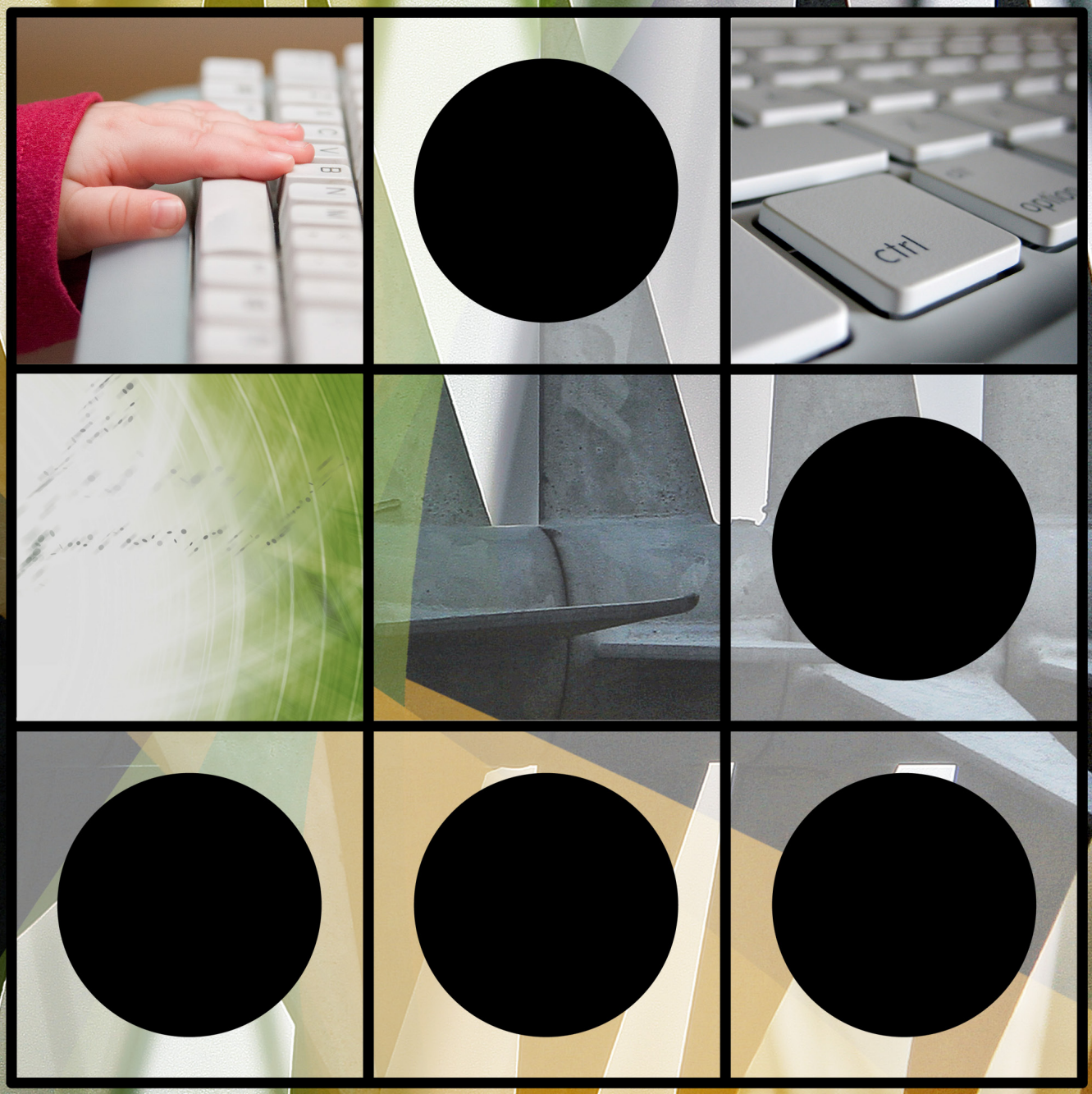


#5

hackhispano

electronic fanzine

Ezine #5

Bienvenidos una vez mas amigos a una nueva entrega de la HH-ezine. Algunos ya sois viejos conocidos y otros nuevos lectores, así que bienvenidos a todos por igual.

Esta ocasión contamos con nuevos retos para desafiar vuestro ingenio y nuevos artículos aunque el verano, las vacaciones y los cuba-libres han hecho que algunos redactores se planteasen escribir en la próxima en lugar de esta. Aun así, tenemos una serie de interesantes texto ilustrativos.

Nunca se puede valorar en revistas de cierta envergadura la calidad en comparación de unos artículos y de otros, pero si que se de uno que os va a gustar. Es la Entrevista a un old-hacker, cuyos autores si que se conocen , pero prefieren el mas absoluto anonimato, eso si os digo queridos amigos... "Internet tiene memoria"

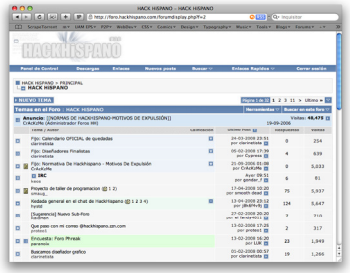
Aprovecho la ocasión para recordaros dos tips acerca del foro. El primero es que seguimos buscando colaborador en el Área de Intrusión - Hacking Ético, y que podéis encontrar más información al respecto en el foro. Por otra parte, también emplazaros a que sigáis pendientes al taller de programación, otro ambicioso proyecto del equipo de HackHispano, y que se creará en breve.

Ya sabéis, un saludo y
Hack the World...

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.



Indice:

- TS Seguridad en accesos remotos
- Introducción frugal a VBScript
- Entrevista a un old hacker
- Introducción a la API de Windows: L/E de la memoria principal
- Moviles perdidos
- Iniciación a radio enlaces de alta frecuencia
- Retos

T.S. seguridad en accesos remotos

Los riesgos de implantar uno o más servidores de ts a traves de internet son muchos más de los que se imaginan. Pero, ¿vamos a privarnos de utilizar estos servicios por la sospecha de que algún listillo con adsl, o algún maldenominado "hacker" intente conseguir root en nuestros sistemas?

En este pequeño escrito trataré de introduciros en la implantación de seguridad para un servidor de aplicaciones sobre acceso remoto.
Al estar centralizados todos los servicios y aplicaciones a través de nuestro servidor TS, nos ahorraremos todo tipo de trabajos como actualización de aplicaciones en los clientes, centralizaremos nuestras copias de seguridad y permitiremos a nuestros usuarios utilizar cualquier cliente de red, ya que una vez que se autentica con sus credenciales en el sistema, se lanza la o las aplicaciones con sus permisos, y hasta le podemos ofrecer un escritorio personalizado con las políticas de grupo. Aunque en realidad, el acceder desde cualquier cliente de red, dependerá de la seguridad a implantar.

LA PRIMERA CAPA

De más está decir que un usuario que logre conectarse a nuestro TS, y además con credenciales de administrador, quién sabe lo que podría pasar.
Para comenzar a barajar opciones de seguridad, debemos comenzar con un cortafuegos. El servidor de seguridad de la casa Microsoft, el ISA Server, viene provisto por un cortafuegos, un moderado sistema de detección de intrusos, un servidor de caché programable y amplias posibilidades de otorgar o denegar los accesos, tanto de entrada como de salida, de protocolos o paquetes IP. Utiliza autenticación integrada a AD, filtrado de acceso a determinadas direcciones IP (de entrada y salida) y, si es que lo requerimos, en franjas horarias predefinidas. Este servidor será nuestra primera capa de seguridad con la que se encontrará el atacante. Hay que recordar que debemos publicar nuestro TS en el servidor de seguridad y rutearlo a la dirección IP interna donde esté ubicado el o los equipos. Idéntica tarea si el ISA está conectado detrás de un router.

Nota: Las capas de seguridad que forman la estrategia de defensa en profundidad incluyen el despliegue de medidas de protección desde los enrutadores externos hasta la ubicación de los recursos, pasando por todos los puntos intermedios. Con el despliegue de varias capas de seguridad, ayuda a garantizar que, si se pone en peligro una capa, las otras ofrecerán la seguridad necesaria para proteger sus recursos.

¿APLICACIÓN O ESCRITORIO DEFINIDO?

Tendremos que decidir si le daremos al usuario un acceso completo al escritorio o solo a determinadas aplicaciones. Si la primera opción es la que vale, sería conveniente utilizar los beneficios de un GPO sobre Active Directory para restringir lo más posible los accesos al sistema. Se recomienda encarecidamente que el servidor de aplicaciones TS, no se encuentre sobre un controlador de dominio, debido a que en el peor de los casos que algún intruso pueda llegar a entrar, estaría directamente en el DC, y por lo tanto tendría acceso también a los demás DC de la empresa.
Lo mejor en caso de otorgar acceso al escritorio, es implantar una UO dentro del dominio, y aplicarle una

política de grupo con sus restricciones. Para limitar el acceso a determinadas aplicaciones podremos utilizar el mismo GPO o ASPEC.exe, que es una utilidad del kit de recursos de Microsoft. Esta última aplicación nos sirve para permitir o denegar a los usuarios (no a los administradores) las aplicaciones que agreguemos a la lista.

CONFIGURACIONES Y PLANTILLAS

Es necesario utilizar configuraciones distintas para los servidores de aplicaciones, los servidores de archivos y los servidores Web para maximizar su seguridad. Debe tener en cuenta que cuantas más funciones desempeñe cada servidor individual, más vulnerable será a los ataques.
Si ubicamos el o los servidores TS a un nivel de UO dentro del dominio, aplicándoles las plantillas de seguridad básica directamente a la UO, y luego modificando determinados aspectos, según la función del servidor, obtendremos una organizada herencia de seguridad predeterminada para los equipos de esa UO. Puede elegir la plantilla a aplicar, entre la plantilla básica, la orientada a controlador de dominio y la segura.
Las plantillas se pueden aplicar a la directiva del dominio UO o equipo local, importar a un objeto de Directiva de grupo, utilizar para realizar un análisis de la seguridad o asignar automáticamente mediante la herramienta de línea de comandos Secedit.
Es posible aplicar una plantilla de seguridad directamente a la directiva del equipo local si el equipo no pertenece a un dominio.
Importante: Estas plantillas de seguridad están construidas asumiendo que se aplicarán a equipos con Windows 2000 que utilizan la configuración de seguridad predeterminada para Windows 2000. En otras palabras, estas plantillas modifican incrementalmente la configuración de seguridad predeterminada si están presentes en el equipo. No instalan la configuración de seguridad predeterminada para después hacer las modificaciones.

CARACTERÍSTICAS DE LOS SERVIDORES DE TS A TRAVÉS DE INTERNET

- Los Servicios de Terminal Server trabajan con el software de emulación de terminal enviando al servidor sólo pulsaciones de teclas y movimientos del mouse. El servidor de Servicios de Terminal Server lleva a cabo todas las manipulaciones de los datos localmente y retransmite su presentación en la pantalla. Este planteamiento permite un control remoto de los servidores y la administración centralizada de las aplicaciones, lo que reduce al mínimo el ancho de banda de red necesario entre el servidor y el cliente.

- Los usuarios pueden obtener acceso a los Servicios de Terminal Server en cualquier conexión TCP/IP, incluidas las de Acceso remoto, Ethernet, Internet, inalámbricas, red de área extensa (WAN) y red privada virtual (VPN)

- Permite el uso de aplicaciones de 32 bits basadas en Windows desde dispositivos que pueden o no estar basados en Windows, como Windows para Trabajo en Grupo 3.11 o posterior; Terminales basados en Windows (dispositivos Windows CE); Clientes basados en MS-DOS; Terminales UNÍS; Macintosh; Clientes no basados en Windows que requieran el uso de un complemento a terceros.

- Requiere el mínimo de espacio de disco, memoria y configuración para los clientes de los Servicios de Terminal Server.

- Simplifica el soporte de los equipos remotos y los entornos de sucursal.

- Proporciona seguridad y administración centralizadas.

- No altera las aplicaciones ni la infraestructura de red existente.

- Inclusive, instalando el Client Access de IBM en nuestro TS, se puede obtener acceso a un AS/400, obteniendo un excelente acceso remoto a este servicio, sin necesidad de tener demasiado conocimiento en AS/400.

- Se puede configurar tanto para el modo administración remota o servidor de aplicaciones. En el modo administración remota, como su nombre lo indica, lo podremos utilizar más que nada para la administración de servidores. Lo pueden utilizar los usuarios con permisos de administrador, y como no requiere licencia adicional, solo lo pueden utilizar dos usuarios en forma concurrente como máximo.

El modo servidor de aplicaciones está orientado a ser utilizado para la distribución de aplicaciones y servicios, y requiere de licencias terminal services para los clientes, excepto que su sistema operativo de escritorio sea Windows 2000 profesional o Windows XP profesional.

EL ENVOLTORIO

Otra opción que sumará puntos a lo que seguridad se refiere, es montar el acceso a través de una VPN. Esto nos permitirá trabajar sobre un túnel IP en forma encriptada. Podremos agregar a esta VPN la seguridad de trabajar con L2TP (Layer 2 tunneling protocol) y utilizar el filtrado de IPSec. Esta última opción es la recomendada.

La herramienta Microsoft® L2TP/IPSec VPN Client, que puede descargarse gratuitamente desde el Web (www.microsoft.com/spain/technet/recursos/articulos/welcome3/asp?opcion=3010025), es un producto que permite a los equipos que ejecutan Windows 98 (todas las versiones), Windows Millennium Edition y Windows NT® Workstation 4.0 utilizar las conexiones L2TP (Protocolo de túnel de capa 2) con el protocolo IPSec (Seguridad de protocolo de Internet)

¿CLIENTE CLÁSICO DE TS O TSAC?

Una vez que está configurado el servidor, podremos acceder a él a través de tres métodos distintos. La forma clásica es con el cliente de TS. Otra opción es utilizar el TSAC que es el cliente avanzado de TS, el cual lo accedemos por web utilizando un control ActiveX. Y por último tenemos la forma de acceso por consola, que esta sería mas orientada a los administradores, a través de la consola MMC.
Cuando se instala el TS, en forma predeterminada atiende las peticiones del RDP (protocolo de escritorio remoto) en el puerto TCP 3389. No sería mala idea cambiar el puerto de escucha, al menos para desorientar a quien esté buscando este servicio específico en nuestra red. Para ello podemos cambiar una clave en el registro del servidor indicándole el nuevo puerto de escucha. También debemos cambiar los registros de los clientes para indicar el mismo puerto de escucha del servidor. El cliente que accede a través del TSAC, mejor dicho con el control ActiveX, no puede cambiar su puerto de escucha, al menos en Windows 2000. En XP.Net será otro capitulo.

Para cambiar el puerto predeterminado para todas las conexiones de TS:

Ejecutar Regedt32 y buscar el siguiente key:
HKEY_LOCAL_MACHINESystemCurrentControlSetControlTerminalServer
WinStationsRDP-Tcp
Buscar "PortNumber", modificar (donde dice 3389) y establecer el nuevo valor.

Para cambiar el valor de una conexión específica:

Ejecutar Regedt32 y buscar lo siguiente:
HKEY_LOCAL_MACHINESystemCurrentControlSetControlTerminal Server
WinStationsconnection
Buscar "PorNumber", modificar (donde dice 3389) y establecer el nuevo valor

Para cambiar los parámetros del lado del cliente:

1º- Abrir el administrador de clientes de TS.
2º- Clic en menú Archivo, nueva conexión y crear una nueva conexión. Después de ejecutar el sistente, asegúrese que está disponible la nueva conexión.
3º- Seleccionar la nueva conexión, en el menú archivo, clic en exportar. Guardarlo como nombre.cns.
4º- Editar el archive nombre.cns utilizando Notepad y cambiando el valor "Server Port=3389" a Server Port=xxxx" donde xxxx el nuevo puerto especificado para TS.
5º- Ahora importar el archive para el administrador de cliente. Deberá sobrescribir el archive actual, siempre que el que esté, posea el mismo nombre. Deberá reiniciar el ordenador para que surtan efecto los cambios realizados.

MODIFICANDO PLANTILLAS

Uno de los principales cambios se debe realizar después de aplicar las plantillas de seguridad, dependiendo de que plantilla se utilice, es asegurarse que las contraseñas cumplan los requerimientos de complejidad (directiva de seguridad). Habilitar el bloqueo de cuenta, activar y establecer a 30 minutos el bloqueo, y el mismo valor al establecer la cuenta.
Será fundamental contar con contraseñas de 8 caracteres de longitud, con los 96 caracteres posibles, puede tardarse 2.288 años en descifrarla (analizando 100.000 palabras por segundo). Esto se obtiene a partir de las 968 (7.213.895.789.838.340) claves posibles de generar con esos caracteres.

Partiendo de la premisa de que no se disponen de esa cantidad de años para analizarlas por fuerza bruta, se deberá comenzar a probar con las claves más posibles, comúnmente llamadas Claves Débiles.

Aplicando un aviso legal en el proceso de inicio de sesión, impedimos, o al menos entorpecemos, los ataques por diccionario o fuerza bruta. Otro punto a nuestro favor sería utilizar el carácter ALT + 255 en las contraseñas de administrador (no olvidar las cuentas de administrador locales), ya que de esta manera, al ser un carácter no imprimible, será más difícil ser capturado por sniffers, crackers o registros de pulsaciones.
Si para el servidor a utilizar, no vamos a compartir recursos como carpetas o impresoras, debemos

deshabilitar el servicio compartir archivos e impresoras. Si también deshabilitamos NETBIOS sobre TCP/IP junto con la opción anterior, evitamos el acceso no autenticado por sesiones nulas y un posible listado de usuarios y recursos compartidos. De todas maneras, si se necesita el servicio NETBIOS sobre TCP/IP para clientes heredados, no lo podremos deshabilitar. Si necesita utilizar este servicio, asegúrese de fijar la opción Restricciones adicionales para conexiones anónimas en 2 (No obtener acceso sin permisos anónimos explícitos) en el GPO (Directivas de seguridad local o dominio, directivas locales, opciones de seguridad). También deberá deshabilitar los servicios de alerta y mensajería, para evitar filtrar información de cuentas de usuarios.

Para evitar que se produzcan hackeos de los hashes LM y NTLM, debemos habilitar la autenticación NTLM v2. Si fijamos en el GPO el nivel de autenticación de Lan Manager a 2 (Enviar sólo respuestas NTLM) aseguraremos el proceso de autenticación desafío-respuesta.

Cambie el nombre de la cuenta Administrador , y cree un señuelo. Aplique la auditoria de inicio de sesión de cuentas y, por favor, revíselo periódicamente (Ver auditoria).

Una vez que se termine de realizar las modificaciones de seguridad, puede guardar su configuración en formato de plantilla ahorrarse algo de trabajo para su próximo servidor similar.

EL CORTAFUEGOS

El ISA Server se puede configurar para permitir accesos en franjas horarias predeterminadas y preconfiguradas, para usuarios o grupos de usuarios específicos. También podemos limitar accesos a servicios en base a la dirección IP del cliente remoto. Es sabido que las direcciones se pueden falsear, pero también es cierto que el atacante debería tener información suficiente como para intentar ingresar con alguna dirección IP del grupo que estaría habilitado para su acceso, además de la autenticación válida.

Podemos filtrar todos los paquetes IP menos los que busquen el puerto que hemos configurado para el TS. No debemos olvidar publicar el servidor TS en el ISA Server, y redireccionarlo a la ubicación interna del servidor.

