Apéndices

Apéndices

Apéndice A. Script para realizar el envío masivo.

```
#!/usr/bin/perl -w
use Mail::MboxParser;
use MIME::Decoder;
use MIME::Base64;
use MIME::Lite;

use Email::MIME;
use Net::Nslookup;
use Net::Whois::IANA;
use Pg;
use Email::MIME::Encodings;
use MIME::QuotedPrint;
use MIME::Base64;
use Unix::PID;
use MIME::Lite;
use MIME::Parser;
use MIME::Entity;


# Set $debug to 1 if you'd like to see server responses
my $debug = 1;

#Validacion de argumentos
if(scalar(@ARGV) < 2 ){
        print "Uso: $0 <dirección de correo> archivo_Mbox\n";
        exit;
}

#Validacion de la existencia del directorio tmp, en donde se guardaran los attachments
if ( -e "tmp" && -d "tmp" ){
        print "Utilizando tmp/ como deposito de attachments\n" if $debug;
}
else{
        `mkdir tmp`;
}
#Opciones del Mail::MboxParser
my $parseropts = {
   enable_cache    => 1,
   enable_grep     => 1,
   cache_file_name => 'mail/cache-file',
};
#Creamos un objeto Mail::MboxPAr
my $mb = Mail::MboxParser->new("$ARGV[1]",
                        decode     => 'BODY|ALL',
                        parseropts => $parseropts);
my $attach;
my $num_mail=0;
my $num_attach=0;
my $decoded=" ";
my $ents;
my $ent;
my $i=0;
my $type = " ";
my $body_str=" ";
```

```
my $encod = " ";
my $body=" ";
my $dispot=" ";
my $from= " ";
my $subject = " ";
my %mails=();
my $nmails=0;
#Para registrar el tiempo de inicio
@months = qw(Enero Febrero Marzo Abril Mayo Junio Julio Agosto Septiembre Octubre Noviembre
Deciembre);
@weekDays = qw(Domingo Lunes Martes Miércoles Jueves Viernes Sábado Domingo);
($second, $minute, $hour, $dayOfMonth, $month, $yearOffset, $dayOfWeek, $dayOfYear,
$daylightSavings) = localtime();
$year = 1900 + $yearOffset;
my $startTime = "$hour:$minute:$second, $weekDays[$dayOfWeek] $months[$month]
$dayOfMonth, $year";

while (my $msg = $mb->next_message) {
        $from = $msg->header->{from};
        $subject= $msg->header->{subject} || '<No Subject>';
        $subject =~ s/\*\*\*SPAM\*\*\*//g; #Sustituye el mensaje ***SPAM***
        $subject =~ s/\*\*\*INFECTED\*\*\*//g; #Sustituye el mensaje ***INFECTED***
        $subject =~ s/\*\*\*Posible SPAM\*\*\*//g; #Sustituye el mensaje ***POSIBLE SPAM***
        print "$num_mail FROM: $from \n" if $debug;
        print "Subject: $subject \n" if $debug;
    #Inicia el correo para ser enviado
        $sendMsg = MIME::Lite->new(
                To      => $ARGV[0] ,
                Subject  => $subject,
                Type     => $msg->effective_type,
                Data => ""
        );
        my %seen_headers = ();
        my %header_values = ();
        my $headersRef = $msg->header;
        my @headerKeys = keys %{$headersRef};
        #Para codificarlo correctamente con el charset
        foreach my $key (@headerKeys) {
                my $val = ${$headersRef}{$key};
                #print "key{$key} val{$val}\n" if $debug;
                if ( $key =~ "charset"){
                        #valido si existe value
                        if ( $value =~ "iso" ){
                                print "Charset $value\n";
                                $sendMsg->attr("content-type.charset" => $value);
                        }elsif( $key =~ "charset="){
                                $charset = substr $key, index($key, "charset=");
                                $charset = substr $charset, 8;
                                $charset =~ s/\"//g;
                                print "Charset $charset\n";
                                $sendMsg->attr("content-type.charset" => $charset);
                        }
                }
        }
        #print "Mail type $msg->effective_type\n" if $debug;
```

```
        #Guardo todos los archivos adjuntos en ./tmp
        $msg->store_all_attachments(path => './tmp');
        my $mapping = $msg->get_attachments;

        while(defined($ents=$msg->get_entities($i)) ){
                $type = " ";
                $body_str=" ";
                $decoded=" ";
                $encod = " ";
                $body=" ";
                $dispost=" ";
                $type = $ents->mime_type;
                $body=$ents->body;
                $encod = $ents->head->mime_encoding;
                $dispost = $ents->head->mime_attr("content-disposition");
                #print "$i typeMail: $type\n" if $debug;
                if(  !defined($dispost) || $dispost =~ m/inline/i ){
                        #print "1.if Encod: $encod\n" if $debug;
                        if( $encod =~ m/base64/i ){
                                #print "2.if Type Encode base 64: $type\n";
                                foreach $row(@$body){
                                        $body_str.=$row;
                                }
                                $decoded = decode_base64($body_str);
                                #print "$decoded\n" if $debug;
                                $sendMsg->attach(Type     =>$type,
                                        Data     =>$decoded);
                        }elsif( $type !~ m/multipart/i ){
                                #print "2.elsif Type Multipart: $type\n";
                                foreach $row(@$body){
                                        $body_str.=$row;
                        }
                                #print "$body_str\n";
                                $sendMsg->attach(Type     =>$type,
                                        Data     =>$body_str);
                        }

                }elsif ( $dispost =~ m/attachment/i) {
                        #print "$num_attach Attachment\n";
                        $fileName = $ents->head->mime_attr("content-disposition.filename");
                        if ( -e 'tmp/'.$fileName ){
                                print "fileName: $fileName\n";
                                $sendMsg->attach(Type     => $type,
                                        Path     => 'tmp/'.$fileName,
                                        Filename => $fileName,
                                );
                                $num_attach++;
                        }
                }
                $i++;
        }
        $sendMsg->send;
        $num_mail++;
        $i=0;
}
```

```
($second, $minute, $hour, $dayOfMonth, $month, $yearOffset, $dayOfWeek, $dayOfYear,
$daylightSavings) = localtime();
$year = 1900 + $yearOffset;
my $finalTime = "$hour:$minute:$second, $weekDays[$dayOfWeek] $months[$month]
$dayOfMonth, $year";

print "\n\nHora de inicio: $startTime\n";
print "Hora de finalización: $finalTime\n";
print "Numero de mensajes enviados: $num_mail\n";
print "Numero de mensajes con attachments: $num_attach\n";
```

