

WordPress 2.6.5 is available! Please update now.

19:07:37 18.05.2012

Gestern Nachmittag hatte ich bei Securityfocus erst die Anzahl von Vulnerabilities zwischen Typo3 und Wordpress verglichen. Typo3 kam auf zwei Seite, Wordpress auf drei. Wobei viele Meldungen auf Typo3 Extensions entfallen. Wordpress hat diesen Bonus nicht.

Bevor ich jetzt jedoch die ganze Update Prozedur starte, habe ich mir die Unterschiede zwischen Wordpress 2.6.3 und Wordpress 2.6.5 angeschaut. Die Wordpress Version 2.6.4 wurde übersprungen, da es wohl eine Fake Version gab.

Das Update würde insgesamt nur 5 Dateien betreffen. Dafür den Aufwand treiben, die Datenbank zu sichern, alles löschen, das ganze Wordpress Paket einspielen, ist mir zu aufwendig. Ich werde es händisch machen und mir dabei gleich mal die Fehler anschauen.

In einer Datei wird bloß die Versionsnummer hochgezählt. Manche Blogs behaupten, das Technorati Blogs aus dem Index schmeißt, die eine ältere Wordpressversion verwenden, wie alt diese jedoch sein muß konnte ich bisher nicht herausfinden. Prinzipiell ist dies jedoch für mich ein Schwachstelle, wenn man "von außen" herausfinden kann, auf welcher Version ein System läuft. Dadurch wird es dem Angreifer umso leichter gemacht, den richtigen Hebel auszuwählen um anzusetzen.

Schauen wir uns einen weiteren Fehler an. Dies soll keine Klugscheißerei werden, denn die kann keiner leiden, aber nur aus Fehlern lernt man.

```
[php]
// File: /wp-admin/users.php

129 129  $go_delete = false;
130 130  foreach ( (array) $userids as $id ) {
131      $id = (int) $id;
131 132      $user = new WP_User($id);
132 133      if ( $id == $current_user->ID ) {
[/php]
```

Zeile 5 bzw. 131 ist neu. Bevor die Variable \$id an WP_User übergeben wird, wird sie nach int gecasted. Dies ist auch sinnvoll, denn ursprünglich kommt das Array direkt aus einem Formular und nicht etwa aus der Datenbank.

Was mir jedoch nicht ganz schlüssig ist, warum wird der Cast nicht direkt in der Methode gemacht? Die Methode wird bestimmt noch an mehreren Stellen aufgerufen und man sollte der Methode die Verantwortung übertragen, sicherzustellen, das nur die richtigen Werte übertragen werden.

Betrachten wir noch eine weitere Lücke, die verdeutlichen soll, warum PHP Anwendungen so anfällig sind. In Programmiersprachen wie Java oder JavaScript wäre der Fehler schon von Anfang an aufgefallen.

Sehen wir uns zuerst den alten Code an:

```
[php]
// File: /wp-includes/feed.php

498 echo 'http'
499     . ( $_SERVER['https'] == 'on' ? 's' : '' ) . '://'
```

```
500     . $_SERVER['HTTP_HOST']
501     . wp_specialchars(stripslashes($_SERVER['REQUEST_URI']), 1);
[/php]
```

Und nun die Neue:

```
[php]
// File: /wp-includes/feed.php

498     $host = @parse_url(get_option('home'));
499     $host = $host['host'];
500     echo clean_url(
501         'http'
502         . ( (isset($_SERVER['https']) && $_SERVER['https'] == 'on') ? 's' : '' ) . '://'
503         . $host
504         . stripslashes($_SERVER['REQUEST_URI'])
505         );
[/php]
```

Bevor jetzt also geprüft ob der `$_SERVER['https'] == 'on'` wird erstmal überprüft ob die Variable überhaupt gesetzt ist. Wenn die Variable nicht gesetzt ist, macht es auch keinen Sinn zu überprüfen ob sie on ist. Sollte die erste Bedingung bereits fehlschlagen wird sofort abgebrochen. Dafür sorgt das `&&`.