Sistema de Correo Electrónico desarrollado en lenguaje de programación Perl

Resúmen

Este artículo, eminentemente técnico pero con conclusiones orientadas a cualquier tipo de usuario, comparte la experiencia del desarrollo de un programa de computador orientado a la web para el manejo del correo electrónico. Este programa es parte del sistema "e-Génesis – El generador de sistemas" creado por el autor de este artículo y que fue galardonado con la Mención Especial en el Premio Colombiano de Informática 2006 de la Asociación Colombiana de Ingenieros de Sistemas, ACIS. En la primera parte se muestra el proceso que vive un programador para llevar a cabo este proceso. En la segunda, se muestran los resultados del control de *spam* basado en el análisis de diez y ocho mil setecientos veinticinco (18.725) datos registrados como términos usados en los correos indeseados que han servido para construir un patrón, que en este caso protege en un alto nivel de este problema.

Abstract

This article, which is mainly technical but with conclusions orientated for any type of user, shares the experience that emerged from the development of a computer program focused on the web, with the purpose of managing electronic mail. This program is part of the system "e-Genesis - the systems generator", created by the author of this article, who obtained the award "Premio Colombiano de Informática 2006 de la Asociación Colombiana de Ingenieros, ACIS". The first part of the article refers to the procedure that a programmer faces in order to carry out such process. The second part shows the results of the spam control, which are based on the analysis of 18.725 data; such data are registered as key words to be kept in the junk folder in order to create a pattern that prevents this problem from happening frequently.

Manuel Dávila Sguerra
mdavila@uniminuto.edu

1. Introducción

El correo electrónico es la aplicación más usada por los usuarios de Internet y fue la que intervino para que la web se convirtiera en la red universal, que es hoy en día.

En este artículo se habla de algunos aspectos técnicos sobre la forma cómo se puede programar un agente de correo electrónico propio y sobre los resultados del análisis del *spam* que el sistema arrojó, después de mucho tiempo de uso.

Los fundamentos de este trabajo se remontan a la experiencia del desarrollo del sistema e-Génesis – El Generador de Sistemas, desarrollado por el autor, que ganó la mención especial del Premio Colombiano de Informática 2006 de la Asociación Colombiana de Ingenieros de sistemas, ACIS.

e-Génesis ofrece una forma de crear aplicaciones, orientadas a Internet, de manera automática para un usuario sin conocimientos de programación y tiene variadas funcionalidades; pero, en este caso nos centraremos en una de ellas: el correo electrónico.

Constituye, como ya se dijo, la aplicación más importante para un usuario y la que mayor productividad le ha dado a las personas; también, está siendo la que más dolores de cabeza trae al administrarla puesto que, si antes nos quejábamos de no tener información al día, hoy nos quejamos de tener demasiada. Hay que pensar que una hora diaria de trabajo permanente en un año es equivalente a un mes laboral y surge entonces la pregunta: ¿cuántos meses laborales al año gastamos para sólo administrar el correo electrónico?

Y si a esto le sumamos el *spam*, que nos ataca por todos los lados, ¿cuántos de esos meses son tiempo perdido administrando información inútil?
Esos son dos temas que se tratarán en este artículo: el primero, orientado a los programadores y relacionado con: ¿cómo se programa

un sistema de correo electrónico? y con los usuarios finales: ¿qué resultados sobre el spam se han obtenido?.

El estilo de redacción que se usará es el de narrar el proceso de desarrollo del sistema intentando mostrar todo aquello que el programador tuvo que haber pensado, estudiado, investigado y llevado a la práctica para lograr el producto final.

2. El sistema de correos. ¿Cómo se programa?

Para comenzar debemos decir que el correo electrónico es uno de los servicio del protocolo tcp/ip llamado SMTP o Simple Mail Transfer Protocol, es decir el protocolo de transferencia de correo.

2.1 Algunas definiciones

En el diagrama de la figura 1 se puede observar que en el servidor, cuyo dominio es e-logicasoftware.com, aparece un programa llamado MTA es decir Mail Transfer Agent o Agente de Transporte de Correo que según Wikipedia (http://es.wikipedia.org/wiki/MTA) es el programa que maneja toda la comunicación en el servidor para transmitirla al usuario. Se dice que el usuario se comunica desde un MUA o Mail User Agent (http://es.wikipedia.org/wiki/MUA) o agente de correo de usuario que es el programa que los usuarios usan para administrar sus correos.