Apéndice B. Script para realizar el conteo de correos

```
#!/usr/bin/perl -w
use Mail::MboxParser;
use MIME::Decoder;
use MIME::Base64;
use MIME::Lite;


#Opciones del Mail::MboxParser
my $parseropts = {
   enable_cache    => 1,
   enable_grep     => 1,
   cache_file_name => 'mail/cache-file',
};
#Creamos un objeto Mail::MboxPAr
my $mb = Mail::MboxParser->new("$ARGV[0]",
                           decode     => 'BODY|ALL',
                           parseropts => $parseropts);
my $attach;
my $num_mail=0;
my $num_attach=0;
my $decoded=" ";
my $ents;
my $ent;
my $i=0;
my $type = " ";
my $body_str=" ";
my $encod = " ";
my $body=" ";
my $dispot=" ";
my $from= " ";
my $subject = " ";
my %mails=();
my $nmails=0;


while (my $msg = $mb->next_message) {
   $from = $msg->header->{from};
   $subject= $msg->header->{subject} || '<No Subject>';
   $subject =~ s/\*\*\*SPAM\*\*\*//g; #Sustituye el mensaje ***SPAM***
   $subject =~ s/\*\*\*INFECTED\*\*\*//g; #Sustituye el mensaje ***INFECTED***

         print "$num_mail FROM: $from \n" if $debug;
         print "SUBJECT: $subject \n" if $debug;

   $num_mail++;
}
print     "Número de mensajes: $num_mail\n";
```

Apéndice C. Script para Revisar la Carga y el uso de Memoria.

```
#!/bin/bash
#programa para medir la carga del cpu y memoria utilizada
echo
"************************************************************************************************************"
echo "este script muestra la carga del cpu y el uso de memoria de manera iterativa cada 5
segundos"
echo
"************************************************************************************************************"
i=0
while [ $i -lt 5 ]
do
 echo "cpu utilizado el ultimo minuto, los últimos 5 min y últimos 15 min respectivamente"

 date
 cat /proc/loadavg | cut -d" " -f1-3
 echo " "
 echo "cantidad de memoria utilizada y memoria disponible"
 cat /proc/meminfo | head -2
 echo "_____________________________________________________________________________-"
 sleep 5
done
```