Detrás del cortafuegos, si nuestra estructura lo permite, deberíamos implantar un IDS. El IDS está mas allá de este documento, aunque es otra capa muy interesante en lo que a seguridad se refiere.

Al implantar la seguridad de nuestros servicios, tendremos que ocuparnos de los virus que circulan en la red. En el pasado mes de diciembre, la empresa Symantec lanzó el Symantec Antivirus para ISA Server. Este antivirus promete analizar todo el tráfico que pasará por el ISA. (También la empresa Pandasoftware tiene disponible una versión Beta www.pandasoftware.es).

SERVICE PACKS Y CORRECCCIONES

Será indispensable para el correcto y seguro funcionamiento de nuestro servidor, también de los clientes, estar al día con los últimos service packs y correcciones que nos provee Microsoft. La empresa de las ventanitas lanzó el SUS Software Update Services (www.microsoft.com/windows2000/windowsupdate/sus/default.asp) desarrollado para facilitar la tarea de los administradores al actualizar tanto los servidores como los clientes.

El SUS cuenta con una aplicación servidor, y otra cliente. Supervisa las actualizaciones actuales y las compara con las disponibles. Este sistema, a nivel de prestaciones sería el intermedio entre el Hfnetchk. exe y el SMS. La única pega que nos encontraremos con el SUS, es que está desarrollado para trabajar con Windows 2000 (service pack 2) en adelante.

Los atacantes se aprovechan de las vulnerabilidades de los sistemas para realizar sus ataques. Deberemos instalar todas las actualizaciones y correcciones para anular esas vulnerabilidades. Por más que tengamos nuestra primera capa de protección que será el firewall, en caso que logren saltar esa primera capa, no se deberá tener ninguna vulnerabilidad al alcance, al menos ninguna que tenga una corrección disponible.

AUDITORIA

A veces, los ataques más sutiles son los más peligrosos, ya que pasan desapercibidos y es difícil determinar los cambios realizados.

Es imprescindible habilitar la auditoria, como mínimo de los procesos de inicio de sesión, tanto los correctos como los erróneos.

De nada servirá tener habilitada la auditoria si no se chequea periódicamente. Sobre todo, revisar si alguien ha intentado autenticarse con la cuenta señuelo de administrador que hemos creado.

Tenga en cuenta que lo primero que intentará hacer un hacker, al menos uno experimentado, será deshabilitar la auditoria. Este evento es registrado por la auditoria del sistema. Será de vital importancia estar al tanto de este suceso.

Si un sistema de administración controla regularmente los registros para detectar suceso específicos, y extrae y reenvía los detalles a una base de datos de administración, se capturarán los datos necesarios y, por lo tanto, podrá establecer que los archivos de registro se sobrescriban y podrá consultar más claramente los sucesos que interesan.

Aquí tenemos un par de aplicaciones de ejemplo:

Dump Event Log es una herramienta de línea de comandos que se incluye en Windows 2000 Server Resource Kit, suplemento uno. Guarda un registro de sucesos de un sistema local o remoto en un archivo de texto separado por tabulaciones. Este archivo puede importarse a una hoja de cálculo o base de datos para realizar una investigación más detallada. La herramienta también sirve para filtrar determinados tipos de sucesos que se desea incluir o excluir.

EvenCombMT es una herramienta con varios subprocesos que analiza registros de sucesos de varios servidores al mismo tiempo generando un reporte de cada servidor, con sus criterios de búsqueda.

DESHABILITAR SERVICIOS

En las instalaciones predeterminadas de Windows 2000, el sistema instala y levanta muchos servicios que, tal vez, no utilicemos nunca. Mientras menos servicios tengamos habilitados, menos posibilidades le estaremos dando al intruso.

Por ejemplo: si no utilizamos los servicios de telnet, tendremos que deshabilitarlos. Lo mismo con el servicio de protocolo simple de transferencia de correo SMTP. Por nombrar mas servicios probablemente innecesarios según la configuración, cliente DHCP, Dfs, Fax, SharedAccess, Mnmsrvc (escritorio remoto compartido NetMeeting), etc. Lógicamente dependiendo de cada configuración de servidor.

Desde luego que si no utilizaremos el TSAC en nuestro TS, tampoco hará falta, a menos que tengamos dos servidores en uno (TS y WEB), utilizar los servicios de IIS.

Podremos utilizar el glosario de servicios de Microsoft, que está en la siguiente ubicación:
http://www.microsoft.com/windows2000/techinfo/howitworks/management/w2kservices.asp

Una vez que considere que ha realizado todas las configuraciones necesarias, aplicados los services packs, correcciones, modificaciones de plantillas etc., puede utilizar el Microsoft Baseline Security Analyzer (http://www.microsoft.com/security/mstpp.asp) para chequear la configuración de seguridad aplicada. El MBSA hará el trabajo por usted, al revisarle sus servidores y/o equipos con lo que Microsoft considera, como el nombre de la herramienta lo indica, la línea básica de seguridad. Esta útil herramienta le indicará si falta alguna actualización o corrección, además de posibles agujeros de seguridad, y sobre todo, como resolverlos.

Pero luego de tomar estas medidas, no todo será tranquilidad (nunca lo será) si no tomamos muy en serio la seguridad interna, sin descuidarnos de la ingeniería social. Las encuestas nos dicen que la mayoría de los ataques, al menos los exitosos, se producen dentro de la misma empresa.

Artículo cedido por Sna@ke. www.vilecha.com

Introducción frugal a VBScript (1ªParte):

Primeramente daros la bienvenida a todos un número más, espero que el artículo preparado para este número sea del agrado de todos y gracias por leerme. Pero ya es hora de ponerse manos a la obra.
Si me lo permitís, quiero empezar dando las razones por las que elegí este tema para este número de la ezine:

Hoy en día aún se siguen usando archivos bat, aún sigue siendo uno de los temas más demandados en Internet y aún se siguen dando en los estudios relacionados con la informática, especialmente en todo lo relacionado con la administración de sistemas. Y sin duda siguen siendo muy útiles, pero, por lo menos en mi opinión, es una pena que mucha gente no sepa que hay vida más allá del batch, que hay mejores alternativas para casi todo lo que todavía usan archivos bat. Los archivos bat siguen siendo útiles para ciertas tareas, pero fuera de esos ámbitos muy restringidos hoy en día no tienen sentido.

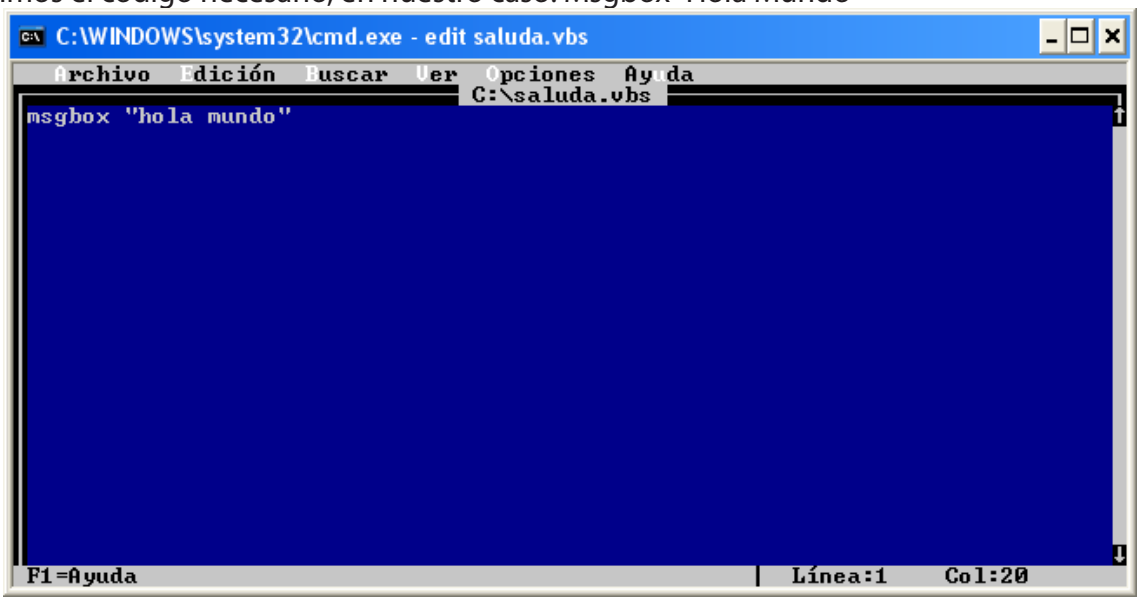
Actualmente hay varias alternativas, incluyendo al reciente PowerShell que viene de serie con Windows 2008, pero también es instalable en algunas versiones anteriores. Pero hay más alternativas, incluyendo la de instalar componentes para usar algún lenguaje que te guste, por ejemplo PerlScript. Una de esas alternativas es VBScript (Visual Basis Script) en archivos vbs o vbe y ejecutados directamente por wscript.exe de forma análoga a los archivos bat, y lo mejor es que viene de serie en todos los Windows desde el Windows 98.

Actualmente podemos usar archivos vbs para ayudar en la administración de equipos Windows de forma análoga a lo que hacen con Perl los de Linux/UNIX (realmente casi todos los usuarios de este potente lenguaje de script son administradores de sistemas Windows). Pero fuera de este ambiente, el uso de este potente lenguaje de script está limitado a alguna web ASP y poco más. Parece que casi todos los demás se olvidaron de él, aunque en otro tiempo no fue así, y gusanos tan conocidos como el "I love you" se valían de este lenguaje… pero poco a poco la gente se paso a clientes de correo web y el tema se olvido, aunque en los últimos meses parece que renace el interés por este lenguaje, incluso un vecino que aspira a convertir en hacker, hace cuestión de unos meses me vino con un tutorial (aparentemente impreso de algún foro), que aunque estaba espectacular porque detalla de forma concreta algunos de los objetos más importantes, en la parte de lo básico del lenguaje flojeaba y precisamente quería que le explicase esas partes. Y me dije, porque no hacer uno básico, adecuado para los que empiezan, de nada sirve uno con objetos como filesystemobject si al final la gente no sabe lo básico.
Bien, pues este pequeño tutorial es para dar una muy pequeña introducción al vbscript (de ahí lo de frugal), lo suficiente para que empecéis a hacer vuestros primeros scripts y para ver algunas de sus posibilidades. Realmente para alguien que ya conozca Visual Basic 6 o similar, pues ya puede hacer sus primeros pinitos.

Empezando que es gerundio:

Como por algún sitio hay que entrar en faena y todos los libros empiezan por el hola mundo típico, pues hagamos nuestro pequeño hola mundo en vbscript:

- Lo primero es crear un archivo de texto con extensión "vbs" (o vbe) y editarlo. Para ello podéis usar el notepad, el Wordpad, el Ultraedid o el EDIT de DOS. Por ejemplo en la linea de comandos con el edit sería: EDIT saluda.vbs En donde saluda es el nombre del archivo.
- Escribimos el código necesario, en nuestro caso: Msgbox "Hola Mundo"

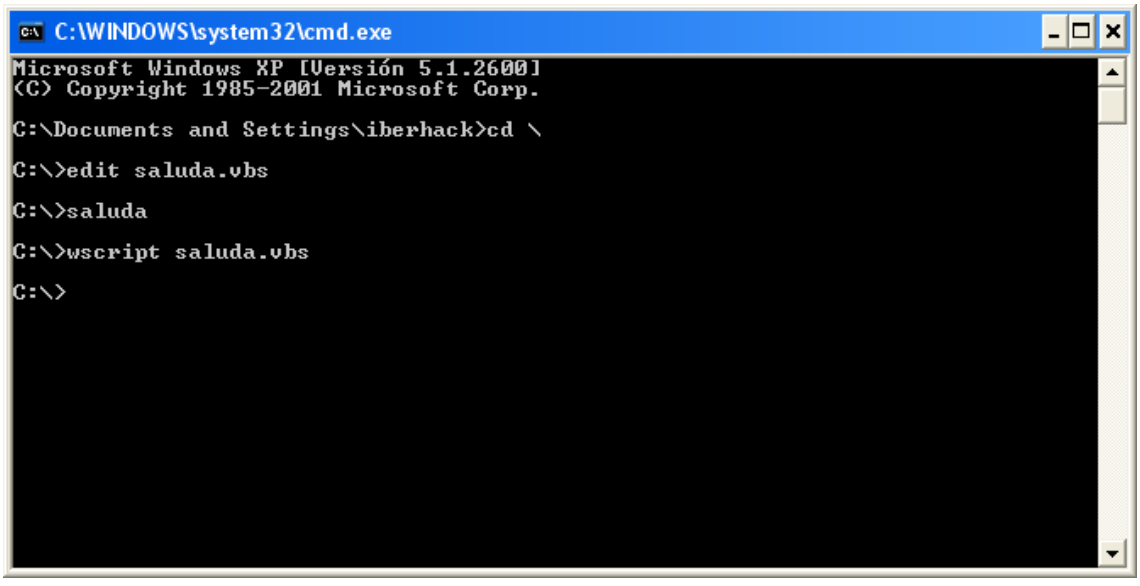


- Guardamos y salimos.
- Ya solo falta ejecutar el script, para ello llega con poner el nombre y darle a enter.

Saluda

Otra opción es:

Wscript saluda.vbs



- Al ejecutarlo nos saldrá una ventana como esta:

Pues ya tenemos nuestro primer script en vbscript.
Como soy de los que opina que un ejemplo como el anterior no es para nada ilustrativo de lo que después se encuentra uno, simplemente es el programa más sencillo que se puede hacer, pues toca el segundo ejemplo, también muy, pero que muy sencillo, pero en donde se pueden observar las posibilidades de este lenguaje:

```
'Ejemplo 2. Autor: iberhack
Dim reg
Set reg =CreateObject("WScript.Shell")
Dim val
val= reg.regread("HKEY_LOCAL_MACHINE\HARDWARE\DESCRIPTION" _
& "\System\CentralProcessor\0\~MHz")
Msgbox(val)
```

Después ya explicaré con detenimiento el código, por ahora simplemente deciros que hace, pues es algo tan sencillo como, que lee en el registro el valor de la velocidad del procesador de vuestro equipo y lo muestra.

En mi caso muestra 2009, que significa 2009 MHz o lo que es lo mismo 2GHz, que es la velocidad del procesador del equipo en donde escribo este artículo, un Atlhon64 X2 3800+.
Alguno ya estará viendo posibilidades a esto y se pregunta: ¿Podemos leer el registro fácilmente pero, se podrá igual de fácil escribir en el registro? Pues la respuesta es sí… pero lo mejor de todo, sin que haga problemáticas preguntas al usuario. Pero ya se llegará.
Manos a la obra: MSGBOX, InputBOX, Bucles…
Para poder sacarle todo el partido a este lenguaje de scripts hay que conocer sus estructuras de control, funciones que nos ofrece el lenguaje, operadores… Quien haya programado en Visual Basic ya conocerá casi todo esto, porque salvo pequeñas diferencias (como la inexistencia del GOTO) por lo demás son iguales.
Antes de entrar en las estructuras de control, comentar 2 funciones que usaremos a lo largo del resto del tutorial: Msgbox y Inputbox. Posteriormente se hará un pequeño resumen de variables y operadores.

MSGBOX

Msgbox() esuna función de Windows para mostrar mensajes, como los vistos en los ejemplos, la forma usual de uso es pasarle un string, ejemplo:

```
Msgbox("Hola")
```

NOTA: en este caso los paréntesis no son obligatorios (como se vio en el primer ejemplo), y con ponerle una cadena de caracteres entre comillas funciona igual. Aun así deberíais ponerlos.

Pero aunque este es el uso más simple y más usado de un MSGBOX, realmente tiene muchas más posibilidades, en el ejemplo siguiente se muestra un uso más avanzado de MSGBOX:

```
Dim a
a=msgbox("Las has jodido",4117,"Cagada")
msgbox(a)
```

Que da como resultado, una vez ejecutado:

Y al pulsar Reintentar:

Si en lugar de Reintentar dieramos a Cancelar:

Como veis devuelve un valor diferente dependiendo de que botón pulsemos.
La explicación en detalle del ejemplo es:
La primera línea (Dim a) es la declaración de la variable "a", no voy a explicar en detalle que es una variable, pero para el que sea nuevo en programación decir que una variable es como una caja para contener cosas, como puedo tener muchas, pues tengo que darle un nombre.
La segunda línea (a=msgbox("Las has jodido", 4117,"Cagada")) es en donde se hace todo el trabajo. En ella hacemos una llamada a la función msgbox, para que muestre el texto "Las has jodido", y que ese mensaje tenga el título "Cagada" y también le indicamos que botones va a mostrar (4117 es el código para que muestre los botones Reintentar y Cancelar), en este caso devuelve un valor, valor que guardamos en la variable a, y que representa que botón se pulso (cada posible botón tiene un número asociado, con ello podemos saber que botón se pulso: Reintentar 4 y Cancelar 2).
La tercera línea (msgbox(a)) es una llamaba simple a la función msgbox para que muestre el contenido de la variable a.
La forma general de la función MSGBOX es:

```
Variable=msgbox(texto [,Configuración [,Título]])
```

Lo único obligatorio es el texto. Si además especificamos una configuración que no sea la de por defecto (la de mostrar simplemente un mensaje), entonces también necesitamos la variable para que guarde lo que devuelve la función. El título solo puede estar presente si establecemos una configuración, nunca es obligatorio.

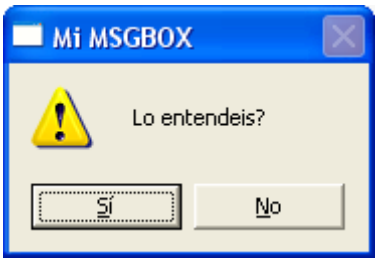
Llegamos a la parte en donde mucha gente se pierde (en parte porque muchos tutos se pone una tabla pero no se dice que hacer con ella), y es en que hay que poner en Configuración. Bien empecemos por la tabla que ponen en todos los libros:

0	**Solo botón OK, es la opción por defecto cuando no pones nada**
1	**Botones OK y Cancelar**
2	**Botones Abortar, Reintentar y Ignorar**
3	**Botones Si No y Cancelar**
4	**Botones Si y No**
5	**Botones Reintentar y Cancelar**
16	**Icono de error**
32	**Icono de pregunta**
48	**Icono de peligro**
64	**Icono informativo**
256	**El botón por defecto es el segundo**
512	**El botón por defecto es el tercero**
4096	**Mensaje de sistema, mostrará un icono en la barra de título, la alerta será independiente de la ventana, aunque cambies a otro programa, el mensaje se mostrará en pantalla.**

Pero lo que muchas veces no se especifica o no se especifica totalmente bien, es que nosotros podemos personalizar nuestro botón, y es tan fácil como coger lo que nos interesa y sumar, por ejemplo: queremos que se muestre siempre, y al mismo tiempo tenga los botones SI y NO y el icono de peligro, pues lo que tenemos que poner en configuración es el resultado de:
4096+4+48
Es 4148, que es lo que tenemos que poner en la zona de configuración, el resultado de:

```
a=msgbox("Lo entendeis?",4148,"Mi MSGBOX")
```

Es:



No se si os fijasteis, pero en la tabla, no todas las líneas las puse del mismo color, eso tiene una explicación y es la que usualmente suelen olvidar de mencionar en muchos tutos y libros, y es que solo podemos coger un número de cada grupo (no es necesario coger de todos los grupos), en mi caso separé los grupos por color. Pero también se pueden observar porque cada grupo tiene una cantidad de cifras distinta, por ejemplo el grupo de botones usa números de una cifra, y el de iconos de 2 cifras…

Del MSGBOX solo falta comentar una cosa, y es que representan los datos que devuelve, en este caso no hay duda, es una tabla muy sencilla (lo único que solo puede devolver el equivalente a los botones que le dijiste que tuviese, no puede 3 (que representa Abandonar) si solo hay los botones SI y NO…

1	OK
2	Cancelar
3	Abandonar
4	Reintentar
5	Ignorar
6	Si
7	No

Y con esto terminamos con la función MSGBOX.

