Un CAPTCHA fácil y sencillo de implementar.

CAPTCHA

Introducción:

Siempre me ha llamado la atención los CAPTCHA ("*Completely Automated Public* Turing test *to tell Computers and Humans Apart*") que hay en muchas páginas web y que permiten diferenciar consultas hechas por humanos de las hechas por "arañas" que pululan por internet (son también conocidas como "robots"). La diferencia esencial en este caso con respecto a la prueba de Turing es que se trata de convencer a una máquina que es un humano, osea el inverso de la prueba.

En el idioma inglés existe la filosofía ***K.I.S.S.*** y la practico frecuentemente ya que los sistemas complejos tienden al caos de manera natural. Es por ello que buscando por internet (precisamente ayudado por las "arañas" que pretendo combatir, ¡qué ironía!) he encontrado el siguiente artículo que me deslumbró por su simplicidad más al cabo de una hora de estudio observo los elementos implícitos sin los cuales NO funciona. **Recomiendo vayan y lean dicho artículo y regresen cuando lo hayan asimilado, yo por aquí les espero. 8-)**

El artículo data del año 2007 pero considero que no ha perdido vigencia alguna, si administran una página web bancaria o son partidarios del último grito de la moda, recomiendo desistan de seguir leyendo esto. ;-)

Pues bien manos a la obra.

Requisitos previos.

Utilizo una máquina virtual con Debian Wheezy con la cual estudio el programa para base de datos PostgreSQL a la cual le he instalado también un servidor Apache con PHP: este tipo de configuración es llamado LAMP por sus siglas en inglés, sólo que en aquí en realidad es un LAPP: Linux, Apache, PostgreSQL y PHP. Hay muchísimos tutoriales para instalar este tipo de servidores por ello partimos de la base que ustedes cuentan con uno en funcionamiento y para pruebas (*no utilicen un servidor "en producción" hasta tanto no estén completamente seguros de este código*). **Importante recalcar que no utilizaremos funciones de bases de datos en esta entrada,** pero muy frecuentemente las CAPTCHA se utilizan en este tipo de ambiente, por ejemplo, registrar usuarios en una lista de correo electrónico (tremenda tentación para los "*robots spam"*), así que allí encaja perfectamente.

Configuración adicional.

Al servidor LAPP anteriormente descrito debemos instalarle unas librerías para el trabajo gráfico bajo el lenguaje PHP. Aquí está muy bien descrito a lo que me refiero, haciendo la salvedad que dichas librerías NO SIEMPRE están instaladas por defecto.

En todo caso debemos crear una carpeta bajo la raíz web del servidor Apache con el nombre "captcha" y donde alojaremos el siguiente archivo "info.php":

y luego iremos a nuestro navegador web preferido a la dirección IP que le hayamos asignado en nuestra red de área local. En mi caso dicha dirección es 192.168.1.27 en la carpeta "captcha":

En dicha página debemos buscar si están instalados los siguientes valores:

