Lenguaje PHP

VULNERABILIDADES

10 Tips para reducir vulnerabilidades en sitios basados en PHP

1) Parámetros no Validados
2) Control de Acceso Inseguro
3) Administración de Sesiones y Cuentas de Usuario Insegura
4) Vulnerabilidades de Cross-Site Scripting (XSS)
5) Buffer Overflow
6) Vulnerabilidades de Inyección de Comandos
7) Problemas en el Manejo de Errores
8) Uso Inseguro de la Criptografía
9) Vulnerabilidades de Administración Remota
10) Mala Configuración del Servidor

1 Parámetros no Validados

Muy Importante !!!!, configure en off el valor de *register_globals* en su *php.ini*. Si no realizamos esto, PHP por defecto setea una variable global con el mismo nombre que cualquier parámetro que se pase por *GET* o *POST*.

Este valor esta configurado en off por defecto en las versiones de PHP 4.2.0 o posteriores lo que nos obliga acceder a valores de URLs, *cookies* y formularios a través de las supervariables *$_GET*, *$_POST* y *$_COOKIE*

Antes de utilizar cualquier valor recibido en una de las supervariables mencionadas, valídelo !, recibir un valor invalido en un supervariable puede ocasionar una caída en el sistema y/o la exposición de de datos potencialmente útiles a un usuario malintencionado.

Si se esta esperando un valor determinado en una *cookie* o en un campo tipo *hidden* para verificar la legitimidad de una petición (enviado previamente por mi aplicación), nunca lo utilice en forma individual, acompáñelo de un *hash* que impida ser modificado.

2 Control de Acceso

En vez de tratar de desarrollar su propio sistema de control de acceso, utilice los módulos *PEAR* o utilice sistemas de login de terceros como Facebook o google.

Muchas veces al tratar de desarrollar métodos de autenticación y control de acceso propios, cometemos errores que exponen toda nuestra aplicación.

3 Administración de Sesiones y Cuentas de Usuario Insegura

Use las funciones integradas de PHP para el manejo de sesiones para un manejo seguro y estandarizado de sesiones. Una vez más, el pretender desarrollar todo muchas veces nos lleva a cometer errores que exponen nuestra aplicación

De todas maneras, sea muy cauteloso respecto de como esta configurado el manejo de sesiones en el servidor con el que opero y como se almacena la información de sesiones, por ejemplo, si la información de las sesiones se almacenan en un directorio con permisos globales de lectura como */tmp*, cualquier usuario que consiga loguearse en el servidor, sin importar su nivel de acceso podrá acceder a la inflamación mencionada. Siempre almacene la información crítica en directorios con acceso restringido.

3 Administración de Sesiones y Cuentas de Usuario Insegura

Configure el entorno de producción de manera de mantener seguro el manejo de sesiones, esto se realiza seteando las siguiente variables:

```
session.cookie_lifetime=0
session.use_cookies=on
session.use_only_cookies=on
session.use_strict_mode=on
session.cookie_httponly=on
session.cookie_secure=on
session.cache_limiter=nocache
session_gc()
session.hash_bits_per_character=“4”
session.hash_function=“sha256”
```

3 Administración de Sesiones y Cuentas de Usuario Insegura

```
 <?
//si es necesario cambiar la config. del php.ini desde tu script
ini_set("session.use_only_cookies","1");
ini_set("session.use_trans_sid","0");

//iniciamos la sesión
session_name("loginUsuario");
session_start();
session_set_cookie_params(0, "/", $HTTP_SERVER_VARS["HTTP_HOST"], 0);
//cambiamos la duración a la cookie de la sesión
//antes de hacer los cálculos, compruebo que el usuario está logueado
//utilizamos el mismo script que antes
if ($_SESSION["autentificado"] != "SI") {
    //si no está logueado lo envío a la página de autentificación
    header("Location: index.php");
} else {
    //sino, calculamos el tiempo transcurrido
    $fechaGuardada = $_SESSION["ultimoAcceso"];
    $ahora = date("Y-n-j H:i:s");
    $tiempo_transcurrido = (strtotime($ahora)-strtotime($fechaGuardada));

    //comparamos el tiempo transcurrido
     if($tiempo_transcurrido >= 600) {
     //si pasaron 10 minutos o más
      session_destroy(); // destruyo la sesión
      header("Location: index.php"); //envío al usuario a la pag. de autenticación
      //sino, actualizo la fecha de la sesión
    }else {
    $_SESSION["ultimoAcceso"] = $ahora;
   }
}
?>
```

4 Vulnerabilidades de Cross-Site Scripting (XSS)

Nunca muestre en su aplicación información que provenga del exterior de dicha aplicación sin validarla previamente, recuerde que Ud. siempre muestra código HTML, por lo tanto si un usuario malicioso sube código HTML espurio y Ud. lo muestra sin filtrar, otro usuario puede verse afectado.

