Universidad Nacional de Córdoba

Cátedra de Sistemas Operativos II

FACULTAD DE CIENCIAS EXACTAS, FÍSICAS Y NATURALES

Trabajo Práctico N°3

Alumno: Oliva Arias, Carlos Agustín
Fecha: 5 de Junio de 2019

Índice

1. Introducción

1.1. Propósito

El objetivo del presente trabajo práctico es que el estudiante sea capaz de diseñar una aplicación para un sistema embebido.

1.2. Ámbito del sistema

Se pide que sobre un sistema embebido tipo Raspberry Pi o similar, que posea MMU, se desarrolle una aplicación consistente de un webserver, el cual provee datos sobre el hardware,permita diversas operaciones respecto a los módulos del kernel y también permita consultar datos sobre un producto del satélite GOES-16 mediante la consulta de su nube AWS pública.

1.3. Definiciones, Acrónimos y Abreviaturas

AWS: Amazon Web Services
CGI: Common Gateway Interface
CLI: Command-Line Interface
CSS: Cascading Style Sheets
GOES: Geostationary Operational Environmental Satellite
HTTP: Hypertext Transfer Protocol
HTML: HyperText Markup Language
S3: Simple Storage Service

1.4. Descripción general del documento

Las siguientes secciones describen el proceso de realización del trabajo, las herramientas y archivos auxiliares utilizados.

1.5. Referencias

[1] `http://brackets.io/`

[2] `https://www.w3schools.com/w3css/tryw3css_templates_dark_portfolio.htm`

[3] `https://tomeko.net/online_tools/cpp_text_escape.php`

[4] `https://www.sitepoint.com/uploading-files-cgi-perl/`

[5] https://www.digitalocean.com/community/tutorials/how-to-edit-the-sudoers-file-on-ubuntu-and-centos

[6] https//rclone.org

[7] https://github.com/notro/rpi-source/wiki

[8] https://www.lighttpd.net/

[9] https://geekflare.com/open-source-web-servers/

[10] https://en.wikipedia.org/wiki/Comparison_of_web_server_software

[11] https://metacpan.org/pod/CGI

2. Descripción general

Se requiere un webserver que pueda servir 3 tareas básicas, estas son:

1. Página que reporte información sobre recursos varios del sistema
2. Página de consulta de archivos disponibles en AWS de GOES-16 para producto ABI-L2-CMIPF, canal 13
3. Página que permita el listado, carga, instalación y remoción de módulos del kernel.

2.1. Restricciones

El desarrollo debe constar de HTML, CGI, y Perl o C como lenguajes a utilizar.

2.2. Suposiciones y dependencias

Dada la naturaleza del requisito respecto a los módulos del kernel, se asume que el servidor estará en una órbita privada, en donde la seguridad del sistema no es una preocupación, por lo que se han tomado libertades respecto a la forma de solucionar este requisito, una explicación más detallada se provee en la sección dedicada a este tema.

En cuanto a las dependencias, se encuentran dos para la página de GOES, la primera, bastante elemental, es la necesidad de estar conectado a Internet para la consulta de la nube. Además la consulta de la nube no podía hacerse

fácilmente usando un browser y HTTP (con un browser las consultas están limitadas a 1000 elementos). Entonces se investigaron alternativas, como la cli de aws (pero esta requería una cuenta, y ésta a su vez, requería información de facturación incluso para su Free Tier), y se terminó decidiendo sobre la herramienta rclone que permite consultas sobre S3 sin el uso de credenciales.

3. Diseño de solución

La solución consta de dos archivos fuente en c, tres en perl, un html, tres css, un makefile, y de archivos de configuración para el webserver, para rclone, y un archivo de configuración de privilegios para usuarios.

Sobre los archivos fuente C, uno corresponde a la página de recursos, en donde se leen algunos archivos del sistema de archivos virtual */proc* y se los parsea para visualizarlos en el html generado por cgi. El otro es el encargado de la página que lista los módulos y ofrece caminos a la instalación y remoción de los mismos.