gd

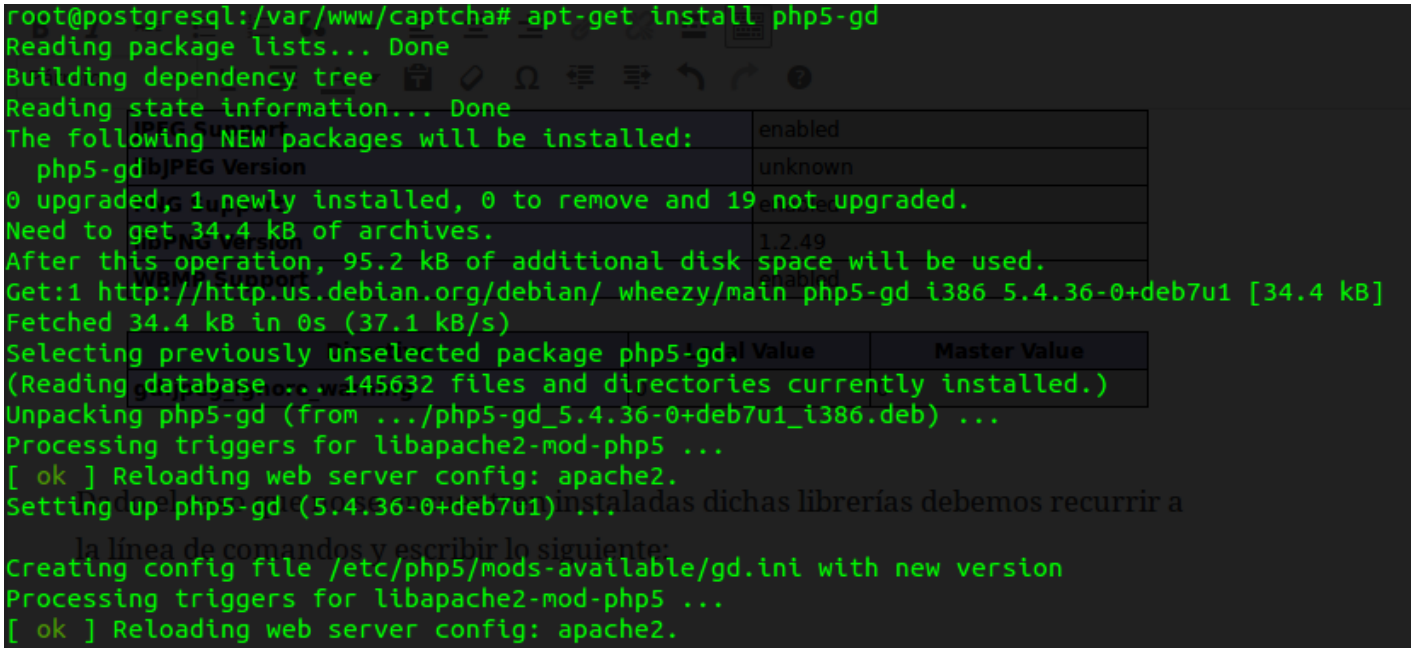
GD Support	enabled
GD Version	2.0.36
FreeType Support	enabled
FreeType Linkage	with freetype
FreeType Version	2.4.9
GIF Read Support	enabled
GIF Create Support	enabled
JPEG Support	enabled
libJPEG Version	unknown
PNG Support	enabled
libPNG Version	1.2.49
WBMP Support	enabled

Directive	Local Value	Master Value
gd.jpeg_ignore_warning	0	0

Dado el caso que no se encuentren instaladas dichas librerías debemos recurrir a la línea de comandos y escribir lo siguiente:

```
sudo apt-get install php5-gd
```

y verán algo más o menos parecido a esto (en mi caso no necesité el comando "*sudo*" porque ya había ganado acceso como *root* con el comando "*su*"):



Dichas librerías son necesarias para el trabajo con imágenes jpg, png y gif, entre otros formatos y funciones.

Imagen de fondo para el CAPTCHA:

Necesitaremos un imagen tipo gif con unas dimensiones de 100x30 píxeles con colores adecuados teniendo en cuenta que las letras serán negras buscaremos colores como azul o verde o el que gusten **pero no de color negro.** Yo elegí la siguiente imagen (aunque al final agregaremos varios elementos para dificutarles la tarea de lectura a los "robots" con técnicas OCR):

Como ven es una imagen con degradado que busca confundir al OCR pero con legibilidad al ser humano; no obstante usar este mismo fondo siempre y con la ayuda de la "fuerza bruta" se puede descifrar al poco tiempo -pero de eso nos encargaremos luego, de "complicarlo" para tratar de eludir la lectura-. La imagen la guardaremos en la carpeta de trabajo en el servidor web bajo el nombre "bgcaptcha.gif" (nemotécnico 'BackGroundCaptcha.gif'='bgcaptcha.gif').

Para bajar archivos por líneas de comando

recomiendo usar "wget", por ejemplo:

"wget http://www.e.com/imagen_a_bajar.gif nombre_imagen.gif"

Creación del archivo "captcha.php".

Dicho archivo se encargará de tomar la imagen anterior y, aleatoriamente, "escribirle" letras encimas, almacenarlas en una variable (que leeremos con luego con el método POST) y mostrarla al navegador; para ello utilizaremos el siguiente código que explicaremos línea por línea:

- Para que el servidor sepa dónde comienza y dónde termina el lenguaje PHP debemos encerrarlo todo con los siguientes comandos: **""** y donde van los paréntesis y puntos suspensivos ubicaremos nuestro código a ejecutar. Todo lo que esté fuera de estos demarcadores será considerado lenguaje HTML e interpretado como tal.
- Cada final de sentencia le será indicado al servidor con un punto y coma **";"**.
- Los comentarios, que son importantísimos para nosotros los seres humanos, para las máquinas carecen de importancia y se identifican con **"*//*"** y la derecha el texto explicativo.
- La función **"**session_start()**"** permite *crear* o *recuperar* un identificador único que servirá para que el servidor pueda atender varios clientes al mismo tiempo sin confundir las respuestas de cada usuario. No debemos preocuparnos mucho ya que todo es automatizado, si acaso dedicaremos una sentencia **"if-then-else"** *por si acaso devuelve el valor "falso", es decir, no se pudo iniciar sesión (todo el mundo da por sentado que devuelve "verdadero").* ***Importante, muy importante, el comprender sobre el*** cómo PHP compara dos variables y/o valores ***para devolver "verdadero" o "falso", merece su estudio de 10 minutos.***