Para Linux existen varios MTA's siendo algunos de ellos Sendmail, Qmail, Postfix y es Sendmail uno de los más populares. Y los MUA's más conocidos, localizados en los pc's de los usuarios, son Eudora, Outlook, Netscape, Mozilla, Evolution, Pine, Thunderbird.

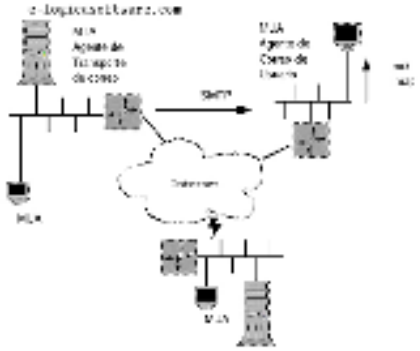

Figura N° 1. El sistema de correo electrónico.

Existen otros protocolos que intervienen como el **pop3** (http://es.wikipedia.org/wiki/POP3) y el **imap** (http://es.wikipedia.org/wiki/IMAP) que están diseñados para la gestión de los correos y manejos de las bandejas en los agentes de los clientes.

2.2 El diseño

El diseño del sistema de correos de e-Génesis considera que éste debe comportarse como un sistema de información que, como tal, usa una serie de repositorios de datos que van a contener la información del usuario. Valga la aclaración que el sistema está totalmente orientado a la web para que el usuario pueda manejar su correo desde cualquier parte y a cualquier hora.

Desde ese punto de vista se definen tres repositorios llamados Correos_enviados,
Correos_Recibidos y Spam cuyos nombres son auto explicativos con respecto de la información que van a almacenar.

El mismo diagrama anterior, pero aumentado en la figura 2, muestra la presencia de estos repositorios de datos con sus títulos en color azul para hacer notar su aparición:

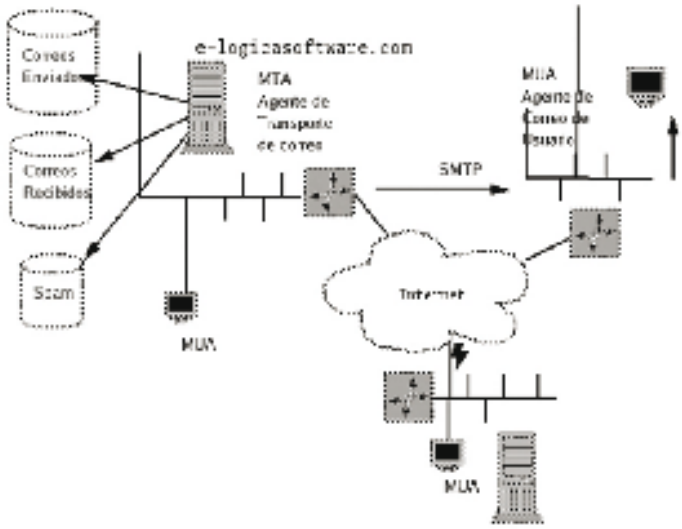

Figura N° 2. Los respositorios del correo.

Para enviar un correo, el usuario, debe usar un interfaz con los campos de información necesarios para lograr ingresar los datos pertinentes que en este caso se han diseñado de la siguiente manera, como se ve en la figura 3

Figura N° 3. El interfaz para el envío del correo en el MTA de e-Génesis.

De este interfaz rescatemos los campos de información de su diseño:

Campos estándar del correo enviado:

Fecha: fecha de envío puesta por el sistema.**De:** usuario origen del correo.
Para: usuario destino del correo.
Copia: con copia a...Tema: es el asunto o subject.
Contenido: cuerpo del correo.

Campos especiales del correo enviado por e-Génesis orientado a seguimiento.

Código_Empresa: código especial de la empresa o contacto para efectos de seguimiento.
Seguimiento: descripción de la nota de seguimiento.
Fecha_próximo_seguimiento: fecha en que se debe contactar de nuevo a la empresa.
Estado: del negocio o acuerdo que se está tratando con esa empresa.

Los usuarios están acostumbrados a ver este tipo de despliegue de información en su agente de correos especialmente los campos como "De, Para, Copia, Tema, Contenido".

Sin embargo, aparecen otros campos como "Código_Empresa. Seguimiento, Fecha_próximo_seguimiento, Estado", que no son usuales dentro de los agentes tradicionales pues han sido incluidos pensando en procesos posteriores que permitan usar los datos de los correos enviados como entrada para procesos transaccionales que pueden actualizar otras bases de datos, para seguimiento de eventos.