NOTA: En algunos sitios os encontrareis con constantes tipo: "VBYESNO" en la parte de configuración de la función MSGBOX, y es otra posibilidad, pero cuidado lo mejor es que useis un valor numérico siempre, porque VBScript se puede encapsular en otras aplicaciones y en esos casos estas constantes dan problemas y al final te las tienes que memorizar igual y simplemente representan al número.

INPUTBOX

Si la principal funcionalidad de la función MSGBOX es mostrar información al usuario, la de la función INPUTBOX es justo la contraria, pedir al usuario datos. Realmente ambas son muy parecidas, y ambas, realmente, son funciones de la API de Windows.

Con la función inputbox pedimos que el usuarios introduzca cierta información, pero directamente la función no procesa lo que introduce el usuario, simplemente muestra una ventana con un cuadro de texto en donde el usuario escribe lo que quiera (puede ser un texto, un número, una fecha…), una vez que le dé a aceptar, esa información es cargada en una variable, si se da a Cancelar devuelve "" (una cadena vacia). Pero en ese proceso no se comprueba que la información se adecuada, por ello debemos tener siempre cuidado.

La estructura general de la función inputbox es:

```
Variable=InputBox(texto[,titulo][,valorxdefecto][,xpos][,ypos])
```

Aunque en algunos sitios veréis:

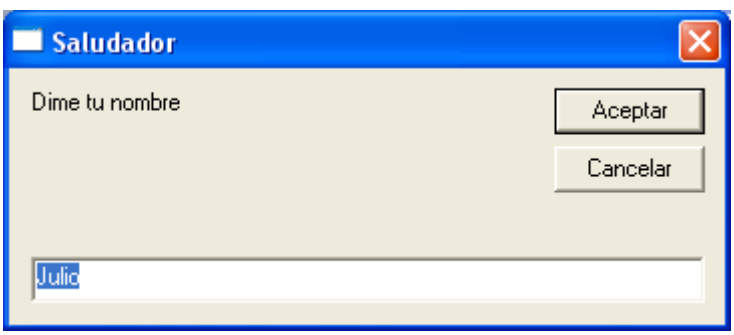
```
Var=InputBox(texto[,titulo][,valorxdefecto][,xpos][,ypos][,fichayuda,cont])
```

Esta última forma jamás la usareis en un script vbscript. Esta es si la usáis dentro de Visual Basic, Delphi…

Como siempre lo mejor es ver un ejemplo

```
dim nombre
nombre=InputBox("Dime tu nombre", "Saludador", "Julio", 20, 30)
MsgBox("Buenos días " & nombre)
```

El resultado es:



Y algo que no podéis apreciar en esta captura, su posición es casi la esquina superior izquierda, porque esta ventana podemos hacer que aparezca en donde queramos, de ahí la opción de **xpos** y **ypos.** Aunque lo usual es usar la función input box con un texto solo, o con un texto y un título, o un texto con título y su valor por defecto. Las otras propiedades no se suelen usar.

En detalle: texto es el texto que aparece encima del campo en donde teclear nuestra respuesta, titulo es el título de la ventana (no necesaria esta propiedad), valorxdefecto es que nosotros podemos darle en la llamada que aparezca un valor, algo muy interesante si es lo que se suele responder (tampoco es obligatoria esta propiedad). Xpos, ypos (no necesarias) indican la posición a partir de la cual se dibuja la ventana (teniendo en cuenta que la posición 0,0 es la esquina superior izquierda.

VARIABLES

En VBScript no es necesario declarar las variables, como pasa en casi todos los lenguajes de script, tampoco hay que declarar el tipo, por defecto todas son del tipo Variant, que no es más que un contenedor que puede almacenar cualquier tipo de variable y dependiendo del dato lo tiene almacenado como entero, texto…

Todo lo anterior esta muy bien y para un pequeño script de pocas líneas sirve. Pero para un script de cierto tamaño o complejidad, incluso para los de solo un par de líneas, es recomendable declarar las variables. Y a partir de cierto tamaño, lo mejor es obligar a que se tengan que declarar todas las variables, para ello hay que escribir al principio del archivo la siguiente directiva:

```
Option Explicit
```

Con ello, si se usa una variable sin declararla previamente, pues dará error.
En cuanto a la forma de declarar una variable, pues es muy sencillo, se usa etiqueta **Dim** seguida del nombre de la variable, ejemplo:

```
Dim variable1
```

Con ella declaramos la variable1. Además en la misma línea podemos declarar más de una variable, separando los distintos nombres por una coma (,), por ejemplo:

```
Dim nombre, apellido, fnacimiento, edad
```

Como ya se indicó, todas las variables son del tipo Variant, pero el tipo Variant simplemente sirve para no tener que hacer cast (cambios de tipo) a cada operación que hagamos, realmente internamente, dependiendo del tipo, pues guarda el valor como un tipo u otro, o hace la conversión si es posible cuando se realizan operaciones, por ejemplo si hacemos una resta, intenta transformar los valores que contienen sus variables a un tipo numérico, en cambio con la operación concatenación de caracteres, pues lo que hace es convertir lo que guarda en un texto. Realmente ninguna novedad, en los principales lenguajes de script ocurre algo similar, por ejemplo Perl, PHP…

Los distintos tipos de datos que se usan internamente son:

Empty (vacío)	un Variant sin inicializar. Es un 0 para variables numéricas y una cadena vacía ("") para variables de texto. Esto significa que si sumo 5 + var y la variable var no tiene nada, el resultado será 5, al tomarse la variable como 0.
Null	es un Variant que no contiene datos válidos intencionadamente.
Boolean (booleano)	es un Variant que solo puede tener el valor lógico de verdadero o falso. Es True o False (-1 y 0).
Byte (octeto)	contiene un entero entre 0 y 255.
Integer (entero)	contiene un número entre -32.768 y 32.767.
Currency (moneda)	-922.337.203.685.477,5808 a 922.337.203.685.477,5807.
Long (entero largo)	contiene un entero entre -2.147.483.648 y 2.147.483.647.
Single (precisión simple)	contiene un número en coma flotante entre 3,402823E38 y -1,401298E-45 para valores negativos, y entre 1,401298E-45 y 3.402823E38 para valores positivos.
Double (doble precisión)	contiene un número en coma flotante entre -1,79769313486232E308 y -4,94065645841247E-324 para valores negativos, y entre 4,94065645841247E-324 y 1,79769313486232E308 para valores positivos.
Date (Time) (fecha)	Contiene un número que representa una fecha (y una hora) entre el 1 de Enero del año 100 y el 31 de Diciembre de 9999.
String (cadena)	contiene una cadena de longitud variable, de hasta 2 mil millones de caracteres de longitud.
Object (objeto)	contiene un objeto.
Error	contiene un número de error.

Ya veis que realmente hay muchos tipos, pero al empezar en el fascinante mundo de los lenguajes de vbscript esto no os tiene porque preocupar.

OPERADORES

Como casi todo el mundo se salta la parte de los operadores (principalmente porque suelen coincidir en un 90% las cosas como son), pues simplemente os los indico:

Aritméticos

Suma	+
Resta	-
Multiplicación	*
División entera	\
División (con decimales)	/
Módulo (resto de la división entera)	Mod
Potencia	^

Concatenación

Unión de cadenas (texto)	&

Comparación

Igual que	=
Mayor que	>
Menor que	<
Mayor o igual que	=>
Menor o igual que	<=
Distinto	<>

Lógicos

Negación (No lógico)	Not
Conjunción (Y lógico)	And
Disyunción (O lógico)	Or

Como la mejor forma de ver como funcionan es con ejemplos, a lo largo de los ejemplos siguientes del texto, veréis como funcionan una buena parte de ellos.

ESTRUCTURAS CONDICIONALES

IF

Estructura básica:

```
If condición then
      Acciones_condición=verdad
Else
      Acciones_condición=falsa
End If
```

Ejemplo:

```
Option Explicit
Dim num
num=Inputbox("Dime un número entre 1 y 10")
If num>0 and num<11 then
      msgbox("Bien Hecho")
Else
      msgbox("Mal echo")
End If
```

La condición es *num>0 and num<11*, si el valor introducido esta comprendido entre 1 y 10 (ambos inclusive) la condición es verdadera y por tanto mostrara el mensaje *Bien echo*, en caso de no haber introducido un número o que esté fuera de ese rango, el mensaje que mostrará será *Mal echo*. Si la condición es verdadera (revisar los operadores de comparación y lógicos) hago lo que hay después del *then*, si es falsa, lo que hay después del *Else*. El Else no tiene porque estar presente.

CASE

Estructura básica:

```
Select case variable
Case n
Acción 1
Case p
Acción 2
Case else
Acción por defecto
End select
```

Ejemplo:

```
Option Explicit
Dim dia
dia=Inputbox("Dime un dia de la semana, en minusculas")
Select case dia
case "lunes"
msgbox("1º dia de la semana")
case "martes"
msgbox("2º dia de la semana")
case "miercoles"
msgbox("3º dia de la semana")
case "jueves"
msgbox("4º dia de la semana")
case "viernes"
msgbox("5º dia de la semana")
case "sabado"
msgbox("6º dia de la semana")
case "domingo"
msgbox("7º dia de la semana")
case else
msgbox("Ni los días de la semana te sabes")
End Select
```

En este caso, dependiendo del valor de una variable, se hacen unas acciones específicas para ese valor, en caso de no ser ninguno de ellos, se hace lo que está en el *case else* (parte que no es obligatoria), solo se hace de los valores que nos interese comprobar… Tanto se puede hacer de números, texto, fechas… Tened cuidado con las mayúsculas y minúsculas cuando reviséis texto, ya que en ese caso no es lo mismo *lunes* que *Lunes* que *LUNES*. Como regla general VBScript no es *case sensitive* (no distingue las mayúsculas de las minúsculas), pero este es un ejemplo de lo contrario.

BUCLES

FOR

Estrutura básica:

```
For valorinicial to valorfinal step cuantovaria
      Acciones
Next
```

Ejemplo:

```
Option Explicit
Dim p, s
s=10
For p=5 to 0 step -1
Msgbox(s-p)
Next
```

Esta es la estructura que necesitamos cuando vamos a realizar una acción un número finito de veces, ejemplos: leer 10 números, sumar los números pares del 0 al 20… Y es muy fácil de usar, simplemente necesitamos una variable, a la que le daremos un valor inicial, y que por cada vez que se repita, variará su valor lo que hay en *step*, hasta llegar al valor final que es lo que hay después de *to* y antes de *step*.

Existe una forma de terminar la ejecución de un bucle for sin hacer todas las repeticiones, es con la orden ***exit for***,

FOR EACH

Estructura básica:

```
FOR EACH variable IN array
      Acciones
Next
```

Ejemplo:

```
Option Explicit
Dim dia
Dim dias
dias=array("lunes","martes","miercoles","jueves","viernes","sabado","domingo")
FOR EACH dia in dias
      msgbox(dia)
Next
```

Por ahora no comente nada de arrays, y esta es una estructura para trabajar con arrays, simplemente contar que se declaran como una variable normal, pero con unos paréntesis y un número al final. En lugar de guardar un solo dato guarda el número que nosotros le indiquemos entre parentesis, por ejemplo:

```
Dim Num(3)
Num(0)=8
Num(1)=15
Num(2)=0
Num(3)=4
```

Otra opción para lo hacer lo mismo sería (aunque en este caso sin declarar el número de elementos):

```
Dim Num
Num=array(8,15,0,4)
```

Si hay una continuación de este artículo, pues explicare con más detalle los arrays.

Esta estructura es muy útil y al mismo tiempo de las menos usadas, especialmente poco usada por los programadores noveles.

La idea de funcionamiento es que se va a repetir el buque tantas veces como elementos tenga el array, y en cada pasada, desposita el valor correspondiente en una variable individual.

En este bucle también podemos usar la orden ***Exit for*** para terminar la ejecución del bucle antes de lo esperado.

DO

Estructura básica:

```
Do
      Acciones
Loop
```

o

```
Do while/until condicion
      Acciones
Loop
```

o

```
Do
      Acciones
Loop while/until condicion
```

Ejemplo:

```
Option Explicit
Dim numero
numero=2
Do
numero = numero * 3
msgbox(numero)
if numero> 30 then
Exit do
End If
Loop
```

o

```
Option Explicit
Dim numero
numero=2
Do while numero <30
numero = numero * 3
msgbox(numero)
Loop
```

o

```
Option Explicit
Dim numero
numero=2
Do
numero = numero * 3
msgbox(numero)
Loop Until numero = 54
```

Estamos ante la estructura de bucle más flexible de todas, pero también es cierto que es donde más errores se cometen por descuido, especialmente tened cuidado cuando hagáis bucles sin condición, a la mínima se convierte en un bucle infinito.
En donde aparece while/until es porque se puede usar while o until dependiendo de lo que quiera el programador. While se usa cuando queremos que se siga repitiendo mientras se da una condición, y Until se usa cuando queremos que se ejecute mientras no se da cierta condición.
Para terminar el bucle antes de darse la condición de fin, y en el caso de los bucles sin condición, podemos usar: Exit do, que una vez que se ejecuta termina la ejecución del bucle.
La principal diferencia de poner el while o until al principio (junto al Do) del bucle o al final del bucle (junto al Loop), es que con la segunda forma nos garantizamos que las acciones de dentro del bucle se van a ejecutar al menos una vez.

WHILE

Estructura básica:

```
While condición
Acciones
Wend
```

Ejemplo:

```
Option Explicit
Dim numero
numero=2
While numero < 10
msgbox(numero)
numero=numero*2
Wend
```

Estamos ante la estructura de bucle más usada y la más sencilla de todas. Pero en buena parte de los casos se adecua más al problema otra de las estructuras mencionadas, ya que de hacerla con un bucle While requeriríamos más líneas de código, lo que implica más posibles bugs.
Y con el while..wend terminamos las estructuras de control.

Funciones y Procedimientos:

Como casi todos los lenguajes, en VBScript podemos definir Funciones (Function) y Procedimientos (Sub). La forma es igual que en visual Basic.

PROCEDIMIENTOS

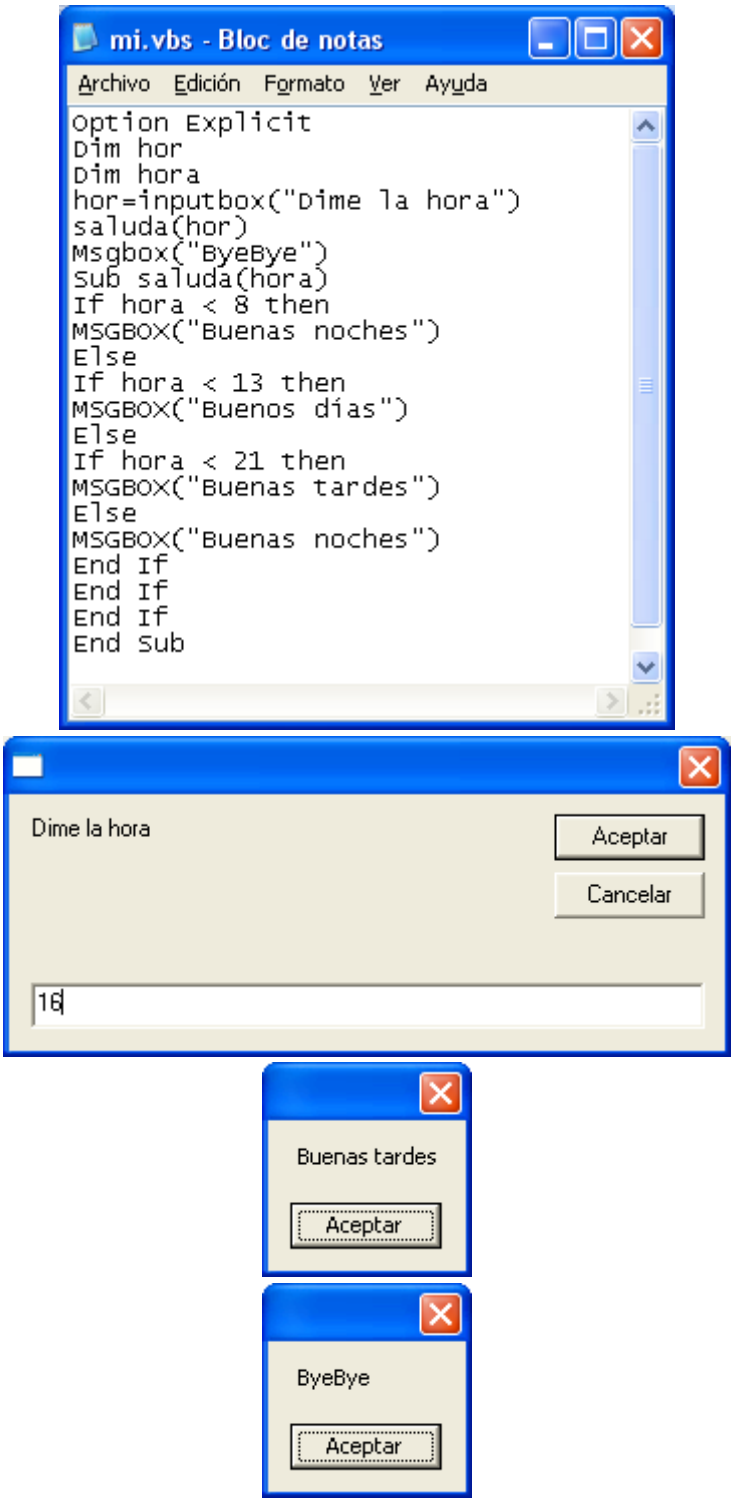
Estructura básica:

```
Sub nombre(argumentos)
Acciones
End Sub
```

Ejemplo:

```
Option Explicit
Dim hor
Dim hora
hor=inputbox("Dime la hora")
saluda(hor)
Msgbox("ByeBye")
Sub saluda(hora)
If hora < 8 then
MSGBOX("Buenas noches")
Else
If hora < 13 then
MSGBOX("Buenos días")
Else
If hora < 21 then
MSGBOX("Buenas tardes")
Else
MSGBOX("Buenas noches")
End If
End If
End If
End Sub
```

Para el que no tenga muchos conocimientos de programación, solo decirle que un procedimiento es un conjunto de líneas de código que hacen algo, algo que probablemente querríamos hacer varias veces, pero para no tener que hacerlo más de una vez, le damos un nombre, nombre que usaremos cada vez que queramos conseguir ese resultado. A diferencia de la funciones, no devuelven nada, simplemente se ejecutan. Puede recibir parámetros (datos para que haga algún cálculo o ajuste la que va a hacer a ellos)…



FUNCIONES

Estructura básica:

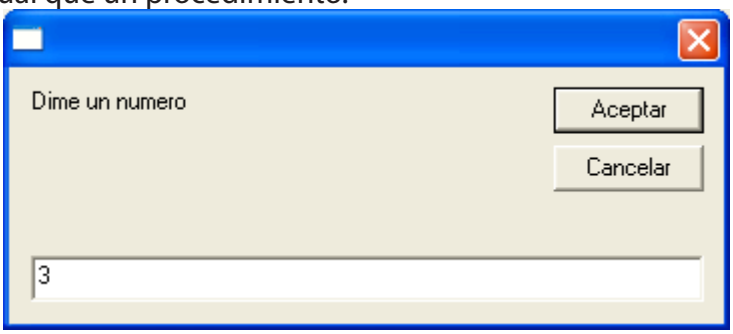
```
Function nombre(argumentos
Acciones
Nombre=algo
End Function
```

Ejemplo:

```
Option Explicit
Dim num
num = inputbox("Dime un numero")
num=cubo(num)
msgbox(num)
Function cubo(num)
cubo=num^3
```

End Function

Las funciones devuelven datos, ya vimos varias que vienen predefinidas, por ejemplo el inputbox. Solo comentar, que la forma de devolver los datos es usando el nombre de la función como una variable, una vez que se termine su ejecución, lo que haya en esa variable especial es lo que devuelve la función. Por lo demás funciona igual que un procedimiento.



