

# 1 Curso POO PHP Inxección de código PHP

## 1.1 Inxección de código

Esta vulnerabilidade baséase na inserción e execución de código alleo. Dependendo de dónde se execute ese código, podemos distinguir:

### 1.1.1 Inxección de código PHP

Inserción e execución de código PHP alleo na nosa aplicación web. Pode producirse entre outros nos seguintes escenarios:

- Se empregamos *include*, *include\_once*, *require* ou *require\_once* empregando o contido dunha variable como parte do nome do arquivo. Por exemplo:

```
include $arquivo . ".php";
```

Unha forma de restrinxir os efectos da vulnerabilidade é empregar a directiva **allow\_url\_fopen** (ou **allow\_url\_include**) para que non se poidan engadir ficheiros dende ubicacións remotas.

- Se empregamos o constructor da linguaxe **eval**. **eval** permite executar o código contido nunha cadea de texto. Por exemplo:

```
$a = "17";  
eval('$b = $a;');  
echo $b;
```

Cando o valor das variables que empreguemos dentro dun *eval* non estéa controlado, pode resultar na execución no noso sitio de código descoñecido. Por exemplo:

```
$a = $_REQUEST["valor"];  
// Nótese a elección "pouco afortunada" de comiñas dobres  
eval("\$b = $a;");  
echo $b;
```

Por exemplo, cando o valor do parámetro *valor* sexa algo como **1; phpinfo()**; obteremos a execución da función **phpinfo**.

- Se empregamos a función **unserialize** para crear un obxecto a partir dunha representación de orixe non confiable. Por exemplo, cando facemos:

```
$var = unserialize($_REQUEST['representacion']);
```

A variable *\$var* pode pasar a ser un obxecto cun conxunto de propiedades con valores descoñecidos. Os obxectos que se recuperan empregando a función *unserialize* executan o método máxico **\_\_wakeup** (se existe), e tamén o método **\_\_destruct** cando se destrúan (tamén se existe). Estes métodos poden empregarse para levar a cabo accións non desexadas.

Por exemplo, se temos un obxecto que emprega ficheiros temporais e os borra cando se destrúe, un atacante podería crear un obxecto tal que cando se destrúe borrara outro ficheiro calquera do sistema.

```
class inxeccion  
{  
    public $fich_tmp;  
    ...  
  
    function __destruct()  
    {  
        if (file_exists($fich_tmp)) @unlink($fich_tmp);  
    }  
}
```

Soamente sería necesario pasar un obxecto serializado da clase "inxeccion" con *\$fich\_tmp* apuntando ao ficheiro que desexemos eliminar.

- Se empregamos a función **preg\_replace** co parámetro **e** no patrón para realizar substitucións en cadeas empregando expresións regulares. O parámetro **e** indica que o segundo parámetro debe avaliarse como unha expresión PHP.

## 1.1.2 Inxección de código no sistema operativo

Inserción e execución de código alleo que se executa sobre o sistema operativo do servidor. Prodúcese principalmente:

- Cando empregamos unha función como **system**, **exec** ou calquera **función que execute algún comando no sistema operativo**, incluíndo o **operador de execución (`)**, e empregamos na chamada o valor dunha variable non controlada, podemos obter como resultado a execución dun comando alleo no sistema operativo.

Por exemplo, se quixeramos listar os ficheiros con unha terminación determinada que se atopan no sistema de ficheiros local, poderíamos facer:

```
$listado = system('ls *.*' . $_REQUEST['terminacion']);  
echo $listado;
```

Pero se no parámetro "terminacion", no canto dunha extensión como "php" obtemos algo como **"php; rm -r \*\*"**, executarase tamén o segundo comando cos privilexios de execución de apache (ou do servidor web que execute PHP).

Para evitalo pódense empregar as funcións **escapeshellarg**, pensada para sanear os argumentos pasados aos intérpretes de comandos, ou **escapeshellcmd**, que sanea unha cadea que vai ser empregada como comando (argumentos incluídos).

```
$argumento = escapeshellarg('*.*' . $_REQUEST['terminacion']);  
$listado = system('ls ' . $argumento);  
echo $listado;
```

--V́ctor Lourido 20:58 20 jul 2013 (CEST)