16.1 Sicherheit Eingabe

Der Sinn einer PHP Anwendung liegt meist in einer Webseite, womit es also der Öffentlichkeit zugänglich gemacht wird. Damit ist sie auch unterschiedlicher Gefahren ausgesetzt. Bei Benutzereingaben sollte man immer vom schlimmsten Fall ausgehen!

Benutzereingaben

Jede Eingabemöglichkeit ist zugleich auch ein Sicherheitsrisiko. Bei einer schlampigen Programmierung wird oft auf das Escapen der Eingabe vergessen. Die Folge: Man kann einen Schadcode über ein Input-Feld an das System schicken (PHP-Code, JavaScript, SQL Befehle usw.) Am besten ist es also, eine Benutzereingabe soweit zu beschränken, wie es ihre Notwendigkeit gebietet. Die kritischsten Zeichen sind:

Zeichen	Beschreibung	Kritisch für	HTML Entity
"	Anführungszeichen oben	HTML, JavaScript, PHP	`&qout;`
'	einfaches Anführungszeichen	HTML, JavaScript, PHP, MySQL	`&apos;`
&	kaufmännisches UND	HTML – HTML Entities	`&`
<	öffnende spitze Klammer	HTML (Einleitung für <script> bzw. <?php ...)	`<`
>	schließende spitze Klammer	HTML (Abschluss von </script) bzw. ?>)	`>`
;	Semikolon	HTML, JavaScript, PHP, MySQL	`&semi;`
\	Backslash	Entwerten von Zeichen	`&bsol;`

Die Funktionen `htmlentities()`, `htmlspecialchars()` *und* `html_entity_decode()` *escapen eine Benutzereingabe recht zuverlässig. Wer mit dem Ergebnis der Funktionen nicht ganz zufrieden ist (z. B. weil zu viele Zeichen ersetzt werden), kann einen String auch mit* `str_replace()` *bearbeiten.*

Vorsicht ist auch bei superglobalen Variablen geboten. Insbesondere bei `$_GET` Variablen, weil diese sichtbar für den Benutzer an ein Script übergeben werden. Damit kann ein Benutzer über die URL unerwünschte Effekte im Script auslösen.

Natürlich ist auch der Sinn einer Eingabe stets kritisch zu hinterfragen. Wenn eine Eingabe eine Zahl benötigt, dann muss das Script diese Eingabe auch als Zahl behandeln. Eine Eingabe eines Strings kann sonst zu einem Fehler führen. Benötigt man also eine Ganzzahl kann die Eingabe mit `is_int()` geprüft und|oder mit `intval()` konvertiert werden. Achtung: Eine Division durch Null erzeugt immer einen Fehler der mit einer einfachen Verzweigung verhindert werden kann.

Einiges kann schon im HTML Formular verhindert werden, z. B. über die Attribute `type` oder `required`. Dennoch ist eine zusätzliche `if(isset(..)) {…}` Fallunterscheidung immer zu empfehlen.