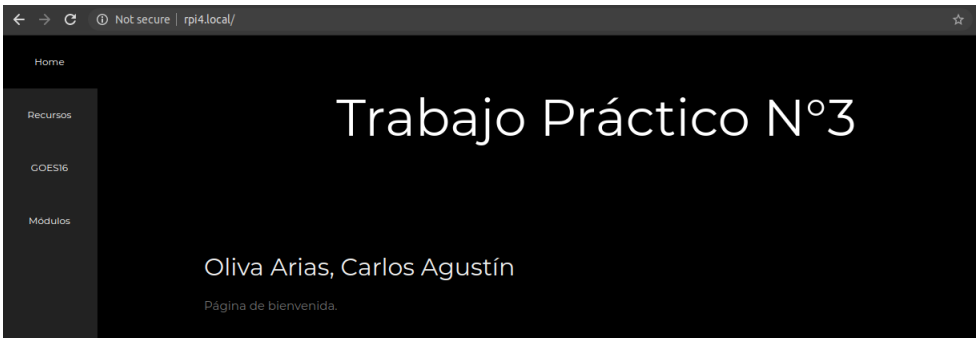
Para la carga del binario del módulo se siguió un tutorial de carga de archivos con perl, que solucionaba esa parte del problema, entonces se continuo ese código perl para la verificación del módulo, se intenta la instalación y de ser errónea se informa al usuario de esta situación proveyéndose información del módulo subido. La remoción de módulos también se resolvió en perl, donde simplemente se ejecuta el comando rmmod y se vuelve a la página anterior.

El archivo html es el índice, y los css son los provistos por el modelo utilizado. El makefile permite la compilación y en alguna medida el testing del diseño. Por último hablaremos de los archivos de configuración, al webserver debio agregarse la posibilidad de ejecutar cgi, y código .pl con perl. Rclone necesita un pequeño archivo para configurar la conexión a AWS-S3, estableciendo estos datos y la región de localización del bucket. El archivo de configuración de privilegios permite al usuario del webserver (www-data) ejecutar los comandos insmod y rmmod.

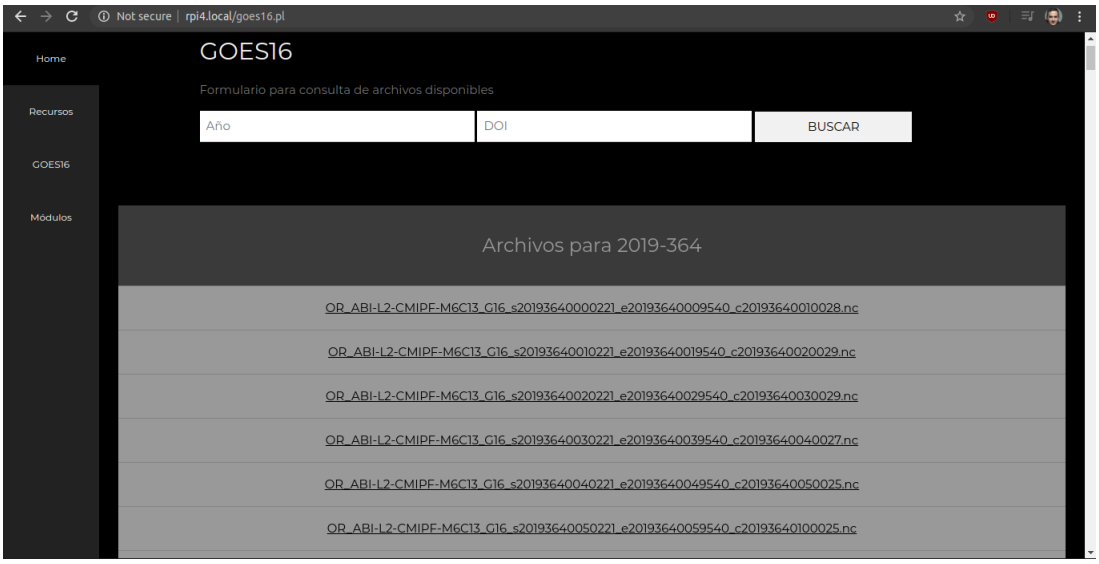
Algunos de los anteriores se pueden consultar en la sección del apéndice dedicada al código.

4. Implementación y resultados

Se adjuntan como resultados, ejemplos de ejecución en la notebook del estudiante. Se visualiza las páginas construidas mediante el browser Google Chrome 79.0.3945.117.

