tasks on other websites where the user has an account, abusing the cookies currently stored in the browser.

Why: This enables ordering products in web shops, posting articles under one's name or changing one's password on an internet bank's site.

How to avoid: Filter **any** text that is later (not only directly) shown to the user, and make it impossible for his browser to re-use forms. This is done through a more complex mechanism combining `Security::get_token()` and `Security::check_token()` respectively before and after any form submission.

References

1. Check `main/inc/lib/security.lib.php` for more of Dokeos existing security features.
2. Check Chris Shiflett's publications and particularly the PHP Security Consortium's PHP Security Guide:
<http://phpsec.org/projects/guide/>