- La función **"$_SESSION[]"** permite asignar un valor a una variable en la sesión iniciada en el punto anterior y es un variable de arreglo global. El texto, que será aleatorio, lo proporcionará la función **"randomText()"** que escribiremos luego.
- La función **"imagecreatefromgif()"** nos permitirá "cargar en memoria" la imagen que destinamos como fondo del CAPTCHA y devolverá un "identificador de imagen" representado en una variable que usaremos para agregar el texto que identificará e

introducirá el usuario, ser humano. De nuevo digo que deberíamos destinar un **"if-then-else"** dado el caso la función devuelva "falso".

- El comando anterior nos permitió establecer el "lienzo" donde vamos a escribir las letras aleatorias; pues el comando **"imagecolorallocate()"** nos permite fijar el color con que las "pintaremos": "0, 0, 0" corresponde al color negro en la codificación de valores "RGB", "Red Green Blue" y cuyos valores van del 0 al 255 cada uno y nos permite usar +16 millones de colores. Luego echaremos mano de esta función para confundir aún más a los "robots", por ahora nos conformaremos con el color negro. Es de hacer notar que para esta función CERO es "falso" y cualquier otro valor es "verdadero", si usted considera esto una tontería le invito a leer la disertación sobre el tema, está avisado o avisada.
- La funcíon **"imagestring()"** dibuja una cadena de texto en nuestro "lienzo", la imagen gif seleccionada. Los parámetros de esta función son: (*image, font, x, y, text, color)* y los detallo a continuación:
    1. *image:* la que cargamos en memoria con la función **imagecreatefromgif()** y que llamamos **$captcha.**
    2. *font:* numeradas del 1 al 5 y es la fuente nativa predeterminada en la librería GD e incluso nos permite cargar nuestras propias fuentes ***pero debemos cargarlas y compilarlas de acuerdo a la arquitectura de nuestro servidor***. Si se entusiasman a realizar esto último deberán cargar dicha fuente "compilada" con la función **imageloadfont()** de acuerdo a unos valores binarios. Más interesante hallo utilizar la función **imagettftext()** NO SIN ANTES VERIFICAR el **phpinfo()** devuelva que el ambiente del servidor lo soporte (si quieren saber más sobre fuentes True Type en GD visiten este enlace):

gd

GD Support	enabled
GD Version	2.0.36
FreeType Support	enabled
FreeType Linkage	with freetype
FreeType Version	2.4.9

    3. *Coordenadas X e Y:* tomadas a partir de la esquina superior izquierda, pónganse de cabeza para que las entiendan (ahhh me recuerdo de la materia Geometría Analítica, ¡qué belleza para Autocad! *¡Y dibujábamos por comandos escritos en papel fuera del laboratorio de computación!*).
    4. *text:* en este caso lo que ya tenemos almacenado en la variable global de sesión "**$_SESSION['tmptxt']".**
    5. *color:* el que establecimos a negro con la función **"imagecolorallocate()".**
- La función **"header()"** permite que nuestro servidor se ciña a las normas del lenguaje HTML que consiste en "notificarle" al navegador web (en formato NO html) que le será enviado un flujo de datos, por defecto "application/octet-stream", pero que nosotros

utilizaremos para indicarle que es una imagen gif "Content-type: image/gif". Si desean conocer más acerca del nacimiento de la web y sus normas de funcionamiento de la mano del mismísimo Tim Berners-Lee hagan click en este enlace.

- Por último la función **"imagegif()"** instruye a nuestro servidor que le envie la imagen al navegador del usuario, el CAPTCHA que queremos interprete el ser humano.
- Una función que considero importante y que yo agrego al código mostrado originalmente -y del que desconozco la autoría- es "imagedestroy()" a fin de liberar la memoria utilizada por la variable "$captcha". Aunque toda la memoria se libera cuando el usuario cierra su navegador o cuando nosotros mismos invocamos "session_destroy()" nunca está demás liberar "trabajo" apenas sea posible.

Creación de la función "randomText()".

Ahora explicaré la función que devuelve una cadena de texto de manera aleatoria pero con dos mejoras al código fuente original, el cual es el siguiente:

```
function randomText($length) { $pattern = "23456789abcdefghijkmnpqrstuvwxyz"; for($i=0;$i
```