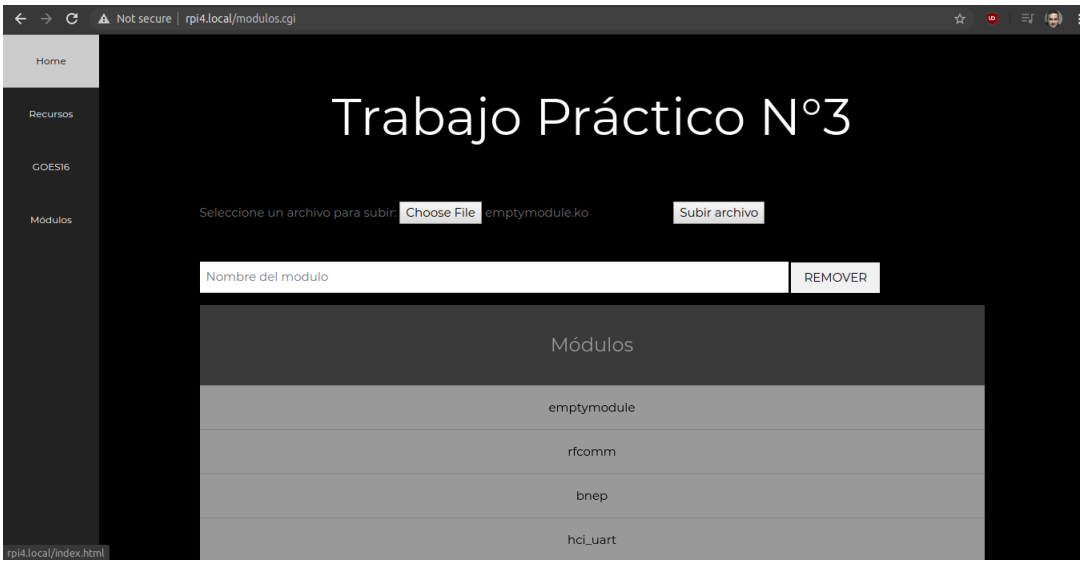


Página de inicio

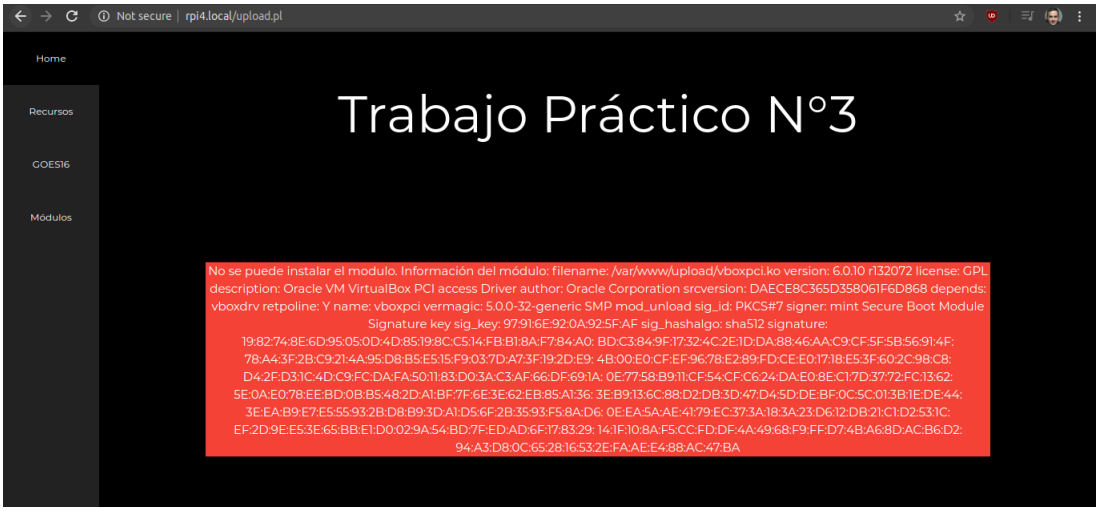


Consulta GOES

Recursos del sistema



Modulos cargados



Intento de carga de módulo incompatible

5. Conclusiones

Durante la realización del práctico se encontraron varios problemas, siendo la consulta de AWS la que más tiempo demando. El desarrollo tuvo sus altibajos en cuanto al avance, pero eventualmente se logro completar todo lo requerido. Se valora como primera experiencia de trabajo en estas tecnologías, principalmente Perl el cual era desconocido y resulto muy cómodo para creación de contenido html dinámico.

