

PROGRAMACIÓN SEGURA EN PHP

Héctor A. Mantellini
@VaSlíbre

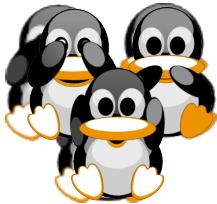


CODIGO
SEGURO



CODIGO
LIMPIO

@xombra



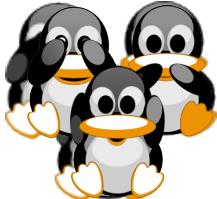
PROGRAMACIÓN SEGURA EN PHP

NINGUNA APLICACION
ES
100% SEGURA



WTF!!!



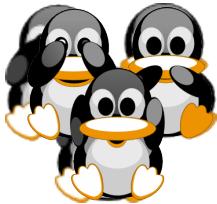


PROGRAMACIÓN SEGURA EN PHP

La programación segura de **PHP** o cualquier otro lenguaje, es esencial para no comprometer la seguridad del servidor donde este alojada nuestra aplicación web.

Una aplicación con mal diseño de seguridad es vulnerable usualmente a:

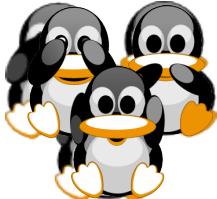




PROGRAMACIÓN SEGURA EN PHP

XSS: Cross-site scripting es un tipo agujero de seguridad basado en la explotación de validación de HTML incrustado. Esta falla suele producirse por varias razones : *Variables no inicializadas correctamente, Ausencia de control de datos, entre otras.*



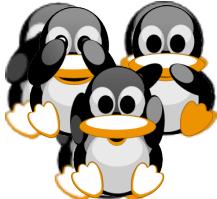


PROGRAMACIÓN SEGURA EN PHP

CSRF (XSRF): *Cross-site request forgery* o falsificación de petición en sitios cruzados, es un tipo de exploit malicioso de un sitio web en el que comandos no autorizados son transmitidos por un usuario en el sitio web que confía.

Es un enlace hostil, ataque de un click, cabalgamiento de sesión, y ataque automático.



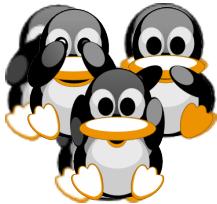


PROGRAMACIÓN SEGURA EN PHP

SQL-injection: El origen es el filtrado incorrecto de las variables utilizadas en las partes del programa con código SQL.

Esta falla suele producirse por razones como: *Ausencia de control de datos.*



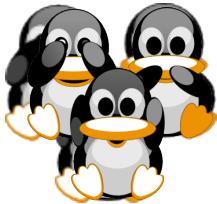


PROGRAMACIÓN SEGURA EN PHP

RFI: Este tipo de ataque es relativamente desconocido entre los desarrolladores, lo que lo hace especialmente dañino.

La inclusión de archivo remoto, explora una aplicación vulnerable e inyecta código malicioso.





PROGRAMACIÓN SEGURA EN PHP

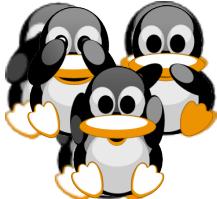
CODIGO SEGURO

Para tener un código seguro hay que tener en cuenta básicamente 2 cosas:



<http://www.vaslibre.org.ve>





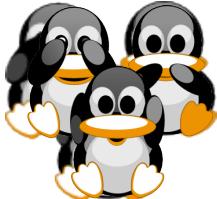
PROGRAMACIÓN SEGURA EN PHP

Sanitizar y Validar



Sanitizar y Validar los datos, son los puntos de más importancia al programar una aplicación segura, y las que más impacto tienen a la hora de analizar la seguridad, debido que estos datos pueden ser manipulados por cualquier usuario, en muchos casos malintencionados.





PROGRAMACIÓN SEGURA EN PHP

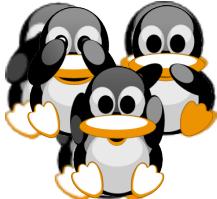
Que es Sanitizar?

Es el proceso de aplicar una limpieza exhaustiva a un dato o grupos de datos para su uso. Por ejemplo, **FILTER_SANITIZE_EMAIL** quita caracteres no apropiados que contiene una dirección email.



<http://www.vaslibre.org.ve>



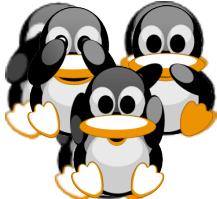


PROGRAMACIÓN SEGURA EN PHP

Que es Validar?