De esta forma el sistema de información de seguimientos se puede representar como lo muestra la figura 4:

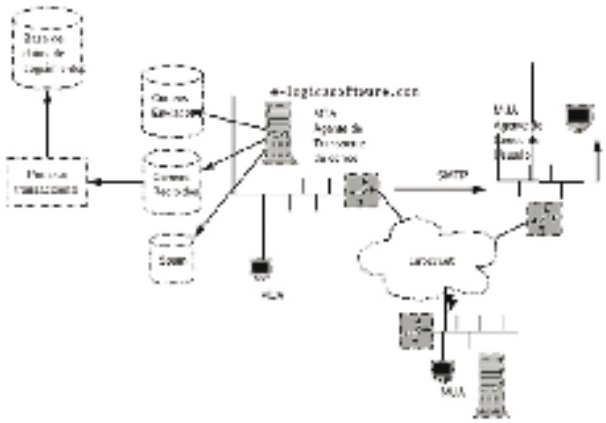

Figura N° 4. El correo para seguimiento de eventos.

2.3 El gusto por leer código fuente

Para aquellos seguidores de la programación de computadores leer código fuente es un "manjar", por lo cual se mostrarán a continuación unos detalles sobre la manera de programar este proceso de enviar un correo lo que conforma el corazón de un sistema de web mail. El lenguaje usado es Perl.

2.4 Programar el envío de correos

En primer lugar se puede usar un módulo que ayuda en el proceso llamado "MIME::Entity" el cual se debe incrustar dentro del programa: Los módulos de Perl son piezas de software predefinidas que se incrustan en los programas a través de la instrucción "use". En este caso se usan varios módulos cuyas definiciones se incluyen a medida en que aparecen. Estos residen en http://www.cpan.org

```
.....
use MIME::Entity; # Para analizar sintácticamente y decodificar mensajes tipo MIME
.....
```

El término "MIME" o "Multipurpose Internet Mail Extensions" que traduce "Extensiones de Correo Internet Multipropósito" son, según Wikipedia, "una serie de convenciones o especificaciones dirigidas a que se puedan intercambiar a través de Internet todo tipo de archivos (texto, audio, vídeo, etc.) de forma transparente para el usuario. Una parte importante del MIME está dedicada a mejorar las posibilidades de transferencia de texto en distintos idiomas y alfabetos".

Es la manera de definir tipos de documentos atados al correo como .doc, .xls, .mp3 etc.

Existe dentro del módulo "MIME::Entity" un método llamado *"build"* que sirve para armar el envío del correo tomando la información que el usuarios haya llenado en los campos "De, Para, Copia, Tema" cuyos contenidos se almacenan en las variables de perl: $de, $para, $copia, $tema.

Observemos la manera de utilizar este modo dentro del programa:

```
.....
$top = MIME::Entity->build(Type    =>"multipart/mixed",
            From    =>          $de,
            To      =>          $para,
            Cc      =>          $copia,
            Subject =>          $tema);
.....
```

El programa debe vigilar si el usuario que envió el correo incluyó un archivo atado en cuyo caso se debe almacenar previamente el nombre en la variable $archivoatado lo cual indica que hay un archivo atado en el correo. Para determinar el tipo MIME se usa el método *attach*. Lo que se pretende es definir el tipo de archivo que puede ser un PDF, DOC, XLS y demás existentes en los sistemas de hoy en día, tipo que se ha almacenado previamente en la variable $cualif para determinar el tipo desde un arreglo llamado $tmime que maneja el programa.

Es decir que si el documento atado es un .doc el arreglo $tmime{`doc'} debe haberse inicializado con el tipo MIME de los documentos escritos en *Word.*

En este caso se usa un método llamado attach para atar el documento que el usuario haya atado en su correo, como lo indica esta porción del programa en *perl:*

```
.....
        if ($archivoatado ne "")
                {
                        $top->attach(Path=>$archivoatado, Type => $tmime{$cualif}, Encoding =>
"base64");
                }
.....
```

Y se envía el correo:

```
.....
         $top->attach(Data=>$mensaje);
        open MAIL, "|  $mail_program -t -oi -oem" or die "open: $!";
        $top->print(\*MAIL);
        close MAIL;
.....
```

2-5 Programar los correos recibidos.

Esta funcionalidad lo que hace es consultar con el agente que maneja los buzones del sistema para tomar los datos en un formato propio del SMTP que deben ser leídos por un programa con capacidad de decodificarlo y poner los resultados en un repositorio de datos llamado Correos_Recibidos el cual podría ser de tipo relacional o de tipo textual.