6. Apéndices

6.1. Código

Se adjunta el código utilizado, removiendo los strings largos que forman el html para facilitar la lectura:

```
1
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <string.h>
5  #include <time.h>
6  #include <unistd.h>
7  #include <getopt.h>
8
9  char s1[]="<!DOCTYPE html> <!--Termina s1 -->";
10 char res_start[]="<li  class=\"w3-padding-16\">";
```

```
11 char res_end[]="    </li>\n";
12 char s2[]="<!--Hasta el final s2 -->\n         </ul>\n      </
      div>    \n  </div> \n</div>\n<!-- END PAGE CONTENT -->\n</
      body>\n</html>";
13
14 void set_cpuInfo();
15 void set_infoMem();
16 void set_upTime();
17 char *rs_file(char path[],char info[]);
18
19 typedef struct cpuData{
20  int cantCPU;
21  char model[100];
22  char totalMem[100];
23  char freeMem[100];
24 } cpuData;
25
26 typedef struct tiempo{
27  long unsigned int valor;
28  int dias;
29  int horas;
30  int minutos;
31  int segundos;
32 } tiempo;
33
34 cpuData cpu;
35 tiempo uptime;
36
37 void set_cpuInfo(){
38
39  char temp[100];
40  FILE *fp;
41  int count=0;
42
43  fp = fopen("/proc/cpuinfo", "r");
44  if (fp==NULL) {
45          fputs ("File error",stderr); exit (1);
46     }
47
48     while(!feof(fp)){
49
50      fgets(temp,100,fp);
51
52      if(strstr(temp,"processor")){
53    count++;
54   }
55       else if(strstr(temp,"model name")){
56        char *pch;
57       pch = strchr(temp,':');
```

```
58        strcpy(cpu.model, pch);
59       }
60      }
61  fclose(fp);
62  cpu.cantCPU=count;
63   memmove(&cpu.model[0], &cpu.model[1], strlen(cpu.model));
64   int len = strlen(cpu.model);
65   if( cpu.model[len-1] == '\n' )
66     cpu.model[len-1] = 0;
67
68 }
69
70 void set_infoMem(){
71
72  char temp[100];
73  FILE *fp;
74  fp = fopen("/proc/meminfo", "r");
75  while(!feof(fp)){
76   fgets(temp,100,fp);
77   if(strstr(temp,"MemTotal")){
78    char *pch;
79      pch = strchr(temp,':');
80       strcpy(cpu.totalMem, pch);
81      }
82      else if (strstr(temp,"MemFree")){
83       char *pch;
84      pch = strchr(temp,':');
85       strcpy(cpu.freeMem, pch);
86      }
87  }
88  fclose(fp);
89 }
90
91 void set_upTime(){
92  char temp[100];
93  int count=0;
94  FILE *fp;
95  long int valor;
96  fp = fopen("/proc/uptime", "r");
97  while(!feof(fp)){
98   char *str =fgets(temp,100,fp);
99   char * pch;
100  char * aux[2];
101    pch = strtok (str," ");
102    while (pch != NULL)
103    {
104   aux[count]=pch;
105      pch = strtok (NULL, " ");
106      count++;
```

```
107     //aux[0] tiene el tiempo desde el inicio del sistema
108     //aux[1] tiene el idle time
109     valor= atol(aux[0]);
110      }
111       // printf("Val[0] es: %d\n",val[0]);
112 
113      int dias,horas,mins,segs;
114      dias = valor / (60*60*24);
115      horas = (valor-dias*(60*60*24)) / (60*60);
116      mins = (valor-dias*(60*60*24)-horas*(60*60)) / (60);
117      segs = (valor-dias*(60*60*24)-horas*(60*60)-mins*60 );
118      //printf("Valor: %ld  --> %d Dias,%d horas,
119      // %d minutos y %d segundos.\n",valor,dias,horas,mins,
         segs);
120      uptime.valor=valor;
121      uptime.dias=dias;
122      uptime.horas=horas;
123      uptime.minutos=mins;
124      uptime.segundos=segs;
125    }
126      fclose(fp);
127 
128 }
129 
130 void print_tiempo(struct tiempo timepo,char nombre[]){
131  printf("%s: %dD %02d:%02d:%02d\n",nombre,timepo.dias,timepo.
     horas,timepo.minutos,timepo.segundos);
132 }
133 
134 char *rs_file(char path[],char info[]){
135  FILE *fptr;
136  char *line = NULL;
137  size_t len = 0;
138  ssize_t read;
139  if ((fptr = fopen(path,"r")) == NULL){
140        perror("Error abriendo archivo.");
141        return line;
142  }
143  while ((read = getline(&line, &len, fptr)) != -1) {
144          if( strstr(line,info) != NULL) break;
145      }
146  fclose(fptr);
147  return line;
148 }
149 
150 int main(int argc,char *argv[]){
151 
152  set_cpuInfo();
153  FILE * file =popen("top -b -n1 | grep -i \"Cpu(s)\"" ,"r");
```

```
154  char consumo[100];
155  fgets(consumo,100,file);
156  pclose(file);
157  set_infoMem();
158  set_upTime();
159  char *btime=rs_file("/proc/stat","btime");
160  time_t bootTime=atol(btime+5);
161  printf("%s",s1);
162  printf("%s Procesador: %s x %d %s",res_start,cpu.model,cpu.
         cantCPU,res_end);
163  printf("%s %s %s",res_start,consumo,res_end);
164  printf("%s Total: %s Free: %s %s",res_start,cpu.totalMem,cpu
         .freeMem,res_end);
165  printf("%s Uptime: %dD %dh %dm %ds %s",res_start,uptime.dias
         ,uptime.horas,uptime.minutos,uptime.segundos,res_end);
166  printf("%s %s %s",res_start,ctime (&bootTime),res_end);
167  printf("%s",s2);
168  return 0;
169 }
```