Es el proceso utilizado para validar o comprobar si los datos cumplen con ciertos requisitos predefinidos. Por ejemplo, **FILTER_VALIDATE_EMAIL** determinar si los datos contienen una dirección válida de correo electrónico, pero sin cambiar los datos en sí.





PROGRAMACIÓN SEGURA EN PHP

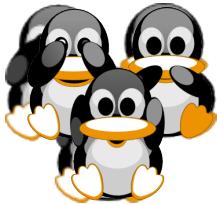


16 Normas
para un
código 99.99% seguro



<http://www.vaslibre.org.ve>





PROGRAMACIÓN SEGURA EN PHP

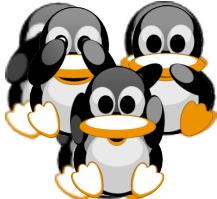


00.- Jamás confiar en los usuarios.



<http://www.vaslibre.org.ve>



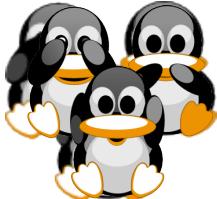


PROGRAMACIÓN SEGURA EN PHP



01.- Siempre hay que validar los datos que están bajo el control de los usuarios. **Nunca** se debe pasar al sistema ningún dato que provenga de una entrada de usuario sin ser validada antes.





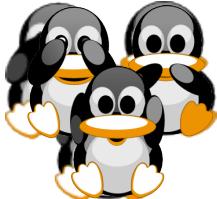
PROGRAMACIÓN SEGURA EN PHP



02.- Usar siempre `$_POST` - `$_GET` - `$_COOKIE` - `$_FILE` para capturar datos que provengan de otras páginas. Verificar que `register_globals` este en OFF .

No usar `$_REQUEST`.



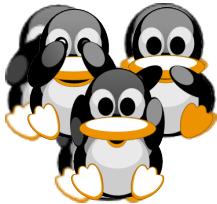


PROGRAMACIÓN SEGURA EN PHP



03.- No usar solamente Javascript para validar datos. Javascript **no ofrece una seguridad real**. Siempre validen todos sus datos de lado del servidor.





PROGRAMACIÓN SEGURA EN PHP

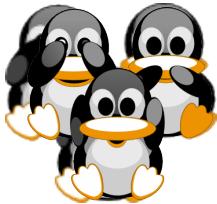


04.- Definir siempre las variables antes de usarlas.



<http://www.vaslibre.org.ve>



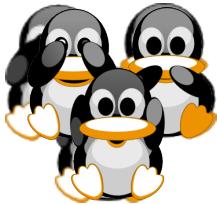


PROGRAMACIÓN SEGURA EN PHP



05.- Evitar el uso de permisos 777, para archivos y carpetas. Para casos especiales use archivos .htaccess para colocar los permisos necesarios.





PROGRAMACIÓN SEGURA EN PHP

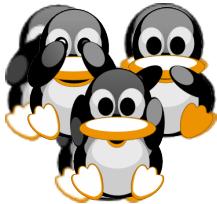


06.- Almacenar las claves en formato **md5()** o **sha1()** o cualquier otro tipo de cifrado. Si es combinado mejor.



<http://www.vaslibre.org.ve>



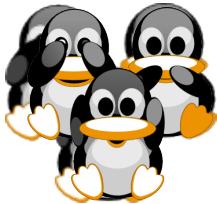


PROGRAMACIÓN SEGURA EN PHP



07.- Si necesita usar tener archivos .inc
(haciendo referencia a includes) o .ini/.cfg
coloque al final .php ejemplo: config.ini.php





PROGRAMACIÓN SEGURA EN PHP

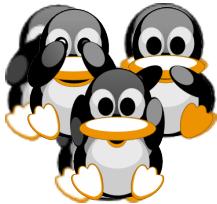


08.- Utiliza el `==` para verificar valores de entrada (*identidad de datos y de tipos*).



<http://www.vaslibre.org.ve>



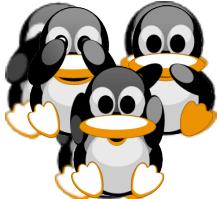


PROGRAMACIÓN SEGURA EN PHP

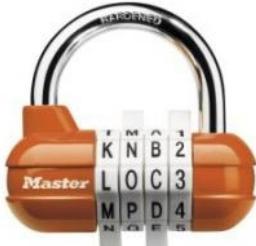


09.- Limitar las entradas solamente a los tipos de datos que deba recibir.





