O ataque XSS (Cross-Site-Script)

Por: Aureliano Martins Peixoto

Publicado: 17/06/2011 em Goianesia, Hacker, Linux, Sem categoria, Software livre, ubuntu

O ataque XSS (Cross-Site-Script) consiste em enviar um script pelo formulário (seja de cadastro, login, etc) para ser executado pelo servidor. Muitas pessoas utilzam-se dessa técnica para capturar dados de um site, como dados de sessão, senhas, etc. Prevenir esse tipo de ataque é muito simples. Nesse post vou apresentar alguns comandos que são uma 'mão na roda' contra esse tipo de ataque:

htmlentities():

Essa função transforma caracteres especiais (e letras com acento) html em entidades html.

Exemplo:
Ele converte o sinal '<' em '<' e o sinal '>' em '>' . Dessa forma, impede que códio PHP seja executado quando o script interpretar os dados do formulário. Ele também é utilizado para tratar caracteres com acento, já que os converte em suas respectivas entidades html. Exemplo: a palavra 'ação' depois de tratada por essa função é convertida em 'açáo' , permitindo melhor interpretação do navegador. Obs: Para que a conversão de caracteres acentuados ocorra corretamente, use essa função combinada com utf8_decode().

Exemplo de uso:
$comentario = htmlentities(utf8_decode($_POST['comentario']));

Dessa forma, o comentário enviado tem seus caracteres especiais convertidos como nos exemplos acima.

strip_tags()

Problema resolvido! Ainda não.

Se o conteúdo do comentário for um código javascript como esse:

'<script>alert('eu sou o XSS!');</script>'.

Por se tratar de código javascript, pode ser executado pelo navegador do cliente na hora em que for exibido no site. O ideal seria remover todas as tags html do conteúdo, certo? Então vamos utilizar a função strip_tags(). Ele remove todas as tags html do código (removendo assim a tag script), impedindo a execução de códigos como esse. Além disso, remove a formatação feita pelo usuário, que geralmente causa problemas de layout (usuário altera o tamanho da letra usando a tag font, por exemplo).

Vamos então melhorar nosso código:

$comentario = htmlentities(utf8_decode(strip_tags($_POST['comentario'])));

O ideal é executar a função strip_tags() antes de htmlentities(), para que ele seja usado apenas para tratar os acentos.

addslashes():

Essa talvez seja uma das funções mais importantes a ser comentada aqui. Ela coloca uma barra '\' antes dos caracteres especiais, como aspas, apóstrofos, cifrões, etc. Em geral, ataques de injeção de SQL (quando o usuário mal intencionado tenta enviar comandos sql pelo seu formulário para obter informações) são feitos utilizando-se de aspas ou apóstrofos, para manipular o código sql definido no script. Ao filtrar os dados com essa função, as aspas e apóstrofos não serão tratadas como delimitadoras da string, mas como parte dela, evitando assim esse tipo de ataque.

Nosso código agora está assim:

$comentario = addslashes(htmlentities(utf8_decode($_POST['comentario'])));

trim():

Por último e não menos importante, a função trim remove espaços em branco do início e do fim da string. Eu sei, isso não ajuda em nada na segurança do site, mas previne erros na hora da gravação. Espaços em branco são considerados parte da string. E se por acaso você tiver um campo no banco de dados que aceita apenas 30 caracteres e o usuário envia 31 (incluindo os espaços)? O resultado é um erro na hora da gravação. Para evitar esse tipo de dor de cabeça, o melhor que temos a fazer é prevenir esse tipo de situação.

Agora, o código final:
$comentario = addslashes(htmlentities(utf8_decode(trim($_POST['comentario']))));

Extenso, não? Mas previne muita dor de cabeça. Para facilitar sua vida, crie uma função que execute todas as citadas acima:

```
function tratarStrings($string){
return addslashes(htmlentities(utf8_decode(trim($string))));
}
```
Em seguida, basta executá-la onde for necessário:
$comentario = tratarStrings($_POST['comentario']);

Campos de data (que geralmente estão no formato dd/mm/aaaa) podem ser tratados tranquilamente com essa função. Caso a informação ser tratada seja um número inteiro ou de dupla precisão ('quebrado'), pode-se converte-lo da seguinte forma:

$id = (int) $_GET['id'];

$valor = (float) $_POST['valor'];

Dessa forma, os números enviados serão convertidos para os tipos informados entre parênteses. Se a informação enviada for uma string, o resutado será zero.