recursos.c

```
1  #include <stdio.h>
2
3  char s1[]= "<!DOCTYPE html> <!-- Fin string1 -->";
4  char mod_start[]="<li class=\"w3-padding-16\">";
5  char mod_end[]=" </li>\n";
6  char s2[]="<!-- Inicio string2 -->\n        </ul>\n      </
       div>\n</div>  \n\n</body>\n</html>";
7
8  int main(int argc,char *argv[]){
9
10  FILE * file =popen("lsmod" ,"r");
11
12  char lista[100];
13  char str[30];
14  printf("%s",s1);
15  fgets(lista,100,file);
16  while( fgets(lista,100,file) != NULL){
17   sscanf(lista,"%s",str);
18   printf("%s %s %s",mod_start,str,mod_end);
19  }
20  printf("%s",s2);
21  pclose(file);
22  return 0;
23 }
```

modulos.c

```
#!/usr/bin/perl -w

use strict;
use CGI;
use CGI::Carp qw ( fatalsToBrowser );
use File::Basename;
use POSIX qw(strftime);

my $query = new CGI;

print <<'START_HTML';
START_HTML

my $year = $query->param("year");
my $doi = $query->param("doi");

if ($year eq ""){
 $year =  strftime "%Y",localtime;
}
if ($doi eq ""){
 $doi =  strftime "%j",localtime;
}else{
  $doi = sprintf("%03d",$doi);#si el usuario no lo hizo
     relleno con ceros
}

print ("<li class=\"w3-dark-grey w3-xlarge w3-padding-32 \">
    Archivos para $year-$doi </li>\n");

my $results = `sudo rclone --config=.rclone.config ls  goes:
    noaa-goes16/ABI-L2-CMIPF/$year/$doi`;
my @lines = split /\n/ ,$results;

#http://noaa-goes16.s3.amazonaws.com//ABI-L2-CMIPF/$year/$doi
    /<Hour>/$filename

my $start_line = "<li class=\"w3-padding-16\">";
my $end_line = "</li> \n";


foreach my $line (@lines){
 if( index($line, "C13") ne -1) # C13 me da los resultados
     para canal/banda 13
 {
  my $filename = (split '/',$line) [-1];
        #el filename me indica la hora, para poder acceder al
            link
        my $hour = substr($filename,34,2); #hay 35 letras
```

```
            hasta que empieza la hora del escaneo
43          my $link = "<a href=\"http://noaa-goes16.s3.amazonaws
                .com/ABI-L2-CMIPF/$year/$doi/$hour/$filename\"
                target=\"_blank\" >";
44   print( $start_line,$link,$filename,"</a>",$end_line);
45  }
46 }
47
48 print <<END_HTML;
49 END_HTML
```

goes16.pl

```
1  #!/usr/bin/perl -w
2
3  use strict;
4  use warnings;
5  use CGI;
6  use utf8;
7  use CGI::Carp qw ( fatalsToBrowser );
8
9  my $query = new CGI;
10 my $mod_name = $query->param("mod_name");
11
12 my $rmmod_output =  `sudo /sbin/rmmod $mod_name`;
13
14 print $query->redirect($ENV{'HTTP_REFERER'}) ;
```