PHP nos da una multitud de métodos para filtrar la información no confiable, por ejemplo:

- *htmlspecialchars()* convierte los caracteres & > " < en su codificación HTML correspondiente, asimismo puede cambiar a comillas simples pasando ENT_QUOTES como segundo argumento.

- *strtr()* filtra cualquier caracter que desee. Pase a *strtr()* un array con los reemplazos que desee realizar. Para cambiar (y) en su codificación HTML correspondiente, lo cual es recomendado para prevenir ataques XSS realice: *$safer = strtr($untrusted, array('(' => '(', ')' => ')'));*

- *strip_tags()* remueve cualquier tag HTML o PHP de un string.

- *utf8_decode()* Convierte los caracteres ISO-8859-1 en un *string* codificada en Unicode UTF-8 a un carácter ASCII de un byte. Los ataques XSS se suelen esconder codificando en UTF-8

5 Buffer Overflow

No se puede alocar memoria en tiempo de ejecución en PHP por lo que no debería existir riesgo de producir un *Buffer Overflow*, lo que no podemos evitar son lo errores de codificación propios del núcleo del código PHP por lo que se debe mantener actualizado el motor de PHP a su última versión y *service pack*.

6 Vulnerabilidades de Inyección de Comandos

Un XSS ocurre cuando Ud. muestra datos no filtrados en el browser del usuario, el que queda expuesto es el browser del usuario, puesto que normalmente lo que se realiza es la descarga de código .js malicioso. La inyección de comandos ocurre cuando Ud. pasa datos no validados y no filtrados a un proceso externo o una base de datos. Para evitarlo, realice los mismos procedimientos de validación de datos en cualquier comando que se pase a una base de datos o proceso externo.

Trate de evitar ejecutar comandos externos al PHP, si ello no es posible utilice las funciones *escapeshellcmd()* y *escapeshellarg()* antes que *exec()* o *system().*

Antes de ejecutar cualquier comando externo, canonicalice el path del mismo con *realpath()* y verifique adicionalmente que el path utilizado este en un lugar lógico (por ejemplo bajo el *webroot*).

Si Ud esta pasado datos ingresados por un usuario directamente a un query SQL, escape los datos con addslashes() antes de poenrlos en el query. Si Ud. está usando MySQL, escape las cadenas con mysql_real_escape_string() o con mysql_escape_string().

6 Vulnerabilidades de Inyección de Comandos

Siempre que sea posible evite pasar directamente datos desde formulario a una instrucción SQL, reemplazando la construcción del sql por instrucciones del tipo `mysqli_prepare`

7 Problemas en el Manejo de Errores

Si un usuario malicioso recibe los mensajes de error del sistema (PHP, base de datos o archivos externos) de forma directa, sin filtrar, puede hacerse una idea de la organización del sistema y a través de ello intentar encontrar vulnerabilidades conocidas o errores documentados en las versiones de software usado.

Los mensajes de error no deberían contener ninguna información útil acerca de las versiones del software usado ni descripciones del funcionamiento del propio sistema.

Una forma de evitar esto es indicando al PHP que coloque todos los errores en los log de errores del servidor, esto se logra serrando las siguientes directivas:

8 Uso inseguro de la criptografía

La extensión *mcrypt* provee una interfase estandarizada para muchos algoritmos de encriptación. Trate de utilizar esta extensión, probada y depurada por muchas fuentes, antes de desarrollar su propio sistema de encriptación. Asimismo sea muy cuidadoso del lugar donde almacena sus claves y sus llaves de encriptación. Recuerde que el algoritmo de encriptación mas fuerte no sirve de nada si un hacker obtiene las claves. Siempre almacene sus claves en lugares separados de la información encriptada.

9 Vulnerabilidades de Administración Remota

Siempre que sea posible ejecute las herramientas de administración remota sobre medios encriptados, ya sea ssh o SSL para evitar sniffeos de passwords. Si utiliza softwares comerciales, cambie todos los usuarios y passwords por defecto del software. Asimismo cambie los URL administrativos por defecto siempre que sea posible.

10 Mala Configuración del Web Server o del Application Server

Mantenga su PHP actualizado y este al tanto de los problemas de seguridad que se anuncien por parte del fabricante o de los mailing-list disponibles. Utilice archivos de configuración de ejemplo disponibles en los sitios oficiales de PHP