FUNCIONES DEL LENGUAJE:

VBScript incorpora un buen surtido de procedimientos y funciones que podemos usar en cualquier momento, la mayoría son de conversión de tipos o para trabajar con cadenas, pero también las hay matemáticas…

ABS (número)
Recibe como argumento un número (o variable que lo contiene) y devuelve el mismo número sin signo. No produce ningún efecto con números positivos y elimina el signo en los negativos.
ARRAY (Elemento1, Elemento2, ... , Elemento N)
Devuelve una matriz con los elementos que recibe, separados por comas, como argumento. Si no se le pasan argumentos, devuelve una matriz de cero elementos.
ASC (carácter)
Devuelve el valor ASCII del carácter que recibe como argumento. Si recibe una cadena, devuelve el código ASCII del primer carácter de la misma. Si recibe un dato Null, devuelve un valor Null.
ATN (número)
Recibe un número (o variable que lo contiene) que representa a un ángulo en radianes y devuelve el arco tangente de ese ángulo.
CBOOL (número)
Recibe como argumento un número (o variable que lo contiene y devuelve un valor lógico. Falso si el número es 0 y verdadero en otro caso.
CBYTE (número)
Recibe como argumento un número (o variable que lo contiene) y lo convierte a un número en formato Byte.
CCUR (número)
Recibe como argumento un número (o variable que lo contiene) y lo convierte a un número en formato Currency.
CDATE (cadena)
Recibe como argumento una cadena alfanumérica representando una fecha y la convierte en un dato se subtipo DATE.
CDBL (número)
Recibe como argumento un número (o variable que lo contiene) y lo transforma a un número en formato Double.
CHR (número)
Devuelve el carácter asociado al código ASCII pasado como argumento. El número deberá estar comprendido entre 0 y 255.
CINT (número)
Recibe como argumento un número (o variable que lo contiene) y lo transforma en un dato con formato Int.
CLNG (número)
Recibe como argumento un número (o variable que lo contiene) y lo transforma en un dato con formato Long.
COS (número)
Recibe un número (o variable que lo contiene) que representa a un ángulo en radianes y devuelve el coseno de ese ángulo.

CSNG (número)
Recibe como argumento un número (o variable que lo contiene) y lo transforma en un dato con formato Single.
CSTR (argumento)
Recibe como argumento un dato (o variable que lo contiene) que no sea una cadena alfanumérica y lo devuelve transformado en una cadena.
DATE ()
No recibe ningún argumento y devuelve la fecha del sistema.
DATEADD (intervalo, cantidad, fecha)
Esta función devuelve el resultado de sumarle un periodo de tiempo a una fecha determinada. Recibe tres parámetros obligatorios:
El intervalo es una cadena de texto que indica la unidad de tiempo que queremos añadir a la fecha (horas, días, etc). Los posibles valores están indicados a continuación.

yyyy	año completo
q	trimestre
m	mes
d	día
w	semana
h	hora
m	minuto
s	segundo

La cantidad es el número de unidades del intervalo especificado que se le sumará a la fecha indicada. Este valor puede ser positivo, para referirse a fechas futuras o negativo para referirse a fechas pasadas.
La fecha es aquella a la que se le suman los intervalos especificados para que la función devuelva una nueva fecha como resultado de la operación.
Por ejemplo. Si tecleamos la siguiente línea de código:
nueva = DATEADD (“yyyy”,1,10-02-1996) resultado será 10-02-1997.
Esta función tiene en cuenta incluso los años bisiestos.
DATEDIFF (intervalo, fecha 1, fecha2)
Calcula la diferencia entre las dos fechas especificadas y la devuelve expresada en los periodos indicados en intervalo. Por lo tanto devuelve un resultado numérico. Los posibles intervalos a especificar son los mismos que en el caso anterior.
DAY (fecha)
Esta función recibe como argumento una fecha y devuelve un número que corresponde al día del mes de la fecha indicada.
EXP (número)
Recibe como argumento un número (o variable que lo contiene) y devuelve el número e elevado a la potencia indicada.
FIX (número)
Recibe un número (o variable que lo contiene). Devuelve la parte entera de un número, truncando los decimales. Si el argumento es un número negativo, esta función devuelve el primer negativo igual o mayor que encuentre.

FORMATCURRENCY (número, dígitos_decimales, cero_decimal, negativos_paréntesis, agrupar_dígitos)
Esta función se usa para representar números en formato de moneda, es decir, con la apariencia de valores económicos. Recibe cuatro argumentos. El primero de ellos es el número (o variable que lo contiene) que hay que representar como cifra económica. Los otros tres son opcionales.
El parámetro dígitos_decimales especifica cuantos dígitos se quiere que aparezcan a la derecha de la coma digital.
El parámetro cero_decimal indica si se quiere completar el número con ceros a la derecha de los decimales. P.e. Si se especificaron dos decimales en el parámetro anterior y un número tiene un solo decimal ¿Queremos un cero a la derecha de ese decimal? Si lo queremos, este parámetro será un -1. Si no lo queremos, el parámetro será 0.
El parámetro negativos_paréntesis especifica si queremos que los números negativos aparezcan entre paréntesis. Si lo queremos, pondremos -1. Si no, un 0.
El parámetro agrupar_dígitos indica si queremos que los dígitos aparezcan agrupados de tres en tres, en millares, millones, etc. Si lo queremos pondremos un -1. Si no lo deseamos así, pondremos un 0.
Estos tres últimos parámetros pueden recibir como valor, en lugar de -1 o 0, un -2. En éste último caso se toman las características establecidas en la configuración regional del sistema, en el panel de control de Windows.
FORMATNUMBER (número, dígitos_decimales, cero_decimal, negativos_paréntesis, agrupar_dígitos)
Esta función opera igual que la anterior, solo que se refiere a un formato genérico, sin asociar a ninguna moneda.
HEX (número)
Recibe como argumento un número decimal (o variable que lo contiene) y lo convierte en hexadecimal.
HOUR (hora)
Esta función recibe como argumento una variable que contiene una hora en formato hh:mm:ss y devuelve la hora como un número de 0 a 23.
INSTR (comienzo, cadena 1, cadena 2)
Busca la primera aparición de la cadena 2 dentro de la cadena 1. Los parámetros comienzo y comparación son opcionales. El parámetro comienzo indica a partir de que carácter de la cadena 1 se empieza a buscar la cadena 2. Si se omite, la búsqueda se inicia desde el primer carácter.
INSTRREV (cadena 1, cadena 2, comienzo)
Esta función es similar a la anterior, solo que empieza la búsqueda por el final de la cadena.
INT (número)
Recibe un número (o variable que lo contiene). Devuelve la parte entera de un número, truncando los decimales. Si el argumento es un número negativo, esta función devuelve el primer negativo igual o menor que encuentre.
ISARRAY (variable)
Esta función recibe como argumento un nombre de una variable y comprueba si es una matriz. Devuelve un valor booleano.
ISDATE (variable)
Esta función recibe como argumento el nombre de una variable y comprueba si es una fecha. Devuelve un valor booleano.
ISEMPTY (variable)
Esta función recibe como argumento el nombre de una variable. Devuelve un valor booleano. Es verdadero si la variable no está inicializada y falso si lo está

ISNULL (variable)
Esta función recibe como argumento un nombre de variable y devuelve un valor lógico. Verdadero si la variable contiene un Null. Falso si no lo contiene.
ISNUMERIC (variable)
Recibe una variable, supuestamente numérica. Devuelve un valor lógico verdadero si el parámetro que ha recibido es una variable que contiene un número y un valor lógico falso en caso contrario.
LCASE (cadena)
Recibe como argumento una cadena (o una variable que contiene una cadena) y la convierte a minúsculas.
LEFT (cadena, longitud)
Devuelve una sub-cadena compuesta por los caracteres que hay a la izquierda de una cadena. Los argumentos que recibe son una cadena (o variable que la contiene) y la longitud (cantidad de caracteres) de la sub-cadena.
LEN (cadena)
Recibe como argumento una cadena (o variable que la contiene) y devuelve la cantidad de caracteres que componen dicha cadena.
LOG (número)
Esta función recibe como argumento un número (o una variable que lo contenga) que deberá ser mayor de 0 y devuelve el logaritmo en base e de dicho número.
LTRIM (cadena)
Recibe como argumento una cadena o variable que la contiene. Devuelve la cadena sin los espacios en blanco que hay a la izquierda de la misma.
MID (cadena, inicio, longitud)
Recibe como argumentos una cadena (o variable que la contiene), una posición de inicio (o variable numérica que la contiene) y una longitud (o variable numérica que la contiene). Devuelve una sub-cadena extraída de una cadena original. La sub cadena tiene los caracteres expresados en longitud, contados a partir del carácter inicio. Por lo tanto, el primer argumento es alfanumérico y los otros dos son numéricos.
MINUTE (hora)
Esta función recibe como argumento una variable que contiene una hora en formato hh:mm:ss y devuelve los minutos como un número de 0 a 59.
MONTH (fecha)
Esta función recibe como argumento una fecha o una variable de fecha y devuelve un número del 1 al 12 que indica el mes de la fecha.
MONTHNAME (mes, abreviado)
Esta función recibe como argumento un número del 1 al 12 (o una variable que lo contiene) y devuelve el nombre del mes correspondiente. El otro parámetro que recibe es un valor lógico para indicar si el nombre debe aparecer abreviado o no.
OCT (número)
Recibe como argumento un número decimal (o variable que lo contiene) y lo convierte en octal.
REPLACE (cadena 1, cadena 2, cambia_por, comienzo, veces)
Esta función encuentra la cadena 2 dentro de la cadena 1 y la sustituye por cambia_por. Comienza a buscar a partir del carácter cuyo número de orden es el indicado en comienzo y, si la cadena_2 aparece mas de una vez en la cadena 1 la cambia el número de veces indicado.

RIGHT (cadena, longitud)
Devuelve una sub-cadena compuesta por los caracteres que hay a la derecha de una cadena. Los argumentos que recibe son una cadena (o variable que la contiene) y la longitud (cantidad de caracteres) de la sub-cadena.

RND()
Recibe un argumento vacío y devuelve un número aleatorio. Para que funcione correctamente, es necesario incluir en el código VBScript una línea con la instrucción RANDOMIZE.

ROUND (número, decimales)
Esta función redondea el número (o variable que lo contiene) y lo devuelve con el número de decimales expresado en decimales.

RTRIM (cadena)
Recibe como argumento una cadena o variable que la contiene. Devuelve la cadena sin los espacios en blanco que hay a la derecha de la misma.

SCRIPTENGINEBUILDVERSION ()
Esta función no recibe ningún argumento y devuelve el número de versión del motor de Script que se está utilizando.

SECOND (hora)
Esta función recibe una expresión de hora en formato hh:mm:ss y devuelve el número de segundos correspondiente.

SGN (numero)
Recibe un argumento numérico y devuelve un 1 si el número es positivo, un -1 si es negativo y un 0 si es 0.

SIN (número)
Recibe un número (o variable que lo contiene) que representa a un ángulo en radianes y devuelve el seno de ese ángulo.

SPACE (numero)
Recibe como argumento un número (o variable que lo contiene) y devuelve una cadena formada por espacios en blanco; tantos como especifica el número.

SQR (número)
Devuelve la raíz cuadrada del número (o variable numérica) que recibe como argumento.

STRCOMP (cadena 1, cadena 2)
Recibe como argumentos dos cadenas alfanuméricas (o variables que las contienen) separadas por comas. Si ambas cadenas son iguales devuelve un 0. Si la primera es mayor que la segunda, devuelve un 1. Si la segunda es mayor que la primera devuelve un -1. Si alguna cadena tiene un valor Null, devuelve Null. En este sentido debemos recordar que un carácter es mayor o menor que otro en función de sus códigos ASCII. Así, p.e., la a es mayor que la A, porque el código ASCII de la A es 65 y el de la a es 97.

STRING (número, carácter)
Recibe como argumentos un número (o variable que lo contiene) y un carácter (o variable que lo contiene). Devuelve una cadena compuesta por el carácter especificado, repetido las veces que indica el número.

STRREVERSE (cadena)
Recibe como argumento una cadena (o variable que la contiene) y devuelve la cadena tras invertir el orden de todos los caracteres que la componen. Así pues, el primero de la cadena original será el último de la cadena resultante.

TAN (número)
Recibe un número (o variable que lo contiene) que representa a un ángulo en radianes y devuelve la tangente de ese ángulo.
TIME ()
Esta función no recibe ningún argumento y devuelve una expresión que representa la hora del sistema en formato hh:mm:ss.
TIMESERIAL (número 1, número 2, número 3)
Esta función recibe tres parámetros numéricos y los convierte a una hora en formato hh:mm:ss. El número 1 debe estar comprendido entre 0 y 23; el número 2 debe estar comprendido entre 0 y 59 y el número 3 también.
TIMEVALUE (fecha)
Esta función recibe una variable de fecha y extrae y devuelve la parte de la hora.
TRIM (cadena)
Recibe como argumento una cadena o variable que la contiene. Devuelve la cadena sin los espacios en blanco que hay a la izquierda y a la derecha de la misma.
TYPENAME (variable)
Esta función recibe un nombre de variable y devuelve el nombre de subtipo del dato que contiene.
UCASE (cadena)
Recibe como argumento una cadena (o una variable que contiene una cadena) y la convierte a mayúsculas.
WEEKDAY (fecha, primer_día)
Esta función recibe dos parámetros. Es primero es una fecha o variable que la contiene; el segundo es una constante que indica el que es el primer día de la semana en nuestro país. Devuelve un número correspondiente al día de la semana de la fecha introducida. Las constantes para identificar que día de la semana es el primero en nuestro país son:

VBSUNDAY	Domingo
VBMONDAY	Lunes
VBTUESDAY	Martes
VBWEDNESDAY	Miércoles
VBTHURSDAY	Jueves
VBFRIDAY	Viernes
VBSATURDAY	Sábado

WEEKDAYNAME (día_semana, abreviado, primer_dia)
Esta función recibe un número de día de la semana, un valor lógico y una constante que indica cual es el primer día de la semana en nuestro país. Devuelve como resultado el nombre del día de la semana que corresponde al número introducido. Si el valor lógico es verdadero, el nombre que devuelve aparece en abreviatura. Las constantes que identifican el día de la semana son las mismas que en el caso anterior.
NOTA: Este listado de funciones estaba publicado en Programacion.com, hice un copy/paste y posteriormente lo retoqué un poco. Me gustaría poder afirmar que conozco todas las funciones, pero no es así, soló memorizo las que uso más a menudo, pero en un tuto hay que ponerlas todas. Gracias a programacion.com he podido ponerlas todas.

VAMOS A JUGAR A JUEGOS DE HOMBRES (POR FIN) ;-):

Vamos a empezar a trabajar con objetos. La potencia de VBScript, precisamente, radica en que podemos usar controles ActiveX directamente, especialmente algunos del propio sistema operativo, como es el caso de WSCript.Shell. Hay muchos, los más usados son:

WSCript.Shell	Para hacer un montón de acciones sobre SO, como ejecutar aplicaciones, leer o modificar el registro…
Scripting. Filesystemobject	Nos permite abrir, leer, escribir…sobre archivos. Con archivos y carpetas permite copiarlos a otra localización, moverlos, borrarlos, renombrarlos…
WScript.Network	Para hacer operaciones en red, como por ejemplo mapear unidades de red…

Antes de seguir, la forma de usar objetos ActiveX en VBScript es:

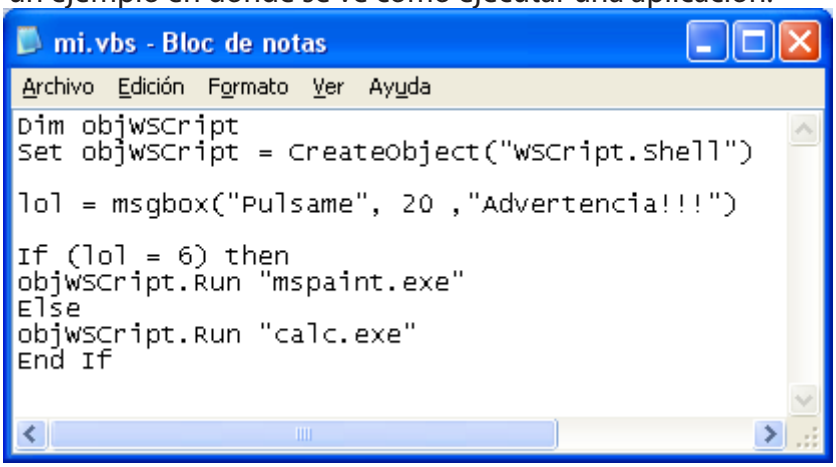
```
Set Objeto = CreateObject("Objeto a usar")
```

Ejemplo:

```
Dim mishell
Set mishell=CreateObject("WSCript.Shell")
```

Como no me voy a extender mucho más, puede que para el próximo número haga la siguiente entrega de este tuto, en donde ya se verán cosillas más interesantes, pues lo mejor es entrar en faena y ver algunos ejemplos interesantes, para que a partir de ellos podáis empezar a hacer cosillas.