rm_module.pl

```
1  #!/usr/bin/perl -w
2
3  use strict;
4  use warnings;
5  use CGI;
6  use utf8;
7  use CGI::Carp qw ( fatalsToBrowser );
8  use File::Basename;
9
10 $CGI::POST_MAX = 1024 * 10000;
11 my $safe_filename_characters = "a-zA-Z0-9_.-";
12 my $upload_dir = "/var/www/upload";
13
14 my $query = new CGI;
15 my $filename = $query->param("photo");
16
17 if ( !$filename )
18 {
19 print $query->header ( );
```

```
20 print "There was a problem uploading (try a smaller file).";
21 exit;
22 }
23
24 my ( $name, $path, $extension ) = fileparse ( $filename, '..*
       ' );
25 $filename = $name . $extension;
26 $filename =~ tr/ /_/;
27 $filename =~ s/[^$safe_filename_characters]//g;
28
29 if ( $filename =~ /^([$safe_filename_characters]+)$/ )
30 {
31 $filename = $1;
32 }
33 else
34 {
35 die "Filename contains invalid characters";
36 }
37
38 my $upload_filehandle = $query->upload("photo");
39
40 open ( UPLOADFILE, ">$upload_dir/$filename" ) or die "$!";
41 binmode UPLOADFILE;
42
43 while ( <$upload_filehandle> )
44 {
45 print UPLOADFILE;
46 }
47 close UPLOADFILE;
48
49 print $query->header ( );
50
51 system("sudo /sbin/insmod upload/$filename"); #intento
       instalar el modulo
52 my $module_name = substr $filename, 0, -3; #el archivo es .ko
53 my $is_present = `lsmod | grep $module_name`;
54 my ($hidden,$disabled,$disabled_string);
55
56 if ($is_present ne "") {
57 #Esta presente, se instalo correctamente.
58  $hidden = "hidden";
59  $disabled = " ";
60  $disabled_string =" Se instalo correctamente.";
61 }
62 else {
63 #No se instalo correctamente. Muestro info del modulo para
       ayudar.
64  $hidden = " ";
65  my $modinfo_str = `/sbin/modinfo /var/www/upload/$filename`;
```

```
66  $disabled_string = "No se puede instalar el modulo.
       óInformacin del ómdulo: $modinfo_str  \n ";
67  }
68
69  print <<END_HTML;
70  END_HTML
```

upload.pl

```
1   OBJECTS := module/emptymodule.o
2
3   obj-m := module/emptymodule.o
4   fifo-objs := $(OBJECTS)
5
6   compile:
7     sudo gcc src/recursos.c -o /var/www/recursos.cgi
8     sudo gcc src/modulos.c -o /var/www/modulos.cgi
9     sudo cp perl/upload.pl /var/www/upload.pl
10    sudo cp perl/rm_module.pl /var/www/rm_module.pl
11    sudo cp perl/goes16.pl /var/www/goes16.pl
12  debug:
13    sudo cat /var/log/lighttpd/breakeage.log
14  service:
15    systemctl status lighttpd.service
16  clean:
17    sudo rm -rf /var/www/upload/*
18  modulo:
19    sudo make -C /lib/modules/$(shell uname -r)/build M=$(PWD)
         modules
```

Makefile

6.2. Guía paso a paso

- Instalar el webserver
- Configurar servidor para cgi con programa escrito en C al estilo "Hello World"
- Aprender básicos de html y css
- Bajar brackets [1]
- Copiar un modelo sencillo para estilizar la salida [2]
- Terminar página de inicio
- Transformar los html en strings de C [3]
- Escribir código fuente en c para recursos
- Escribir código fuente en c para mostrar los módulos
- Adaptar código de perl para subir un archivo al servidor[4]
- Escribir con visudo el archivo de configuración de sudoer para usuario www-data[5]
- Escribir en perl la página de carga de módulos
- Instalar rclone [6]
- Escribir archivo de configuración de rclone
- Escribir código de perl para formulario de Goes y muestra de resultados
- Compilar un modulo "Hello World"para la rpi4 [7]

