Curso POO PHP Cross-Site Scripting

Sumario

Cross-Site Scripting (XSS)

Unha páxina ou sitio web vulnerable a un ataque XSS permite ao atacante introducir e executar na páxina o seu propio código en linguaxe de script (xeralmente Javascript).

Por exemplo, se no noso blogue deixamos aos usuarios introducir comentarios ás novas, e non comprobamos o seu contido, algún podería introducir a seguinte cadea de texto como comentario:

```
<script>window.location = "http://www.google.es";</script>
```

Isto non supón un problema á hora de almacenarse na base de datos, como sucedía coas inxeccións SQL. O problema ven cando amosamos esa información en pantalla, por exemplo cada vez que alguén visita a noticia que publicamos no noso blogue. Nese caso publicarase o comentario malicioso, executarase o código javascript e o navegador redireccionará ao usuario a outra páxina.

A solución ao problema é revisar e modificar os contidos de orixe alleo (por exemplo os que proveñen de información almacenada nunha base de datos, ou os introducidos polos usuarios) que se publican no sitio web, principalmente nas seguintes ubicacións:

- Como parte do corpo da páxina web.
- Como valor dun atributo dun elemento HTML.
- Como parte dunha folla de estilos.
- Como parte dunha URL (nos enlaces ou nas redireccións).
- Como parte dun código en linguaxe de script xerado dinámicamente.

Imos ver como protexer o noso código fronte as principais vulnerabilidades XSS.

Contido publicado no corpo dunha páxina web

Neste caso temos que vixiar a presenza no texto de etiquetas HTML como por exemplo "<script>". Temos varias alternativas en función do que procuremos:

- Eliminar completamente as etiquetas HTML presentes no texto. Neste caso podemos empregar a función **strip_tags**.

  ```
  $saida = strip_tags($texto);
  ```

  É posible indicar como parámetro algunhas etiquetas que queremos que se poidan empregar. Por exemplo, poderíamos deixar poñer enlaces como parte dos comentarios dun blogue.

  ```
  $saida = strip_tags($texto, "<a>");
  ```

  É importante suliñar que a función strip_tags non comproba a validez do código HTML que procesa, polo que o texto obtido pode ter etiquetas sin pechar ou mal anidadas.

- Converter as etiquetas presentes no texto ás súas respectivas entidades HTML, para que poidan visualizarse correctamente pero non formen parte da estrutura da páxina. Para isto podemos empregar as funcións **htmlentities** ou **htmlspecialchars**, dependende de se queremos converter a entidades todos os caracteres posibles ou soamente os máis significativos preservando o resto.

  htmlspecialchars traduce en entidades soamente os caracteres **&** (&), **'** ('), **"** ("), **<** (<) e **>** (>).

  ```
  $saida = htmlentities($texto);
  ```

Tamén é posible empregar a libraría **HTML Purifier**, que entre outras melloras aporta a verificación da validez do código HTML e a comprobación dos atributos.

Contido publicado como parte dunha URL

As veces é necesario empregar contido de orixe alleo como parte dunha URL. Por exemplo, se queremos facer unha busca en Google dun termo introducido polo usuario como "PHP", podemos facelo coa URL *"http://www.google.es/search?q=PHP"*. Isto é:

```
$url = "http://www.google.es/search?q=" . $_REQUEST["termo_busca"];
```

Para asegurar que a URL é válida e se axusta aos nosos requerimentos, debemos filtrar a cadea engadida para transformar caracteres como os espazos ou as barras empregando a notación do símbolo de porcentaxe seguido de dous díxitos hexadecimais. En PHP a función **rawurlencode** encárgase da devandita transformación:

```
$url = "http://www.google.es/search?q=" . rawurlencode($_REQUEST["termo_busca"]);
```

Outra función semellante é **urlencode**, na que soamente cambia a transformación dos espazos con respecto á función anterior. En rawurlencode os espazos tranfórmanse en %20, seguindo as normas da RFC3986. En urlencode, que se mantén por razóns históricas e de compatibilidade con sistemas antigos, os espazos tranfórmanse no caracter "+".

Contido pasado como valor a unha variable dun linguaxe de script

En ocasións necesitamos insertar código na páxina web que se envía ao navegador. Por exemplo, se queremos engadir algún tipo de validación no lado cliente antes de enviar un formulario, poderíamos engadir un atributo á etiqueta "<form>".

```
<form onsubmit="return validar(this);">
```

Se imos xerar de forma dinámica parte do código Javascript que se envíe ao cliente, deberemos comprobar previamente a súa seguridade. Por exemplo, cando facemos algo como:

```
<script>x='<?php echo $x; ?>';</script>
```

E a variable $x vale algo como **',window.location = "http://www.google.es",**//, o resultado é a redirección do navegador a outra páxina.

Nestes casos a mellor solución é empregar codificación JSON (JavaScript Object Notation) para o valor da variable, botando mán da función **json_encode**.

```
<script>x='<?php echo json_encode($x); ?>';</script>
```

--Víctor Lourido 12:42 19 jul 2013 (CEST)