Primero empezar por un ejemplo en donde se ve como ejecutar una aplicación:



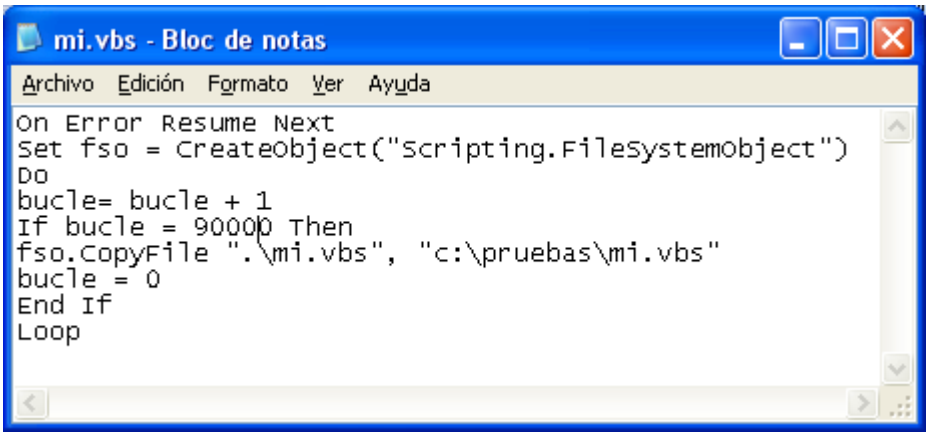
Código:

```
Dim objWSCript
Set objWSCript = CreateObject("WSCript.Shell")
lol = msgbox("Pulsame", 20 ,"Advertencia!!!")
If (lol = 6) then
objWSCript.Run "mspaint.exe"
Else
objWSCript.Run "calc.exe"
End If
```

Explicación:

Creamos un objeto Shell que llamamos objWSCript y usamos su método Run, este método ejecuta la aplicación que le pasemos. En este caso la aplicación depende de que botón pulsemos en un msgbox.

Segundo ejemplo: en este ejemplo vemos un pequeño script que se copia a si mismo a otra ruta, en concreto a C:\pruebas\

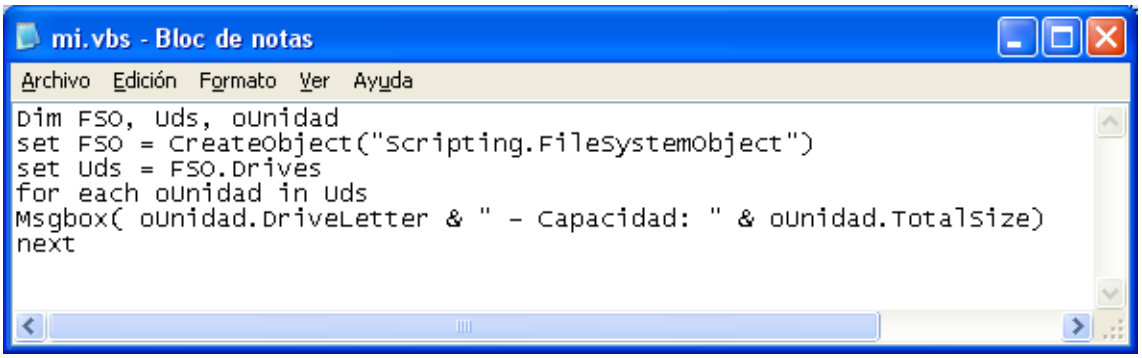


Código:

```
On Error Resume Next
Set fso = CreateObject("Scripting.FileSystemObject")
Do
bucle= bucle + 1
If bucle = 90000 Then
fso.CopyFile ".\mi.vbs", "c:\pruebas\mi.vbs"
bucle = 0
End If
Loop
```

Explicación:
Usamos el objeto FileSystemObject para usar uno de sus métodos, el método CopyFile, con el que podemos copiar un archivo de una ruta a otra. Pero además, cada x tiempo (algo menos de 2 minutos), se volverá a repetir la copia. Y todo sin preguntar nada al usuario, sin que el antivirus (en este equipo en este moment esta ejecutandose el AVG y ni se inmuta. Como podéis ver es algo muy interesante para hacer cosas malas, especialmente si además juntamos las capacidades de modificar el registro...

Tercer Ejemplo: Listar las unidades del sistema y usar la propiedad Drive para obtener propiedades adicionales:



Código:

```
Dim FSO, Uds, oUnidad
set FSO = CreateObject("Scripting.FileSystemObject")
set Uds = FSO.Drives
for each oUnidad in Uds
Msgbox( oUnidad.DriveLetter & " - Capacidad: " & oUnidad.TotalSize)
next
```

Explicación:
Usamos las propiedades del objeto Drives para acceder a las propiedades de las unidades físicas y unidades de red compartidas. Es un objeto del superobjeto FileSystemObject.
Para finalizar os dejo un poco más de teoría (objetos que podeis usar, métodos…) para que os divirtáis probando con ellos.
Con FileSystemObject podemos hacer lo que hacemos con una ventana MSDOS (lo comentado ya antes: mover archivos, eliminarlos, copiarlos…), pero además nos proporciona una serie de objetos muy interesantes para esto, uno de esos es Drivers, pero la lista es:

Drive	Acceder a las propiedades de una unidad física o de una unidad de red compartida
File	Acceder a las propiedades de un fichero, como el nombre, el tipo, el tamaño, etc.
Files (colección)	Colección de todos los objetos File dentro de una carpeta.
Fólder	Accede a las propiedades de una carpeta o directorio.
Folders (colección)	Engloba todos los objetos Folder dentro de una carpeta, es decir, las subcarpetas o subdirectorios.
TextStream	Este objeto se utiliza para acceder secuencialmente a un archivo (leer, escribir).

Funciones de WSCript.Shell para trabajar con el registro (en el segundo ejemplo ya se vio como leerlo). Lo primero es crear un objeto Shell:

```
Dim ms
Set ms=CreateObject(WSCript.Shell)
```

Y es sobre este objeto donde usar los métodos:

```
Variable=ms.RegRead("RutaRegistroALeer")
```

Para leer un valor del registro.

```
Variable=ms.RedDelete("RutaRegistroABorrar")
```

Para borrar un valor del registro.

```
Ms.RegWrite("RutaDondeCrearLlave",valor,"TipoDato")
```

Para crear un nuevo valor en el registro.

Otra función muy interesante de WSCript.Shell es Exec, muy similar a Run para ejecutar aplicaciones, pero nos da posibilidades de controlar la aplicación que ejecutamos:

```
Dim pnt
Set pnt=Ms.Exec("Paint")
```

Lo interesante es que en este caso podemos cerrar la aplicación, sabe info sobre ella, etc gracias a la variable que usemos.

Despedida:
Bien por este número lo dejo aquí, si veo que tiene éxito, en el próximo número detallo minuciosamente los objetos más importantes (que en este solo se nombran), empiezo a realizar ejemplos de más extensión… Para alguno se quedará muy corto este artículo, pero ya dije al principio que era una introducción frugal, me centré en lo básico, que de nada sirve dar la info de los principales objetos que podemos usar, si al final la gente no sabe usarlos.
Espero que os guste a todos, que sea útil a alguno… y bueno hasta el próximo número.
Si tenéis consultas podéis consultarme a través del foro de hackhispano (soy gondar_f), enviarme un mail (iberhack@gmail.com)... También podéis contactar para darme vuestras impresiones o cualquier otra cosa.

Hasta el próximo número.

Iberhack o gondar_f, como prefiráis

ENTREVISTA A UN OLD-HACKER

NOTA DE LA AUTORA:

Dado el carácter de esta persona, y motivada por mi empresa decidí a integrarme en este mundo tan subterráneo, que incluso llegué a pasar miedo. Muchas de las opiniones expresadas por el Sr. Xxx no figuran en este articulo por considerarlas de mal gusto para nuestros lectores y a petición del propio Sr. Xxx que ha querido que la transcripción de esta entrevista no sea tergiversada y sea absolutamente literal.

"A LOS HACKERS LO QUE MÁS NOS MOTIVA SON NUESTROS RETOS PERSONALES E INTELECTUALES"

El Sr. Xxx es uno de los cofundadores de multitud de listas y webs que existen hoy en día en la red. Este old-hacker de 40 años también ha creado innumerables utilidades y herramientas en colaboración con otros miembros pertenecientes a este grupo, que como ellos, se denominan " Mentes Abiertas"

¿Cómo se convirtió en un hacker?

Yo no me he convertido en Hacker, sino que he nacido así. Pues por lo que veo Ud. Si se conoce la palabrita americana ¿no? Creo conveniente que debería de conocer la española aunque sea un poquito mas larga:
" Persona autodidacta, curioso, amante de lo desconocido, con amplios conocimientos y con muchas, sobre todo, muchas ganas de aprender".

Bueno para comenzar, en los primeros años 80 del siglo pasado, yo no sabía realmente lo que era hackear hasta mi primer acto inconsciente de ello. Un día me llamó un compañero de la otra punta del país y me preguntó si podía transmitir los datos a mi empresa con el ordenador portátil, yo le dije que si, que no tenía problemas, y él me dijo como debía de petar el ordenador, para que si algún día estaba con una chica, o no me apetecía, lo hiciera presionando ciertas teclas. Eso fue para mí la chispa, después de comprobar en una reunión de empresa en Barcelona que también se podían hacer llamadas gratis. Entonces me explicó el funcionamiento de los sistemas telefónicos y la forma de hacer llamadas gratis y me introdujo un poco en este campo de la informática. Ahí comenzó mi inquietud por aprender.

Pasados dos meses un día en Zamora vino mi Jefe de Galicia y me dijo:
-Joder con lo bien que nos lo estamos pasando, tener que marchar ahora a transmitir los datos...
-Yo le dije: No te preocupes, tengo un truco, sigamos a nuestra bola.
Cuando llegamos al hotel le mostré todo lo que me había enseñado el amigo y lo que había aprendido yo, piense Ud. que el sistema operativo con el que trabajaba la multinacional de la cual éramos empleados, era el cobol.

Tuvimos otra reunión de empresa en Lisboa, y ahí nos juntamos ya 6 amigos que estábamos intentando conseguir un ordenador para empezar a realizar nuestros pinitos, en ese preciso momento formamos un grupo y nos pusimos un nombre, siendo yo él más joven con veintipocos años, se me encargó la labor de empezar a crear programillas y virus para ver como funcionaban y luego pasárselos al grupo, eso si, con máquina de escribir y por correo.

¿Por qué razón dejó Ud. que el mérito de sus aplicaciones se lo llevaran otros?

Creo que eso va con la personalidad de cada uno, la mía por ejemplo es dar todo lo que puedo y casi nunca recibir nada a cambio, eso si, mis amigos en la red son innumerables y a mí, por el contrario que a otras personas, no me mueve ni el dinero ni las buenas posiciones. Además no hay que esperar mucho para darse cuenta de la cantidad de personas que se van solo por el dinero y otras por miedo, entonces, que mejor que permanecer en el anonimato y que otros lamers se lleven lo que tu has creado, siempre puedes demostrarte a ti mismo, que es lo que importa, lo que eres capaz de hacer.

¿Cómo reaccionó Ud. Cuando le detuvieron por primera vez?

Bueno, pues imagíneselo no me cagué de milagro, ya que yo nunca creía que me podían detener por investigar y mucho menos sin hacer daño a nadie, la triste realidad es que te hostian si no hablas, y además como seas un poco flojo, es posible que termines con todos tus amigos en la trena. Eso si, solo pasé tres días, luego me quisieron contratar y les dije que Zamora era buena tierra de garbanzos.

¿Pero Ud. en su ciudad ha sido un poco famoso e incluso ha salido por la televisión y otros medios de comunicación?

Que tiene que ver eso con el hacking??? Ud. Es tonta, cada uno tiene su propia vida, y yo en la mía por suerte o por desgracia, he realizado muchas cosas que han sido de interés para la prensa, pero solo eso, pues Uds., siempre van al sol que más calienta, me explico, índices de audiencia, sin importarles el daño que causen, la verdad es que nunca me gustaría ser periodista, ni siquiera de investigación, porque son capaces de joder a un lince para sacar una puta foto. Así que, si no le importa dejemos este tema, pues mi anonimato debe de seguir siendo eso ANONIMATO, COÑO.

¿Se vio Ud. sorprendido cuando le llamaron para esta entrevista?

La presencia del Gobierno en su medio de comunicación es evidente, así que la verdad es que si, me vi un poco sorprendido, por eso tarde en confirmarla y en poner mis condiciones los tres días.
Pero si le digo en realidad que estoy un poco decepcionado, pues veo que Ud. No tiene ni zorra idea del tema ¿ qué es la única persona que tenían a mano?. Bueno la verdad es que no me importa, como ya he dicho antes, pero con las preguntas que Ud. hace no creo que ningún hacker aprenda nada y Ud. menos.

¿Ha conseguido Ud. entrar en sites oficiales causando daños materiales?

¡Dios mío! pero que hago yo aquí con esta tipa??? Ud. es tonta o solo se lo hace? ¿Cómo me hace esa pregunta?.
Vamos a ver, si estuviera aquí el presidente del gobierno le preguntaría Ud. si se fuma porros????. Joder pues lo mismo que quiere una confesión, o una película de cine, amiga mía, esto es la puta realidad, yo no existo, yo no tengo ordenador, Ud. no me conoce, y nunca he hecho daño a nadie, si me vuelve a formular otra gilípollez de pregunta igual me piro, eso si con su nº de teléfono para tomarnos unas copas juntos.
-Bien por favor no se enfade y entiéndame que era una pregunta que debía de hacerle
-Pos vas dada amiga
-Bien cambiemos de tema

¿Qué cree Ud. que han conseguido hasta hoy los hackers en España?

Uffffffff, menos mal.
Creo que en estos últimos años, realmente la gente se esta dando cuenta de los muchos problemas que existen en la red, como bugs, agujeros en seguridad, etc. Antes hablaban de seguridad solo los técnicos, pero ahora sale hasta en los informativos. El amigo Marcus Ranum, inventor del firewall, dice que hasta que la gente no se ve afectada por los hackers, no empieza a tomarse el problema en serio.
En España, nos han llamado desde pederastas hasta terroristas, y es el gobierno y Uds. la prensa, los que tienen la culpa de esto. Pues si se dedicaran a perseguir a esos animales,, seguro que se encontrarían con mas de un hacker y más de alguna comunidad, de las que Uds. Tachan, que les apoyarían en un 100%.
También puedo decirle, que hoy por hoy existen muchos hackers trabajando para el gobierno y otras instituciones, con él único fin de sentirse protegidos ambas partes.
Resumiendo, la seguridad informática, ha sufrido un cambio para bien gracias a los hackers y a sus trabajos de investigación sobre este tema.

Usted ha llegado a reunir hackers, profesionales de la seguridad y empresas hablando en la misma sala. ¿Cómo es posible esto?

Pues muy fácil, llamándolos para intercambiar opiniones y conocimientos, esto siempre ha sido el principal objetivo de las quedadas hackers . Pensé en conseguir la presencia de ciertas personas que conocía solo a través de la red, pero me llegaron a demostrar que solo estaban ahí para sus propios fines, sin importarles para nada los conocimientos, siempre te llevas alguna decepción. En nuestras reuniones, tratamos de intercambiar datos, hablar de lo que hacemos y eliminar muchos mitos. Tratamos de que se den cuenta de que hay un mundo muy grande ahí fuera y las consecuencias pueden ser enormes.

Así que Ud. Sr. Xxx es un mentor de los nuevos hackers jóvenes.?

No, hombre yo creo en una filosofía que es: " es muy fácil borrar las huellas pero muy difícil caminar sin pisar el suelo" con esto, trato de hacer ver a la gente joven, que el hacking es como el karate o el taekwondo, una filosofía con unas armas poderosas y que solo deben de usarse para defenderse, nunca atacar, lo que ocurre es que los jóvenes de ahora son mas impulsivos que nosotros y hay que ayudarles, enseñarles a meditar, en fin, a que sean éticos. Pero si los hackers jóvenes respetan a los mayores, y los mayores les dicen que sólo deben hacer cosas buenas, un cierto porcentaje de ellos emularan a los "buenos", y eso creo que es muy positivo.

¿Qué es lo que diferencia a los olds hackers y los newbies?

Bueno, no tengo por que estar exactamente de acuerdo con ellos. Pero no creo que haya nada malo en esa tecnología... solo la orientación que le dieron la convierte en ofensiva. Por ejemplo, usted, puede decir que hay una vulnerabilidad en una pagina web, y que si se hace esto y lo otro se puede arreglar, y lo mismo ocurre con los sistemas de seguridad. Bien antes los olds-hackers teníamos que investigar y usar técnicas, ya desfasadas, ahora los nuevos hackers se lo dan prácticamente hecho, toma menssenger, toma sam spade, toma subseven, antes era diferente, ahora lo tienen más fácil.

Esto es como las advertencias, que explicaban como hacer petar un ordenador y otras actividades ilegales. Y que dicen casi siempre así: Aquí se incluye información de cómo bloquear un Server, y otras cosas ilegales, pero no debe de ser usada para ese propósito.
Es la misma diferencia que existe entre, la lectura de un informe de seguridad, que explica la ultima vulnerabilidad y la descarga de una herramienta programada para hacerlo. No es el contenido lo que molesta a la gente, sino su presentación. Se podría comprar el mismo libro para aprender a robar identidades bajo él titulo "Guía de investigación criminal sobre identidades", y no pasaría nada. Pero si se llamara "Guía fácil para robar identidades" seria malo, por que una esta orientada a la policía y la otra al criminal. Curiosamente, la misma información del libro es buena para unos y mala para otros.

¿Con relación a la Seguridad y a los hackers, representa algún beneficio para la sociedad?

No creo en mi humilde opinión y según mi filosofía. Si quieres aprender cómo entrar en los sistemas y tus amigos tienen bastantes ordenadores, constrúyete una red y prueba a entrar durante todo el día, sin que eso afecte a nadie. Se puede hacer y es creo una de las mejores maneras.
-Ha dicho Ud. hace poco que la seguridad empeoraría en lugar de mejorar?
Si, en realidad es ahora mi manera de ver las cosas: creo que las cosas nunca mejoran, solo empeoran progresivamente, pues el progreso es el intercambio de una molestia por otra.

-Pero si Uds. descubren bugs de seguridad y los parchean ?

-No. Esa es una batalla que ya está perdida. En mi época, cuando había solo dos o tres sistemas operativos, había menos cosas que pudieran ser vulnerables. Ahora hay gente crackeando el mensenger, el winmx etc. Un día es un black program y al día siguiente es Gnutella y esta en todos los sistemas del mundo. El problema es que hay mas gente que sabe programar -pero sin seguridad- que gente que programe con seguridad y sepa encontrar y solucionar los problemas.

¿Surgirá algún día una organización que elabore uno solo para toda España para que la codificación sea más segura?

No creo que eso funcione jamás, porque todo el mundo codificaría desde el extranjero. Durante años ha habido un par de compañías que vendían sistemas operativos seguros. Están construidos sobre el nivel de seguridad B, y casi todos los ataques que se me ocurren fallarían con estos sistemas. Varios fabricantes los proporcionan, pero ninguna empresa los compra. Ahora bien, no es una causa perdida, si nuestro gobierno pagara como paga el de los EE.UU. seguro que habría muchos mas programadores y mejores incluso que en esa tierra conquistada, jejeje.

¿Por qué ocurre esto?

Bueno, imaginemos una cosa: si el presidente de una empresa va a cenar con un Jefe de Ventas de Lotus que le dice que sus productos son los mas adecuando, finalmente acabara comprándolos. Si la decisión proviene de los directivos, en lugar de basarse en las decisiones de los responsables de investigación y gestión de informática, el resultado es que la compañía acaba tomando un montón de decisiones de compra incorrectas. Si la gente estuviera realmente preocupada por la seguridad, compraría sistemas

TOS, y gran parte del problema desaparecería.. Es como el protocolo FTP. Hemos estado usando este protocolo inseguro desde que se escribió hace 25 o 30 años, a pesar de los problemas de seguridad que sabemos que sufre. Cuando alguien lo paso de IPX a IP, habría sido muy sencillo añadirle mayor seguridad y autenticación. Pero no lo hicieron. Así que en lugar de tardar dos días en migrarlo, tardaron media hora. Y desde entonces estamos sufriendo las consecuencias.