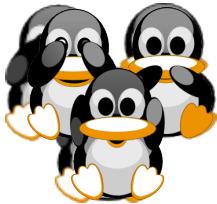
PROGRAMACIÓN SEGURA EN PHP



10.- Para evitar el **RFI** verificar las directivas:
allow_url_fopen y *allow_url_include* en el *php.ini*

La directiva **allow_url_fopen** activada, y
allow_url_include desactivada. Se puede usar
archivo **.htaccess**



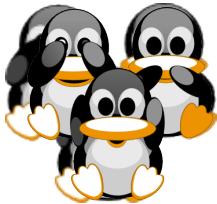


PROGRAMACIÓN SEGURA EN PHP



11.- Deshabilitar mostrar errores de ejecución o código. Solo mostrar los errores en fase de desarrollo, jamás en producción.





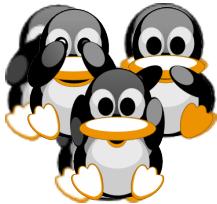
PROGRAMACIÓN SEGURA EN PHP



12.- Verificar las sentencias SQL (MySQL) -
mysql_real_escape_string() Escapa ciertos
caracteres de una cadena en una sentencia SQL.

Esta función debe ser cambiada por
mysqli_real_escape_string()



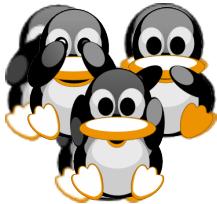


PROGRAMACIÓN SEGURA EN PHP



13.- Evitar el uso el comodín * en las sentencias SQL





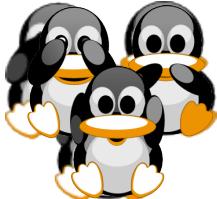
PROGRAMACIÓN SEGURA EN PHP



14.- Evitar uso de Cookies, es preferible usar
SESSION

Cookies conlleva (*inseguridad, límite de tamaño de cabeceras de peticiones, etc*). Con sesiones solo se debe tener cuidado de no perder el ID de session.





PROGRAMACIÓN SEGURA EN PHP

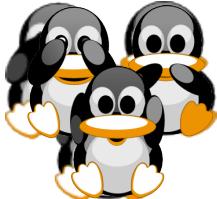


15.- Usar urlencode y urldecode
Codifica/Decodifica como URL una cadena.



Ninguna prevención es inútil.





PROGRAMACIÓN SEGURA EN PHP



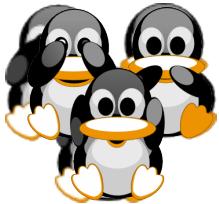
**PROTEGE TUS
TRABAJOS
USA LICENCIAS
LIBRES**

<http://www.safecreative.org>

@SafeCreativeLA

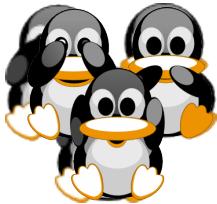
<http://www.vaslibre.org.ve>





PROGRAMACIÓN SEGURA EN PHP

PREGUNTAS



PROGRAMACIÓN SEGURA EN PHP

Héctor A. Mantellini

Twitter: [@xombra](https://twitter.com/@xombra)



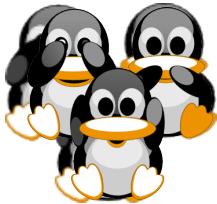
<http://www.xombra.com>



[@awvene](https://twitter.com/@awvene)

[@vaslibre](https://twitter.com/@vaslibre)





PROGRAMACIÓN SEGURA EN PHP

Herramientas y/o enlaces recomendados:

Getting Clean With PHP:

<http://net.tutsplus.com/tutorials/php/getting-clean-with-php>

Writing Secure PHP: <http://www.addedbytes.com/writing-secure-php>

PhpSecInfo: <http://phpsec.org/projects/phpsecinfo/index.html>

PHP Security Scanner: <http://sourceforge.net/projects/securityscanner>

Ratproxy: <http://code.google.com/p/ratproxy/>

Inspekt: <http://code.google.com/p/inspekt/>

Protección contra las técnicas de Blind SQL Injection:

<http://www.elladodelmal.com/2007/07/proteccin-contra-las-tcnicas-de-blind.html>

Pixy: XSS and SQLI Scanner for PHP Programs:

<http://pixybox.seclab.tuwien.ac.at/pixy/index.php>

PHP-IDS: <http://php-ids.org/>

Proxmon: <http://www.isecpartners.com/proxmon.html>