6.3. Elección de SO y Web Server - Tabla comparativa

La elección del sistema operativo y webserver a utilizar estuvo circunscripta por la siguiente situación. Al inicio del desarrollo solo se contaba con una Raspberry Pi 1 con una tarjeta Sd de 4GiB para memoria secundaria, en la cual solo podía instalarse Raspbian lite (sin escritorio) o menor, cuyo peso de 2.2GiB daba margen para la instalación de software. Al haber trabajado anteriormente con lighttpd [8] en una computadora con RAM de 512MB se considero que también era apto para las restricciones de hardware actuales. De este modo dado que ambos eran conocidos, situación que aceleraría el

desarrollo, se optó por ellos. Promediando el tiempo de desarrollo, se consigue una Raspberry Pi 4, y una tarjeta Sd de mayor tamaño, y aún así se mantuvo la elección del web server por las cuestiones explicadas anteriormente. De todas maneras, para cumplimentar lo requerido se realiza la tabla solicitada:

SO	Soporta rpi4	Descripción
Raspbian	si	soporte oficial
Noobs	si	para novatos
UbuntuMate	no	propósito general
Kali Linux	si	orientada a ciberseguridad
UbuntuCore	no	ligero
UbuntuServer	si	servidor
OSMC	no	orientada a mediacenter
LibreELEC	si	orientada a mediacenter
Mozilla WebThings	no	monitor de IoT
RISC OS	no	propósito general
Windows 10 Iot Core	no	propósito general
Lakka	si	orientada a videojuegos

Webservers	Soporta CGI	Descripción
Apache HTTP Server	si	escalabilidad
Nginx	no	escalabilidad
Apache Tomcat	si	Java servlets
lighttp	si	ligero
jetty	si	Java servlets
caddy	parcial	HTTPS Automático
Mongoose	si	programable
Naviserver	si	programable

6.4. Métodos GET y POST

El método GET es el más comúnmente usado de los métodos HTTP, se utiliza para requerir datos desde una fuente especificada. En el presente trabajo se utiliza GET para todas las páginas que muestran información directamente al usuario sin requerir de entrada alguna, estas incluyen el inicio, los recursos y la página de módulos. Si bien puede utilizarse GET con información en las url para el formulario de GOES se prefirió no hacerlo de este modo.

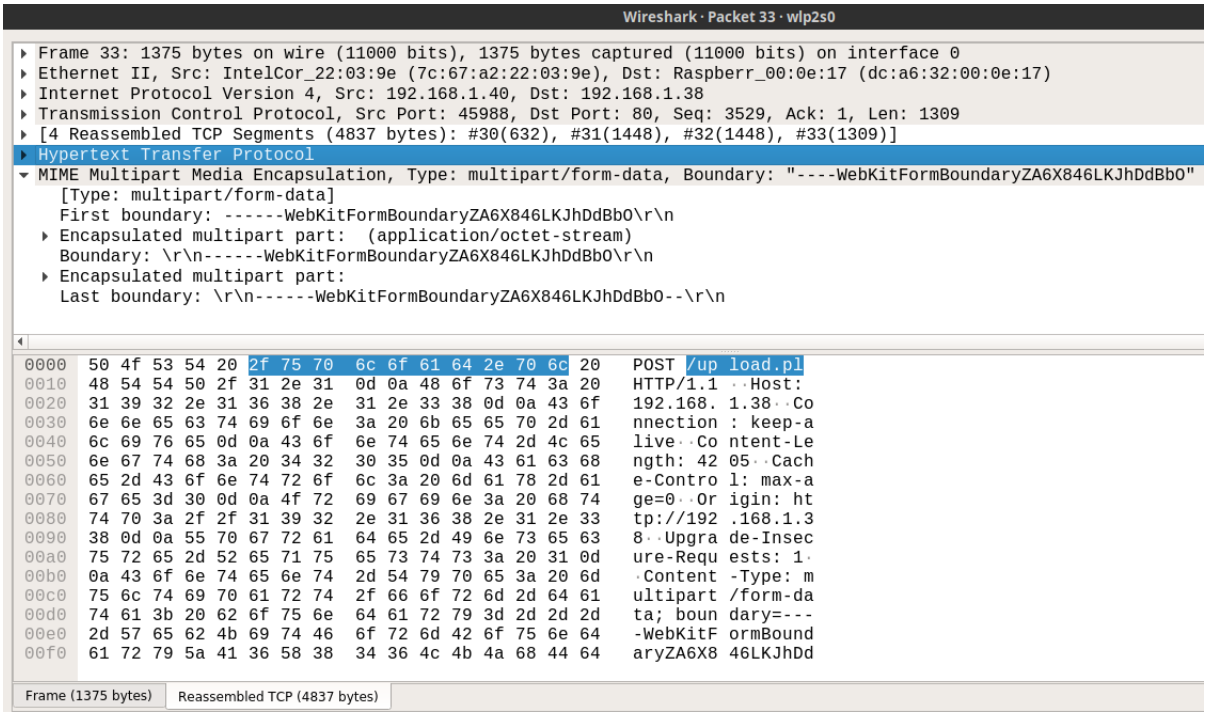
El método POST es utilizado para enviar datos al servidor creando o modificando recursos. Se utiliza para el formulario de GOES y en la subida del archivo como única opción puesto que debemos transferir un binario.