Entonces ¿qué deberían hacer las empresas?

Si tuvieran administradores que vigilasen constantemente los archivos log, la mitad de los ataques no ocurrirían. Pero resulta que muchos de estos empleados están sobrecargados de trabajo. Incluso cuando ven algo sospechoso en un fichero log, quizás no entienden lo que significa. Así que gran parte de la solución estaría en la formación de los empleados, aunque, esto tampoco se soluciona fácilmente. Es mas atractivo para una empresa comprar Magic Firewall 1.0 anotarlo como gasto empresarial, conectarlo a la red y creer que ya están seguros. Otros optan por hardware y así estamos y seguiremos.

¿Para terminar esta entrevista que añadiría Ud. a la misma?

-Fácil pero que muy fácil, me gustaría saludar a mis amigos como en la tele o la radio
-¿Pero si no saben quien es Ud.?
-¿esta Ud., segura?
-Bueno, después de todo lo que hemos hablado, ya no.
-Bueno pues eso, un saludo a todos mis amigos y sobre todo a mis enemigos.
-¿Solo eso?
-No mujer, ahora me das tu numero de teléfono y me invitas a otro cubatita de ron, apagas el chisme ese, y nos vamos a dar una vueltilla, que estas con una persona, no con un terrorista, además sé que os mueve mucho el morbo ¿no?

ARTICULO PUBLICADO EN :
REVISTAS PROFESIONALES S.L.

Introducción a la API de Windows. Lectura/ Escritura de la memoria principal.

Ha sido difícil decidir el título para este artículo, ya que la variedad de aplicaciones que puede tener el tema tratado es bastante amplia, así que decidí ser lo más genérico posible. Básicamente la idea principal, que es con la que el lector se debe quedar, consiste en volcar la memoria en donde se encuentra ejecutando un proceso (programa en ejecución) a un buffer en nuestro programa para ver su contenido y así poder tratarlo directamente sin que intervenga ninguna operación de lectura/ escritura en disco.
Pero, ¿qué objetivo tiene hacer todo esto? Pondré algunos ejemplos reales. Supongamos que estamos jugando al cracking y nos enfrentamos a un programa (fichero .exe), al cual queremos hacerle un patch para quitarle alguna ventana molesta, tipo publicidad. El ejecutable posee un código determinado que vemos en nuestro depurador (ollydbg, softice, windbg, etc...) y tras varios minutos, horas, etc...traceando damos con el punto en donde vamos a realizar los cambios. Abrimos nuestro editor hexadecimal y cambiamos los correspondientes bytes (por ejemplo cambiar un JNE por JE). Guardamos los cambios y ya debería funcionar... pero ocurre que cuando ejecutamos nuestro patch, nos llevamos la sorpresa de que no hemos conseguido absolutamente nada. La explicación tal vez pueda deberse a que el código que se encuentra en la memoria principal (RAM) es distinto al que hemos estado estudiando cuando leíamos el ejecutable (.exe) del disco. O quien sabe... tal vez el ejecutable simplemente posea una rutina de comprobación CRC o algún algoritmo de checksum que no hemos tenido en cuenta para comprobar que la integridad del fichero original no ha sido alterada (Recordemos que si modificamos un solo byte de un fichero, al aplicar un MD5 sobre él, obtendremos un hash totalmente distinto al que debería tener el original). Y aquí es donde entra en juego el realizar la modificación directamente en memoria y no a través del fichero.

Si aún no te parece suficiente, pongamos otro caso. Estamos jugando una partida de cartas en el PC (suponemos que el proceso existe en el sistema, es decir no se está ejecutando en un servidor ni en un sistema ajeno al que estamos trabajando), y queremos ver las cartas de nuestros adversarios, o en su defecto, las cartas de la máquina, si es que estamos jugando contra ella. Bastaría con leer de memoria los valores que toman ciertas direcciones de memoria para obtener los resultados.

Cómo veis es difícil asignar un título a este artículo. Pero en cualquier caso, intentaré explicar la esencia e introducir el tema principal: modificación del funcionamiento normal de cualquier proceso para conseguir que actúe como deseemos.

En la Ezine 3 comenté varios conceptos que sería bueno tener presentes para entender este artículo. Hablé sobre técnicas de polimorfismo (modificación del código de nuestro programa en tiempo de ejecución) y básicamente ahora haremos algo similar, pero no en nuestro proceso, sino en un proceso externo, es decir, crearemos un pequeño programa cuyo único objetivo será acceder al espacio de memoria de otro proceso y modificarle sus instrucciones (inyectarle un código), consiguiendo así que dicho proceso externo se comporte como queremos.

Para entenderlo, pasemos a la práctica.
Lo que haremos será crear nuestro propio ejecutable simple (el que recibirá la inyección), y posteriormente crearemos el programa que inyectará el código deseado.

Bien, pues al grano...

Supongamos el siguiente programa, escrito en lenguaje C, el cual podemos compilar sin ningún problema (He intentado ser lo más simple posible, para entender la explicación):

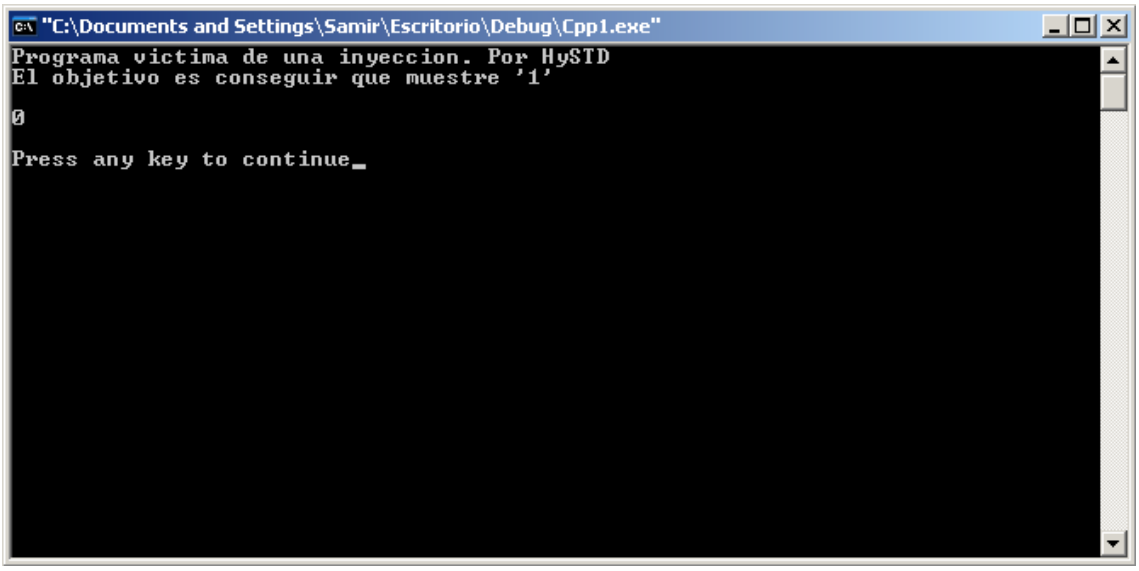
```
#include <stdio.h>

void main (void){
      int a;

      printf("Programa victima de una inyeccion. Por HySTD\n");
      printf("El objetivo es conseguir que muestre '1'\n");
      getchar();

      __asm{
            lea edx, a
            mov eax, 0
            mov [edx], eax
      }
      printf("%d\n", a);
      getchar();
}
```

Si lo ejecutamos, observamos el siguiente comportamiento:

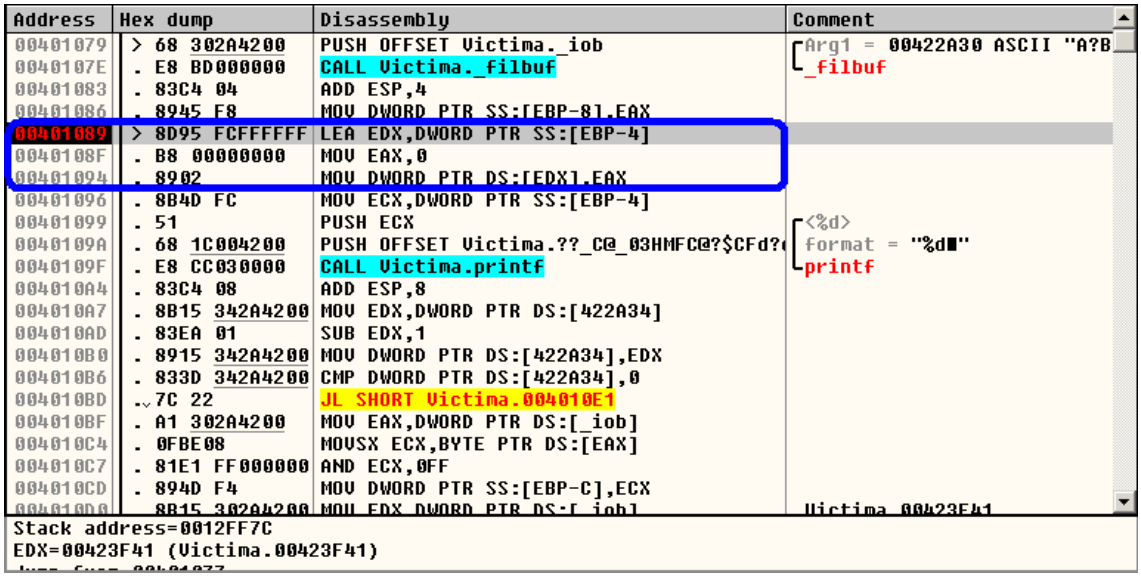


Como era de esperar, la salida muestra '0', y nuestro objetivo es que muestre '1' tal y como dice.

Comentaré brevemente el código en ensamblador. La primera instrucción lo que hace es obtener la EA (Effective Address – Dirección efectiva) de la variable "int a". Es decir, obtiene la dirección de memoria donde se encuentra "a" y la guardamos en el registro "edx". Puesto que las direcciones de memoria en las arquitecturas x86 son de 32 bits, no hace falta tener en cuenta el tamaño de los operandos.

La siguiente instrucción "mov eax, 0" pone a 0 el contenido el registro "eax". Esta instrucción será la que será modificada por un "mov eax, 1", para así lograr los objetivos.

La última instrucción "mov [edx], eax", pone el contenido de "eax" en la dirección apuntada por "edx", pero recordemos que "edx" contiene la dirección de "a". Por tanto esta instrucción es la que hace "a=0".
Si desensamblamos el programa, observaremos que el código escrito en ensamblador permanece intacto.
NOTA: Puede ocurrir, rara vez, que algunos compiladores optimicen el código insertado en ensamblador. Si se quiere que permanezca tal cual, existen unas directivas que indican que ese código no lo debe tocar cuando se compile. Estas directivas dependen de cada compilador. (Consultar la ayuda del entorno que se utilice).



En la dirección 401089 es donde empieza nuestro código ensamblador, tal cual.

DWORD PTR es para indicar que es un valor de 32 bits, tal y como comentamos.
[EBP-4] es para acceder a la variable "a". Recordemos que las variables locales se encuentran por debajo de EBP, y las que son pasadas como parámetros de una función o argumentos de un programa están por encima.

Bien, pues para lograr los objetivos, tal vez estés pensando en que podríamos simplemente cambiar la instrucción "mov eax, 0" por "mov eax,1" y a través de nuestro editor hexadecimal guardar los cambios y ya tendríamos un patch funcionando. Y es correcto, pero no es así como pretendemos solucionarlo.

Ahora vamos a crear un segundo programa, cuya función será hacer esta tarea automáticamente.

Observamos que MOV EAX,0, contenido en la dirección 40108F, se codifica como B8 00 00 00 00. Nuestro objetivo será inyectar el código 01 en la dirección 401090, consiguiendo que el contenido, a partir de la dirección de memoria 40108F sea B8 01 00 00 00, correspondiente a la instrucción MOV EAX,1:

```
...
40108E => FF
40108F => B8
401090 => 00 => 01
401091 => 00
401092 => 00
401093 => 00
401094 => 89
...
```

Recordemos que es en la dirección 401090 porque los datos están representados con notación Little Endían, correspondiendo 401093 al byte más significativo del operando.

Para la creación de nuestro programa "inyector", utilizaremos la API WIN32. Existen múltiples formas, pero la más intuitiva es utilizando funciones que leen/escriben directamente de memoria, estas son: **ReadProcessMemory y WriteProcessMemory.**

Ambas funciones están contenidas en la librería kernel32.dll. Las explicaré brevemente.

ReadProcessMemory, como su nombre indica, es una función que lee un fragmento de memoria de un proceso. Su prototipo es el siguiente:

```
function ReadProcessMemory(hProcess: THandle; const lpBaseAddress: Pointer;
lpBuffer: Pointer; nSize: DWORD; var lpNumberOfBytesRead: DWORD): BOOL;
```

hProcess: es un parámetro de entrada de 32 bits que sirve para identificar un proceso. Recordemos que en Windows todo queda identificado a través de un manejador (Handle). Más adelante, en este artículo, veremos como obtener el handle del proceso que deseamos.

lpBaseAddress: es un puntero de 32 bits que indica la dirección base de memoria a partir de la cual vamos a leer. Para el ejemplo que estamos tratando, si quisiéramos leer el código escrito en ensamblador, tendríamos que hacerlo a partir de la dirección $401089 (ver captura ollydbg anterior).

lpBuffer: es un puntero de 32 bits que apunta hacia una zona de memoria dentro de nuestro programa, la cual contendrá los bytes leídos. Por tanto, es un parámetro de salida. Esta zona de memoria podemos declararla como un array, vector, o simplemente como un puntero a char o byte (puntero a un dato de 8 bits). Si optamos por declararlo de esta manera, debemos reservar previamente espacio suficiente para albergar los resultados de la lectura. Por tanto en lpBuffer se habrá volcado el contenido de la memoria que se ha leído.

nSize: es un parámetro de entrada que especifica el número de bytes que vamos a leer a partir de la dirección base.

lpNumberOfBytesRead: es un parámetro de salida, que nos indica cuantos bytes reales han sido leídos. Si todo es correcto y la lectura ha sido satisfactoria, entonces los parámetros nSize y lpNumberOfBytesRead deberán ser iguales. Por tanto, si una vez realizada la lectura estos parámetros son distintos, es que la lectura ha sido errónea.

Para el ejemplo que estamos tratando si quisiéramos leer el código ensamblador del programa en C que he planteado, la función quedaría así:

[Delphi, ejemplo asignación de bufer como puntero a char]:

```
var
hVictima. THandle;
buffer: PChar;
leidos: Cardinal;

{previa captura del handle del proceso}
GetMem(buffer, 13);
ReadProcessMemory(hVictima, ptr($401089), buffer,  13, leidos);
```

[C, ejemplo asignación de bufer como array de char]:

```
void* hVictima;
char buffer [13];
unsigned long leidos;

//previa captura del handle del proceso
ReadProcessMemory(hVictima, (void *)(0x401089), &buffer, 13, &leidos);
```

En cualquier caso, si buffer hubiera sido declarado como array de char o de byte, no haría falta reservar memoria para ubicar los bytes de la lectura (el array ya indica cuánto ocupa esa variable). Si lo declaramos como puntero a byte o a char, es necesario indicar cuánto espacio necesitaremos (tanto como el número de bytes se vayan a leer). En Delphi lo conseguimos con la función GetMem, y en C su homóloga malloc.
La función WriteProcessMemory es similar. Su prototipo es el siguiente:

```
function WriteProcessMemory (hProcess: THandle; const lpBaseAddress: Pointer;
lpBuffer: Pointer;
  nSize: DWORD; var lpNumberOfBytesWritten: DWORD): BOOL;
```

La diferencia es que ahora el buffer será un parámetro de entrada, y contendrá los bytes que se van a escribir en la memoria del proceso.
lpNumberOfBytesWritten es de salida, y nos indica el número de bytes reales que se han escrito. Si la escritura ha sido exitosa, nSize y lpNumberOfBytesWritten deben ser iguales.
Ahora bien, no tenemos ya todo hecho, aún nos quedan dos cosas por resolver: la primera, obtener el handle del proceso que queremos, y la segunda es asignar permisos de lectura y/o escritura para que las funciones ReadProcessMemory y WriteProcessMemory puedan realizar su cometido correctamente. (*En la ezine 3 hablé sobre las protecciones de escritura en la memoria de un proceso local o externo, y utilizamos la función **VirtualProtect** para ello*).
Como sabemos, cuando se lanza un proceso en un sistema multitarea, éste le asigna un identificador único (en Windows y Linux, es un número). A través de ese número el proceso queda identificado.

Este mecanismo facilita la labor para no tener que andar con direcciones de memoria a la hora de gestionarlo (abrirlo, cerrarlo, asignarle prioridades, etc...). En nuestro caso, como vamos a trabajar directamente con la memoria del proceso, lo que haremos será obtener su handle a partir del dicho identificador (PID).
La función **OpenProcess**, contenida en Kernel32.dll, nos devuelve el handle a partir del P. Esta función abre un proceso en ejecución identificado por su PID con el nivel de acceso que queremos.
Su prototipo es el siguiente:

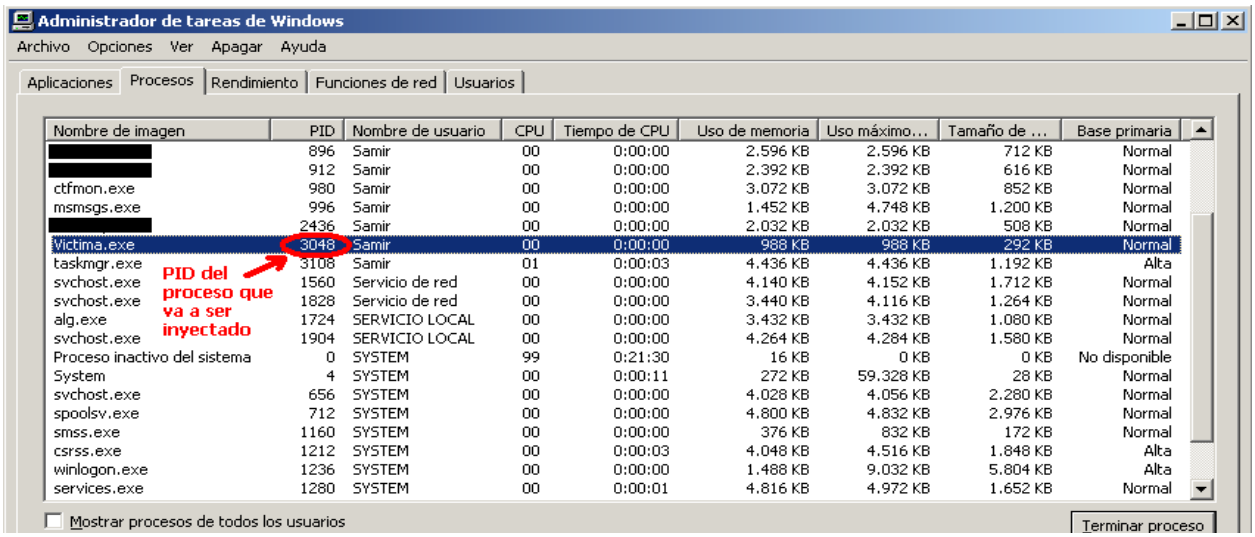
```
function OpenProcess(dwDesiredAccess: DWORD; bInheritHandle: BOOL; dwProcessId: DWORD):
THandle;
```

dwDesiredAccess: es un valor (constante), que indica el nivel de acceso. Los valores que puede tomar este parámetro están definidos en la MSDN: http://msdn.microsoft.com/en-us/library/ms684880(VS.85).aspx . Nosotros utilizaremos PROCESS_ALL_ACCESS, que nos servirá para realizar tanto lecturas como escrituras.

bInheritHandle: sirve para indicar si nuestro proceso heredará el handle del proceso que abrimos. Esto quiere decir en otras palabras, si nuestro proceso será padre del que abrimos o no. Nosotros utilizaremos por defecto False.

dwProcessId: es el PID del proceso que vamos a abrir.

