Web Application Security Checklist

Objectives

We want to help developers making their web applications more secure. This checklist is supposed to be a brain exercise to ensure that essential controls are not forgotten.

Items on this list are frequently missed and were chosen based on their relevance to the overall security of the application. It's a starting point.

Visit the checklist at https://probely.com/checklist for more detailed information.

Probely

https://probely.com
hello@probely.com

Web Application Security Checklist

General security

- [ ] I use prepared statements in SQL queries
- [ ] I do not concatenate any other input data to SQL queries other than the bound parameters
- [ ] I validate all input data server-side
- [ ] I encode all input data before sending the response to the browser
- [ ] I have disabled directory listing in the web server
- [ ] I include a CSRF token in requests that change state (or I use the SameSite cookie attribute for the session cookie)
- [ ] I do not show errors with stack traces, source code, full paths or any other internal data.
- [ ] I verify the content type of uploaded files and delete the bad ones
- [ ] If I handle XML files, I disabled external entity and DTD processing
- [ ] I use HTTPS and I send the Strict-Transport-Security header
- [ ] I only accept TLS 1.2 or higher
- [ ] I set the Secure, HttpOnly and SameSite=lax attributes in session cookies
- [ ] I set the Secure in all other cookies, and if possible HttpOnly also
- [ ] All 3rd-party JavaScript libraries that my app uses, are updated to the latest version

I have a login feature and cannot use an already existent service:

- [ ] I store the password using a strong cryptographic function (PBKDF2, HMAC-SHA256, bcrypt)
- [ ] I ask for the current password to change the password, email and any other information used in the password reset process
- [ ] I only accept passwords longer than 12 chars and reject common passwords (top 1000)
- [ ] I support multi-factor authentication
- [ ] I limit the number of attempts to endpoints such as login, password reset and 2FA validation
- [ ] I use the language libraries to create and validate JWT tokens
- [ ] I destroy the session server-side and invalidate the matching JWT tokens when the user logs out
- [ ] I destroy the password reset token after it is used and after a pre-defined time