6.5. Paso de archivos

En el html que lista los módulos se encuentra lo siguiente:

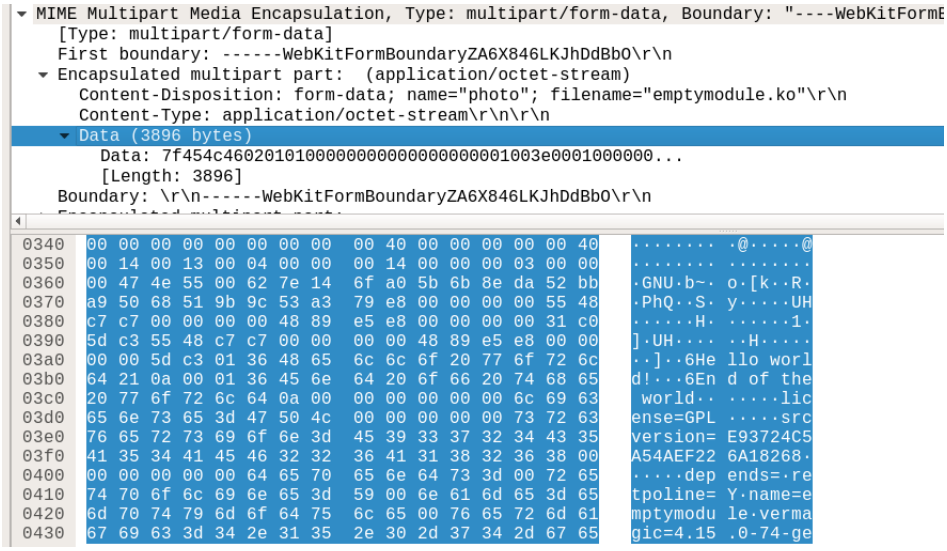
```
<form action="upload.pl" method="post" enctype="multipart/
    form-data"> Seleccione un archivo para subir:\n  <input
    type="file" value="Abrir..." name="photo" id="photo">   <
    input type="submit" value="Subir archivo" name="submit">
     </form>
```

Esto permite al cliente seleccionar el archivo desde su filesystem mediante la apertura de una ventana de diálogo del sistema operativo. El browser del mismo prepara una petición HTTP con el método POST, que incluye al archivo como puede verse en la siguiente captura de Wireshark:



Método Post

La petición es recibida por el script upload.pl, el cual opera con el módulo de perl cgi.pm [11], almacenando los datos del parámetro *photo* que fueron parte de la petición, como vemos en la siguiente figura:



Método Post

De la petición se obtiene además el nombre del archivo en el cliente, para poder crear un archivo del mismo nombre y contenido en el filesystem del servidor.

6.6. Inserción módulo por CGI

La inserción del módulo puede hacerse mediante el uso del comando ***insmod*** o ***modprobe***, siendo el último superior en cuanto manejo de dependencias y errores. Aun así insmod permite la inserción de un módulo desde cualquier directorio, como es nuestro caso. En el presente trabajo se intenta la instalación del módulo y luego se verifica si esta fue positiva mediante el comando **lsmod**. Si la instalación falla simplemente se imprime la información relativa al módulo a cargar.

Algunos de los comandos mencionados deben ejecutarse con privilegios, puesto que dan o remueven funcionalidad del kernel, pudiendo poner en riesgo al equipo. Como esto ya es de por sí arriesgado, se permite al usuario www-data (que ejecuta el servidor web) la ejecución de cualquier comando con privilegios sin requerirle la contraseña. Esta configuración no es óptima pero es más sencilla:

```
1 www-data ALL=(ALL) NOPASSWD: ALL
```

010_www-data-nopasswd