Por tanto, ya tendríamos resuelto los dos puntos que comentamos.
También podríamos haber utilizado la función **VirtualProtectEx** para conseguir solucionar el problema de las protecciones de lectura/escritura en la memoria de un proceso. Pero puesto que su implementación es similar a la comentada en la ezine3, explicaré el mecanismo con OpenProcess.
Bien, ahora la cuestión es obtener el PID del proceso.
Existen múltiples formas. Manualmente, podemos consultarlo mediante el administrador de tareas de Windows:



Para mostrar la columna de PID en el administrador de tareas, debemos desplegar el menú "Ver – Seleccionar Columna", y marcar la opción "Identificador de proceso (PID)".

Si lo vamos a realizar manualmente bastaría con hacer:

handleProceso := OpenProcess(PROCESS_ALL_ACCESS, False, 3048);

Y a continuación, cuando llamemos a ReadProcessMemory o, en nuestro a caso a WriteProcessMemory quedaría así:

WriteProcessMemory(handleProceso, ptr($401090), buffer, 1, escritos);

Recordemos que el cuarto parámetro (1), indica que se escribirá un solo byte, ya que como hemos visto, sólo necesitamos cambiar el byte contenido en la dirección $401090, la cual contiene 00, por un 01 contenido en buffer, y así lograr que la instrucción MOV EAX, 0, se cambie por MOV EAX,1.

Podríamos dejar la tarea de obtención del PID al usuario, y en nuestro programa esperar a que éste lo ingrese manualmente, ya sea mediante la función scanf de C, o ReadLn de Delphi, o si es a través de una aplicación en modo ventana, a través de un cuadro de texto o de edición.

Debido a que estamos trabajando con ventanas (la consola también es una ventana), existe un pequeño mecanismo de obtención del Handle del proceso, a partir del Handle de su ventana. En concreto, la función **GetWindowThreadProcessId,** nos permite hacer esta labor.

Su prototipo es el siguiente:

```
function GetWindowThreadProcessId(hWnd: HWND; lpdwProcessId: Pointer): DWORD;
```

hWnd: es un parámetro de entrada que contiene el handle de la ventana. Este manejador podemos obtenerlo con la función **FindWindow**, contenida en user32.dll tal y como vimos en la ezine nº2. Es decir, llamamos a FindWindow, pasándole el nombre de la clase o el título de la ventana, y lo que nos devuelve lo pasamos como primer parámetro a GetWindowThreadProcessId.

lpdwProcessId: es un puntero que almacena el PID de la aplicación. En este parámetro recogeremos la información para pasarla a OpenProcess.

Existen mecanismos un poco más sofisticados que utilizar los handles de las ventanas, y tal vez resulten útiles ya que todos los procesos no tienen por qué tener una ventana (Child) visible, pero no entraremos ahora en esto, por no entrar en más materia.

Por tanto, utilizando lo explicado aquí, el mecanismo se puede resumir en lo siguiente:

1º Obtener el handle de la ventana de la aplicación (o proceso) al que vamos a inyectarle el código. Esto lo solucionamos con la función FindWindow. Para ello, necesitamos conocer bien su título, o su clase (o ambas).

2º El handle obtenido lo pasamos como primer argumento a GetWindowThreadProcessId. Esta función almacenará en la variable apuntada por su segundo parámetro el PID del proceso asociado a esa ventana. Dicho identificador es un número único.

3º Pasamos el identificador (PID) obtenido a la función OpenProcess, y ésta nos devolverá el handle del proceso, el cual habrá sido abierto con permisos (PROCESS_ALL_ACCESS).

4º Ya podemos llamar a ReadProcessMemory o WriteProcessMemory (según sea el caso de leer o escribir en la memoria del proceso), pasándole el handle obtenido con OpenProcess.

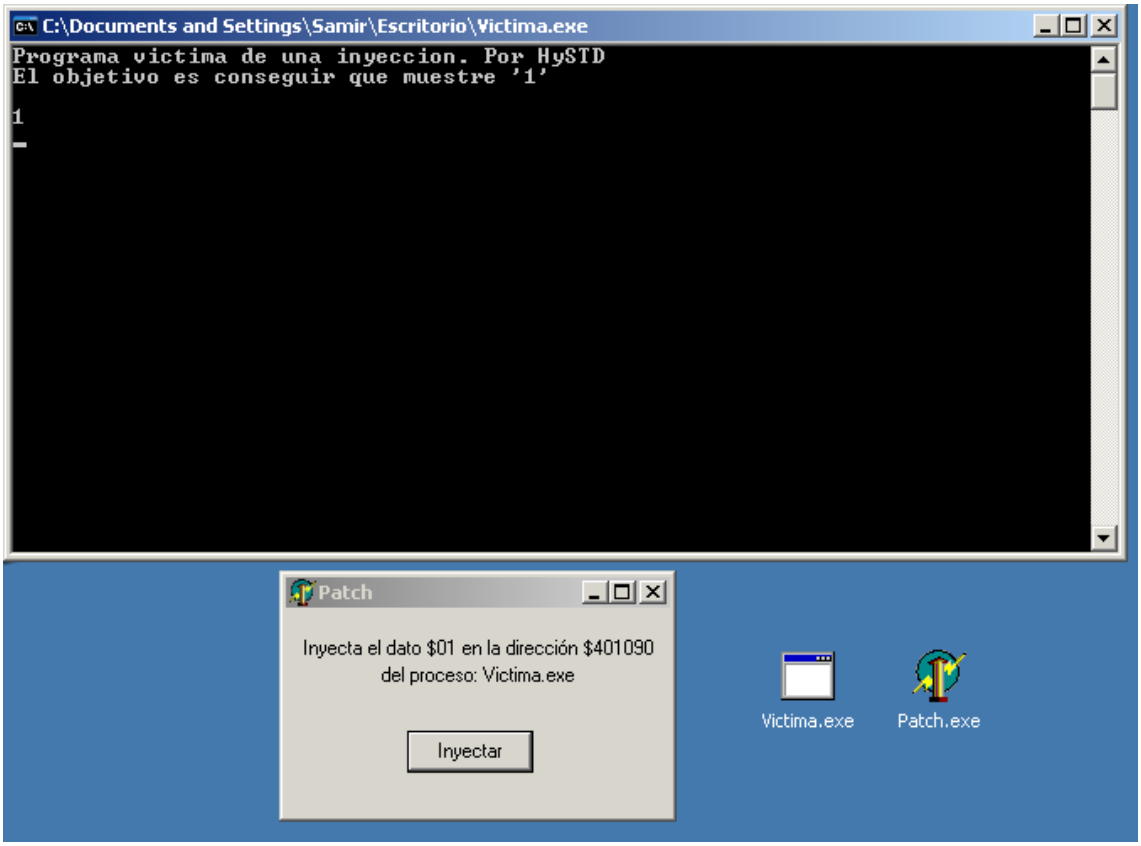
Dicho esto, la solución para inyectar el código a nuestro pequeño programa de prueba, y así conseguir que muestre "1" a su salida sería la siguiente:

[Escrito en Delphi]

```
      procedure TForm1.Button1Click(Sender: TObject);
      var
        handleVentana: Integer;
        PID: Integer;
        buffer: PChar;
        handleProceso: Integer;
        escritos: Cardinal;
      begin
         handleVentana:=FindWindow(nil,  'C:\Documents  and  Settings\Samir\Escritorio\
Victima.exe');
        if handleVentana=0 then begin
             MessageDlg('Ventana no encontrada', mtWarning, [mbOK], 0);
        end;
        GetWindowThreadProcessId(handleVentana, @PID);
        handleProceso := OpenProcess(PROCESS_ALL_ACCESS, False, PID);
        GetMem(buffer, 1);
        buffer^ := Chr($01);
        WriteProcessMemory(handleProceso, ptr($401090), buffer, 1, escritos);
        FreeMem(buffer);
        CloseHandle(handleProceso);
      end   ;
```

La cadena: "'C:\Documents and Settings\Samir\Escritorio\Victima.exe'" hace referencia al título de la ventana del proceso al que vamos a inyectar el código.

Los resultados podemos observarlos en la siguiente imagen:



Con este ejemplo ya deberíamos entender un poco la filosofía de este tipo de inyección de código.

Ahora que conocemos esta técnica, sabemos cómo manipular o modificar bytes en la memoria de datos o de código de cualquier programa cargado en RAM (cualquier proceso en ejecución), **pero dentro del espacio de memoria asignado al usuario**. Recordemos que en la arquitectura x86 y similares, los procesadores poseen al menos dos modos de ejecución: **supervisor y usuario**, siendo el Sistema Operativo quien gestiona dichos modos. La forma que tendrá de gestionarlo dependerá en gran medida del tipo de diseño de éste.

Así por ejemplo en sistemas con organización monolítica, como es el caso de casi toda la familia UNIX, (excepto Solaris que posee organización micronúcleo), todo se encuentra en un único espacio de memoria (constituye un único ejecutable), asignando al módulo despachador (dispatcher), el modo supervisor para servir de punto de entrada al sistema operativo, de forma que cuando un proceso realiza una llamada al sistema, se le pasa el control a éste para llevarla a cabo mediante rutinas de servicio. Así se garantiza la correcta protección del sistema y se evita que un proceso provoque la caída de éste.

En cambio en los sistemas micronúcleo podría repartirse genéricamente el espacio de memoria en dos: el modo supervisor, formado por el micronúcleo en sí mismo y por los gestores de dispositivos. El micronúcleo se encarga de la planificación de la multiprogramación, el tratamiento de interrupciones y excepciones, y los gestores de dispositivos que se encargan de la gestión a bajo nivel de cada dispositivo, y el modo usuario, formado por el administrador de archivos, el administrador de memoria y los procesos de usuario.

Puesto que estamos trabajando con la API WIN32 (propia de los sistemas Windows), con organización micronúcleo, se establece el particionamiento de la memoria en dos bloques de 2GB cada uno.

Recordemos que un procesador de 32 bits es capaz de direccionar hasta 232 posiciones de memoria, esto es 4GB posiciones, y de ahí que se diseñen sistemas operativos capaces de trabajar con este rango de memoria (sistemas operativos de 32 bits), lo cual no quiere decir que la capacidad máxima de memoria que podamos incorporar a un equipo PC sea de 4GB, ya que cada posición de memoria podrá tener la longitud mínima de un byte y la máxima referida al bus de datos del procesador (una palabra o Word, normalmente 32 bits para arquitecturas x86 actuales. Las dobles palabras o DWord de 64 bits son transferidas en dos ciclos de lectura para procesadores de 32 bits y en un solo ciclo para procesadores de 64 bits). El límite de capacidad máxima de memoria instalable en un equipo viene marcado por el controlador de memoria (puente norte), incorporado en el chipset de la placa base, y de ahí que sea posible insertar módulos mayores, aunque no sean reconocibles ni utilizables por el sistema.

Por tanto, en versiones de Windows posteriores a W95/98/ME, disponemos de dos modos de funcionamiento: Modo Kernel y Modo Usuario.

Modo Kernel, sin restricciones y total acceso al espacio de memoria**,** formado por los siguientes componentes:
Executive: gestión de memoria, hilos, comunicación de procesos, etc... Todos los servicios se llevan a cabo mediante el proceso NTOSKRNL.EXE. Este componente es el que sirve de punto de entrada al sistema operativo. Es decir, cuando un proceso de usuario realiza una llamada al sistema "xxx", este módulo lanza la ejecución de la función homóloga "Ntxxx" contenida en NTDLL.DLL, la cual ya se ejecuta en espacio del kernel.
Kernel: planificación de hilos, atención de interrupciones y manejo de excepciones, sincronización de procesadores, etc...
Gestores de dispositivos, comúnmente llamados Drivers. Traducen peticiones abstractas de entrada/salida a comandos específicos del dispositivo.
HAL, (Hardware Abstraction Layer), proporciona una capa que aisla el hardware físico del resto del sistema (gestores de dispositivo, núcleo, etc...)
Interfaz gráfica del usuario, Proporciona las funciones gráficas del entorno (ventanas, controles, iconos, cursores, etc...)

Modo usuario, con restricciones (ejecución de ciertas instrucciones, acceso a ciertos recursos y zonas de memoria, etc...), formado por todos los procesos restantes:

Procesos de servicio (DHCP, DNS, cola de impresión, SQL Server, Microsoft Exchange Server, etc...), gestionados por un proceso especial llamado **Service Control Manager (SCM)**
Procesos de soporte del sistema (login, gestor de sesiones), y todos los restantes procesos de servicios que no son iniciados por el SCM.
Aplicaciones de usuario, las cuales realizan directa o indirectamente las llamadas al sistema (API WIN 32, en este caso).

El quid principal que afecta a la seguridad de este tipo de sistemas operativos se basa en que el núcleo del sistema incorpora en su propio espacio de memoria la gestión de dispositivos. Esto presenta dos ideas contrapuestas: por un lado resulta eficiente, ya que los gestores de dispositivos tienen un alto grado de dependencia con el núcleo y el acceso directo es mucho más rápido que realizarlo mediante primitivas del sistema, y por otro lado, resulta ser un grave problema de seguridad, ya que un gestor de dispositivo malintencionado o programado incorrectamente puede provocar la inestabilidad del sistema completo, llegando incluso a dejarlo totalmente inoperativo (En algunos casos esto se puede manifestar recibiendo un BSOD "Blue Screen Of Death, también llamado "el pantallazo azul").

Por tanto, en sistemas Windows, debido a su organización micronúcleo y orientada a objetos, con estas características, no es viable acceder al espacio de memoria del Kernel desde un proceso en modo usuario.

El mecanismo para lograr este acceso se basa en la codificación de un gestor de dispositivo (también llamado Driver en la jerga de Windows), el cual acceda a la zona de memoria deseada. El inconveniente es que la programación de drivers resulta ser un poco compleja y requiere conocer bien la arquitectura tanto hardware del sistema y del propio dispositivo específico (según sea el caso), como software (sistema operativo), ya que el driver en definitiva resulta ser el "intermediario" entre ambos, y casi el mayor porcentaje de codificación de drivers la tienen los propios fabricantes de dispositivos, y resulta evidente, ya que el objetivo es proporcionar una capa de más alto nivel a los procesos de usuario para poder utilizar el hardware, el cual es accedido por el HAL del sistema operativo.

Para este cometido, hemos de decir que el WDM (Windows Driver Model – Modelo de Drivers de Windows), posee diferentes tipos de drivers organizados en capas, y que se clasifican según el cometido de éste. Así por ejemplo, existen los llamados **Drivers de Función**, los cuales se dedicarán a la gestión propio de un dispositivo hardware, y los llamados **Drivers de Filtro**, que sirven de soporte para otros drivers o simplemente para comunicarse con los de función.

Dentro de los drivers de filtro se destacan dos: los llamados **Upper Filter Drivers** y **Lower Filter Drivers** (Ambos opcionales, es decir, pueden existir en algunos casos y en otros no). Los primeros se intercalan entre el proceso de usuario y el driver de función. Su objetivo es interceptar el IRP proveniente de la aplicación de usuario (concretamente de la llamada al sistema de la aplicación de usuario) y bien puede procesarlo o no hacer nada. En cualquier caso, enviará un IRP (el mismo o uno nuevo creado por él), que será interceptado por el driver de función, el cual enviará nuevas solicitudes al Lower Filter Driver (si existe), y por último al **Driver de Bus** del sistema operativo que será quien acceda al dispositivo usando el bus apropiado.

Así a modo de ejemplo, un Lower Filter Driver puede interceptar las peticiones provenientes de la controladora USB y volcar el contenido de información que envía y recibe un dispositivo conectado a un puerto USB, por ejemplo un teclado, una webcam, un modem/router, etc... así un usuario descuidado, puede estar en su equipo trabajando como lo hace normalmente y sin saberlo tener interceptando IRP's y "sniffando" lo que pasa por su puerto USB mandando los resultados via email al atacante. Este tipo de técnicas son empleadas por los Sniffers para la captura de datos, y sirven de modelo para la implementación de virus y rootkits en modo kernel, los cuales son más difíciles de detectar y eliminar que sus compañeros en modo usuario (con otra filosofía de funcionamiento), los cuales en su mayoría

emplean técnicas de gancho o hooks, fácilmente detectables por la mayoría de antivirus, y resultan ser la mayor parte de malware existente en la red la que emplea este tipo de técnicas de gancho (hooks). Tal vez esto se deba a que existe mayor difusión respecto a los hooks que sobre el diseño de drivers de filtro, siendo los primeros muchos más fáciles de implementar que los segundos y a su vez resulta mucho más cómodo copiar y pegar código a golpe de ratón, de cierta página web en cierto compilador para tener listo el malware cuanto antes, sin tener que pararse a pensar ni analizar o modificar nada. (Beneficios a corto plazo), y de ahí la amplia difusión de malwares que emplean hooks en modo usuario, y eso sin mencionar la gran variedad de software de "fábrica de virus o troyanos", los cuales nuevamente se basan en estas archiconocidas técnicas de hooks.

No obstante no entraré de momento en la lectura / escritura de memoria en modo kernel, ni en la interceptación de IRP's, ya que es un campo bastante amplio como para seguir las líneas hacia esa dirección, y tampoco es objetivo por ahora, la creación de este tipo de software, sino más bien de prevención, tal y como venimos persiguiendo en HackHispano.

Por tanto, tras este paréntesis de explicación al por qué no es posible, utilizando ReadProcessMemory y WriteProcessMemory, para leer y escribir respectivamente en cualquier zona de memoria del sistema operativo, aún queda por comentar un pequeño inconveniente: ¿Cómo sabemos a qué dirección tenemos que acceder para modificar los bytes necesarios? La respuesta se basa en tres pasos:

1º Volcar la memoria del proceso, identificado por su PID para la obtención de su Handle
2º Analizar el código tal cual lo hacemos como si hubiéramos leído el ejecutable del disco.
3º Programar el patch usando WriteProcessMemory hacia la dirección deseada, programándolo tal y como lo he explicado.

La cuestión y uso de ReadProcessMemory radica en el punto 1º. Si nos queremos ahorrar gran parte de trabajo, podemos usar la opción de los debuggers de "Adjuntar proceso". Ya que en definitiva internamente está haciendo las correspondientes llamadas a ReadProcessMemory, para volcar la memoria, iniciando dicha lectura en la correspondiente dirección (Entry Point) indicada en la cabecera PE del ejecutable.

No obstante y por completar este artículo, podríamos programarlo nosotros de forma muy trivial.

Así por ejemplo usando OllyDbg, volcamos la memoria del proceso "Prueba.exe", y observamos el código ensamblador expuesto al principio a partir de la dirección $401089:

```
00401089 ---- 8D95 FCFFFFFF
0040108F ---- B8 00000000
00401094 ---- 8902
```

Si programamos de forma similar a WriteProcessMemory, pero llamando a ReadProcessMemory, para leer a partir de la dirección $401089, tendríamos un código similar a éste:

```
var
  i: integer;
  handleVentana: integer;
  PID: integer;
  buffer: PChar;
```

```
        handleProceso: integer;
        leidos: Cardinal;
      begin
         handleVentana:=FindWindow(nil, 'C:\Documents and Settings\Samir\Escritorio\
Victima.exe');
        if WindowName = 0 then begin
              MessageDlg('Proceso no encontrado', mtWarning, [mbOK], 0);
        end else begin
              ThreadId := GetWindowThreadProcessId(handleVentana, @PID);
              handleProceso := OpenProcess(PROCESS_ALL_ACCESS, false, PID);
              GetMem(buffer, 13);
              ReadProcessMemory(handleProceso, ptr($401089), buffer, 13, leidos);
              CuadroTexto.Clear;
              for i:=0 to 12 do begin
                      CuadroTexto.Lines.Add(IntToHex(ord(buffer[i]), 2));
              end;
              FreeMem(buffer);
              CloseHandle(handleProceso);
        end;
      end;
```

Este código lee 13 bytes a partir de la dirección $401089, y los muestra en formato hexadecimal en un cuadro de texto, mostrando los bytes correspondientes a las instrucciones LEA EDX,DWORD PTR SS:[EBP-4]; MOV EAX,0; y MOV DWORD PTR DS:[EDX],EAX.
Si quisiéramos leer todo el ejecutable, tal y como lo haría un debugger, debemos leer a partir de la dirección de entrada indicada por la cabecera PE, esto es $401000, y leer tantos bytes como sea el tamaño del proceso.
Para completar este artículo sólo nos faltaría añadir lo comentado anteriormente sobre el trabajo con la memoria en modo Kernel, pero como ya hemos dicho, por ser un campo de contenido muy amplio lo dejaremos para otra ocasión.

Aplicaciones y reflexión final:

Las funciones que hemos visto de la API WIN32 (ReadProcessMemory y WriteProcessMemory), y su forma de uso, tienen múltiples aplicaciones, entre las que podemos citar:

- **La depuración de procesos**. Los debuggers que utilizamos normalmente (ollydbg, softice, windbg, etc...) utilizan estas funciones para poder leer el contenido de la memoria de un proceso, y según sea el caso, poder hacer las escrituras de los bytes necesarios.
- Siguiendo la línea de depuración de procesos, pensemos el caso de intentar desensamblar un programa compilado en código intermedio (bytecode, IL, etc...). El resultado será que no podemos desensamblarlo puesto que no se trata de un fichero binario (lenguaje máquina). Sin embargo si ejecutamos el programa y leemos el contenido de la memoria, tendremos el conjunto de instrucciones de dicho proceso. Por tanto una aplicación más es el **desensamblado de código intermedio, tipo Java o .NET**
- Realizar **volcados de memoria** (Memory dumps). O simplemente la elaboración de herramientas tipo **editores hexadecimal**, los cuales vuelcan el contenido de la memoria a un buffer interno, ya sea realizada la lectura/escritura a través de un fichero o de un proceso.
- Los conocidos **cheats engines** para algunos juegos se basan en este procedimiento. Sería algo similar al ejemplo que hemos visto

- La ejecución de un código propio en el contexto de un proceso con privilegios en el sistema. Este caso es bien conocido por los diseñadores de **virus y rootkits**. El procedimiento consiste en que el virus inyecta su código en el proceso de una aplicación que sabemos seguro existe en el sistema y además posee algún tipo de privilegio. Tal es el caso de Internet Explorer (sabemos que existe en todas las versiones de Windows, y suele poseer permisos de conexión en el firewall por defecto). Por tanto el código inyectado se ejecutará también con dichos permisos. Esta técnica básicamente se utiliza para saltar las protecciones impuestas por un firewall.

- Idem para el caso de un proceso con privilegios de administrador, o un servicio del sistema. Un servicio es un proceso en segundo plano, residente en memoria, y que suele ser iniciado por un proceso en modo kernel (por ejemplo un driver). Por tanto, una aplicación más sería **escalar privilegios.**
- Manipulación de cualquier aplicación para alterar su funcionamiento normal. Esto es otro caso más genérico de lo que hemos estado viendo... básicamente es una técnica más que se emplea en **ingeniería inversa y cracking**. Por ejemplo, el caso de que hagamos que una aplicación no pida que insertemos un CD.
- Elaboración de software tipo **sniffers**, para leer datos transferidos entre aplicaciones (mensajes, eventos, etc...), o entre drivers-aplicación.
- Diseño de **software de mantenimiento, diagnosis, benchmark, gestión de memoria, máquinas virtuales**, etc...
- Como técnica de **polimorfismo** en nuestra aplicación, así como **protección de software**. Simplemente escribiendo en la zona de memoria asignada a nuestro proceso, similar a lo visto en la ezine 3.
- Y un largo etc...

Tal vez pueda parecer que el uso de estas funciones para leer/escribir manualmente de la memoria, sea para conseguir fines nada buenos (o éticos?), y la cuestión es que realmente un sistema operativo debe proveer de toda la funcionalidad para que el usuario se despreocupe de tener que hacer nada manualmente con la memoria. Con lo cual, en la mayoría de los casos, un software con fines no éticos que llame a estas funciones, será para hacer algo "sospechoso".

Por tanto, la cuestión se basa en el dilema moral del uso de las cosas... "Un cuchillo es una herramienta que se utiliza para cortar alimentos, y no para matar a personas" (por gondar_f), del mismo modo que "un troyano es una aplicación para la administración remota de un equipo, y no para espiar a un amigo", o "el protocolo P2P se diseñó para el intercambio de ficheros, y no para descargar contenido pirata o con derechos de autor".

Hasta la próxima,

HySTD

MOVILES PERDIDOS

Primeramente debemos de ver una introducción a las diversas técnicas que vamos a encontrarnos. Lo cierto es que el teléfono móvil ha sido uno de los vértices de la revolución digital, de lo que algunos ya califican como la era Geek. Antaño, el tener un teléfono móvil era un lujo, ya no solo porque los precios fueran privativos, sino por la tecnología entonces de primera generación. Lo cierto es que ahora casi todo el mundo tiene al menos un teléfono móvil, incluso muchos se han convertido en móvil adictos.
De esta forma, si tu terminal no tiene Bluetooth o Cámara de 2 megapíxeles, es de otra época. Los móviles, son mas caros, mas completos y en definitiva, mas valiosos.

El problema es cuando te roban o lo extravías. La primera sensación es de mero pánico. No sabes si lo perdiste, si te lo sustrajeron, aunque realmente es lo mismo, ya que es complicado que te sea retornado en ambos casos.

Los números rápidos de las tres principales operadoras en España son:

- Vodafone: 123.
- Movistar: 609 (si se llama desde un teléfono Movistar) o 1485 (si se llamada desde un fijo o un móvil de otra compañía).
- Orange: 1414.

Hasta no hace demasiado tiempo, lo mas importante y eficiente era comunicar su falta a la compañía, que procedían a dar de baja esa SIM, para realizarte una clonación de la misma, para no perder tu número, a pesar del daño de la pérdida de la agenda o del contenido del terminal. Al menos, no te llegan facturas escandalosas de números que no conoces y que tú jamás has llamado. Además, tenemos la opción de bloquear el IMEI del móvil. Para saber nuestro numero de dispositivo, es tan simple como *#06# y automáticamente aparecerá el número. El IMEI viene inscrito en un microchip dentro del teléfono, consta de 15 dígitos y ese número es único, como la MAC de nuestro PC, y si te roban y al hacer la denuncia le das ese dato a la operadora, el teléfono pasa a estar en una "lista negra" denominada EIR (Equipment Identity Register) y no lo pueden volver a reactivar en ninguna otra compañía.

Según podemos ver en varios Wikis, el código de IMEI consta de cuatro partes y sigue el siguiente esquema: 358987010052195

- La primera parte (358987) se denomina Type Allocation Code (TAC), en donde los primeros dos dígitos indican el país.
- La segunda parte (01) es el Final Assembly Code (FAC) e indica el fabricante del equipo.
- La tercera parte (005219) es el número de serie del teléfono.
- El último dígito (5), es el dígito verificador, usado para verificar que el IMEI es correcto.

Una Web bastante interesante al respecto:

www.numberingplans.com

Es muy difícil en caso de robo la recuperación, pero puede ocurrir. Lo primero que te hacen cuando tu estas realizando la denuncia, es una descripción completa del terminal, para poder identificarlo: marca, modelo, operador telefónico y código IMEI. Con estos datos, las fuerzas del orden pueden rastrear el móvil. Pero si no se conoce el código IMEI, será imposible recuperarlo, ya que no se podrá asociar ningún teléfono recuperado al usuario.

Esto esta muy bien, pero los usuarios de móviles queremos (o al menos solemos querer) venganza, que le detengan y si puede ser, que le metan una ristra de ostias.
Para localización de un terminal, las formas más viables son la triangulación y la localización mediante GPS. Consultando otra vez, podemos ver que la localización GSM es un servicio que permite determinar, donde se encuentra físicamente un terminal móvil determinado. Este proceso se realiza mediante triangulación, pero no me inmiscuiré demasiado, ya que es un tema complejo, y farragoso.
El siguiente es más actual, moderno y preciso, pero requiere que tu móvil sea de última generación y que disponga de un GPS integrado.
Latitude, la aplicación desarrollada por el hoy en día omnipresente Google, la nueva funcionalidad que permite informar de nuestra posición actual o publicarla en Internet.
El servicio opcional Latitude utiliza datos de mástiles de telefonía celular, del Sistema de Posicionamiento Global (GPS, por sus siglas en inglés) o de equipos con sistemas de comunicación inalámbrica Wi-Fi para actualizar automáticamente la ubicación de los usuarios. Pero no es el único servicio de este tipo: también están SpotMe para iPhone, My Location para BlackBerry, miniGPS y Nokia FriendView para Symbian, e incluso hay localizadores GPS autónomos.

Ahora vienen los peros; la realidad, es que tienes que tener previsión en estos casos, ya que todas estas tecnologías solo permiten su uso a posteriori. Ente otras cosas, solo los dispositivos mas modernos soportan estas tecnologías, y la policía no se para demasiado tiempo en estas cosas, por desgracia. ¿Solo nos queda entonces la resignación y la primera parte de este articulo? No, no me consuela eso para nada. La prheak-manía ha sufrido de baches, pero es una tecnología resistente y muy adaptable, por lo que enseguida resurge con nuevas tendencias y características, tan solo es cuestión de leer y buscar, lo importante es perseverar.

A mi amiga Diana, por la idea

By Magic (ronaldo)

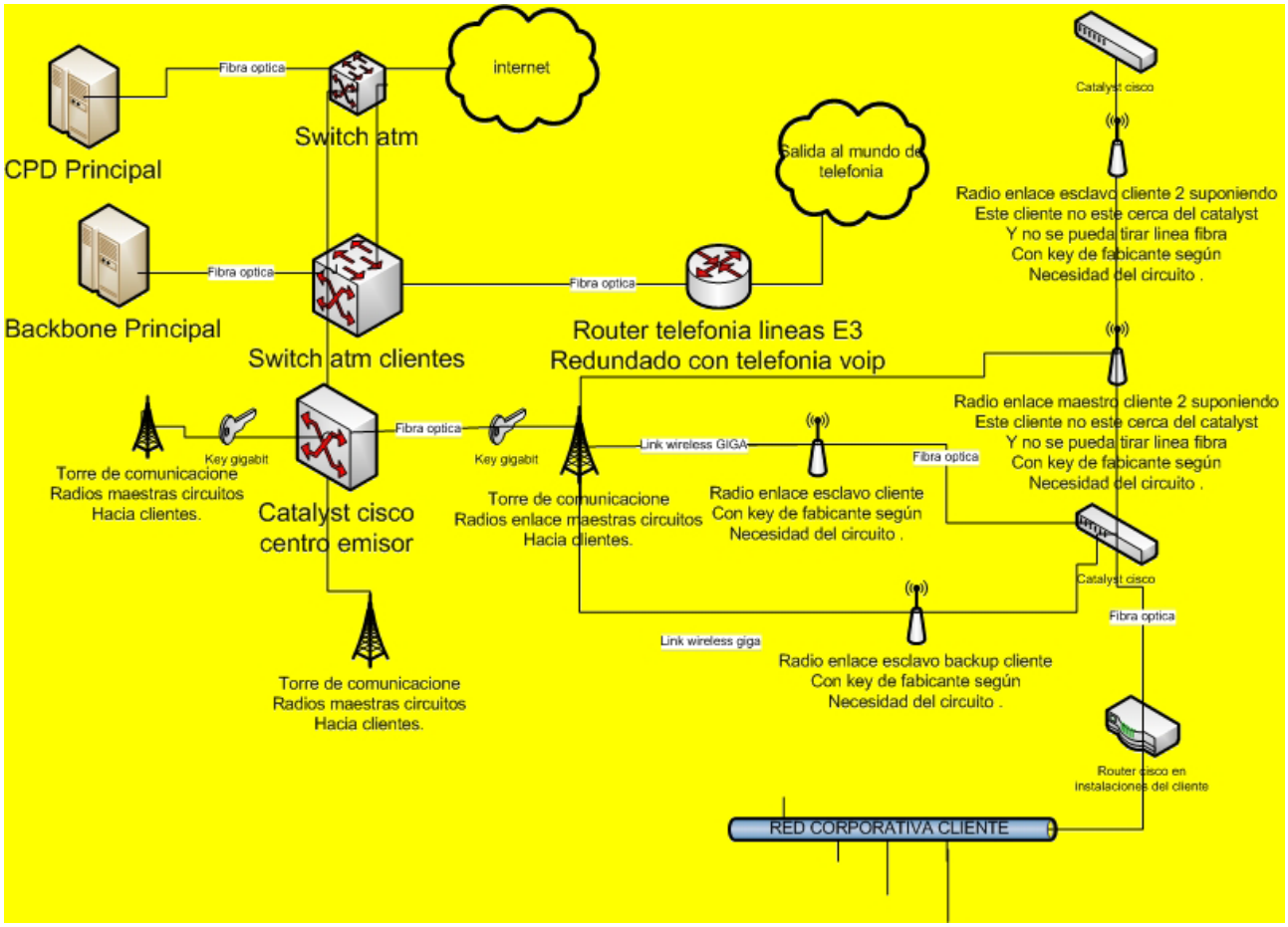
INICIACIÓN A RADIO ENLACES DE ALTA FRECUENCIA.

Bien en este primer manual vamos a explicar a grandes rasgos como estructurar y montar una red wireless con enlaces dragonwave ceragon radwin motorola todo ellos funcionando con dispositivos de enrutamiento tales como catalyst cisco router cisco y junnipers, cambiadores de medios, concentradores regeneradores de fibra etc, todo ellos con sistemas de alimentacion ininterrumpida también redundados.

Bueno en principio deberemos de estructurar la red creando un anillo extendido que a mi juicio es la mejor manera de estructurarla. Se creará a la vez otra red secundaria de respaldo o backup, como querais llamarla, para dar redundancia a fin de minimizar fallos y perdidas de conectividad, ya sea por apantallamiento, por orografía, por condiciones metereológicas adversas extremas o por pérdida de link por larga distancia, o por fallo en equipamiento debido a averías en el mismo.

Primeramente vamos a crear la estructura de red que necesitamos .

CONFIGURACIÓN PRIMARIA.
Vamos a estructurar creando en principio la infraestructura backbone con las troncales principales que a traves de ct (catalyst cisco) formarán un centro emisor redundado.



Vamos a explicar un poco esta estructura.

Tenemos en primer lugar el cpd y backbone conectados cada uno a un switch atm y a su vez interconectados entre sí, el backbone y cpd llevan integrados un junniper M7i, uno de los switch lleva la salida a internet total de la ciudad y el otro lleva la salida de telefonía de las lineas E3. De este modo redundamos la telefonía mediante voip a la misma vez que tenemos este switch dedicado a las líneas E3 para que no sature y no se oigan ecos ni se entrecorte la voz .Todo este cableado se realizará con fibra multimodo y si hay mucha distancia ente ellos se utilizará un regenerador de fibra de 24 km o mas según necesidad. Este switch atm que lleva la telefonía a su vez esta conectado a otro catalyst por fibra que será el encargado de dar las comunicaciones por radio enlace de larga distancia en un centro emisor en el que estará ubicado el mismo y las diferentes torres de comunicaciones que necesitemos. En dichas torres instalaremos equipamiento de radio enlace tipo dragonwave el cual tiene que tener licenciada en industria la frecuencia en la que emite y recibe señal, de la misma forma necesitará para cursar ese tráfico una key del fabricante que llevará un seguimiento de los errores, fallos y actualizaciones instaladas. Ese radio enlace puede cubrir distancias de 50 km sin problemas, siempre y cuando se oriente horizontal y verticalmente y siendo consecuentes con la configuración del dispositivo teniendo en cuenta condiciones de orografía, climatología y apantallamiento que puedan interferir, al igual que posibles interferencias de otros dispositivos que sin licencia emitan en la misma frecuencia y puedan interferirnos.

Dicho enlace linkará con un radio enlace situado lo mas cerca posible del cliente y se realizará con tantas troncales como necesitemos. Aquí hemos puesto tres, estos enlaces esclavos irán linkados por fibra a un catalist cisco de fibra y el mismo o directamente al router de un cliente que por ejemplo consuma 700mbps sincrónicas.
También, como puede ser apreciado, por otra boca del switch podemos extender la red hacia otro cliente mediante radio enlaces si la distancia es grande, o fibra si nos interesa. Si nos fijamos bien, la estructura principal de la red está toda redundada, pudiendo así optar por pasar el tráfico por uno u otro enlace según necesidad.

Escrito por: hail

Retos HackHispano Ezine Nº5.

Bienvenido una vez más a esta macabra sección. Hemos de decir que la participación de usuarios en el número anterior ha sido reducida, pero con un porcentaje alto de ganadores en relación al ńumero de participantes, y ya sabemos que "más vale calidad que cantidad"

La solución y explicación a los retos del número anterior podéis encontrarla en nuestro foro, en la siguiente dirección:

http://foro.hackhispano.com/showthread.php?t=33974

Los ganadores han sido:

Kirtash
Clarinetista

Esta vez os proponemos un reto relacionado con los conocimientos de formatos de imagen básicos (BMP, GIF, JPEG, etc...), en el que tendréis que apañárosla para encontrar lo que se busca... pero **¿qué es lo que se busca?** Responder a esta pregunta es clave para solucionar el reto. Podéis descargarlo de aquí:

http://www.hh-server.com/archivos/ezine/retos/logo.rar

No olvidéis enviar la solución de este reto a la dirección de la ezine: ezine@hackhispano.com, y para cualquier duda, sugerencia o comentario, podéis hacer uso del foro.

Buena suerte a todos y un saludo.

Staff:

Redactores:
Sn@ke
Iberhack / gondar_f
magic
HySTD
hail

Maquetación y Diseño:
mystery-man

Subdireccion del Proyecto:
HySTD

Dirección del Proyecto:
Clarinetista