Los correos llegan en formatos codificados con normas definidas los estándares de SMTP que muestran más cosas adicionales al mensaje para lo cual doy este ejemplo basado en un mensaje que dice "¿Hola cómo estás?". El SMTP entrega al programador un texto como el que se muestra a continuación en dónde marcamos en negrilla el mensaje mismo para resaltar todo lo adicional que debe ser decodificado por el programa:

Ver imagen en la siguiente página

El programa de correo debe ser capaz de leer este texto es decir, pedírselo al agente de correo, y decodificarlo para mostrarle al usuario final solo el mensaje "¿Hola cómo estás? " y usar los datos de control para armar la presentación adecuada en el navegador del usuario.

Algunos de esos datos, que aquí he denominado como de control, y como se ve en texto anterior son:

```
X-POP3-Rcpt: madavila@e-logicasoftware.com
Received: (from madavila@localhost)
        by host23.webserver1010.com (8.12.11.20060614/8.12.10) id kBB1tH2a002827;
        Sun, 10 Dec 2006 20:55:17 -0500
Date: Sun, 10 Dec 2006 20:55:17 -0500
Message-Id: <200612110155.kBB1tH2a002827@host23.webserver1010.com>
Content-Type: multipart/mixed; boundary="----------=_1165802117-2734-0"
Content-Transfer-Encoding: binary
MIME-Version: 1.0
X-Mailer: MIME-tools 5.411 (Entity 5.404)
From: madavila@e-logicasoftware.com
To: madavila@e-logicasoftware.com
Subject: Hola

This is a multi-part message in MIME format...
----------=_1165802117-2734-0
Content-Type: text/plain
Content-Disposition: inline
Content-Transfer-Encoding: binary

¿Hola cómo estás?

----------=_1165802117-2734-0--
```

```
Date:
Message-Id:
Content-Type:
Content-Transfer-Encoding:
MIME-Version:
X-Mailer:
From:To:
Subject:
Content-Type:
Content-Disposition:
Content-Transfer-Encoding:
```

Lo que sigue puede ser "griego" para los usuarios pero puede ser la salvación para un programador que está trabajando en este tipo de aplicaciones aclarando que sólo es una parte de todo el programa.

Se trata del programa que decodifica el texto entregado por el SMTP para convertirlo en un registro de datos que será almacenado en el repositorio llamado Correos_Recibidos

Para esto se usan varios módulos de Perl que se muestran a continuación:

```
use Mail::POP3Client;       # Para hablarle a un servidor pop3
use MIME::Entity;           # Para analizar sintácticamente y decodificar mensajes tipo MIME
use MIME::Parser;           # Para analizar sintácticamente y decodificar mensajes tipo MIME
use MIME::Words qw(:all);  # Decodificar iso 8859
```

El agente servidor, en este caso el programa "sendmail", necesita saber cuál es el nombre del servidor y un "login" y una contraseña para autenticarse, datos que deben venir ya en las variables $servidor, $logincorreo, $passwordcorreo para que el método "Mail::POP3Client" haga la autenticación:

```
$pop = Mail::POP3Client->new("$logincorreo", "$passwordcorreo", $servidor);
```

El sistema devuelve un valor con el número de mensajes que en ese momento residen allí sin haber sido leídos, dato que se atrapa en la variable $num_mesg:

```
$num_mesg = $pop->Count; # Cuántos mensaje hay?
```

Con ese valor se deben recorrer todos los mensajes para darle el tratamiento adecuado a cada uno.

En esta parte incluyo los comentarios explicativos dentro del cuerpo del programa como lo estilaría un buen programador para mayor claridad en la lectura del programa fuente:

```
for ($i = 1; $i <= $num_mesg; $i++) {          # Recorre todos los mensajes desde 1 hasta el ultimo
#---------------------------------------------------------- ----------------------------------
#          Lee el encabezado del cada correo en $temp1 usando el método Head
#-------------------------------------------------------------------------------------------
          $temp1 = $pop->Head($i);
#-------------------------------------------------------------------------------------------
#    Limpia los datos recibidos, lo cual es una acción subjetiva para cada caso
#    usando expresiones regulares . Por ejemplo la expresión s/,//g Busca (s de search)
#    el caracter coma (/,/) y lo reemplaza por nada (//) para todas las ocurrencias
#    de cada linea (g). Lea esto mas abajo observando la instrucción que dice "Quita comas"
#    para una mayor claridad.
#    Observe las otras instrucciones de limpieza de caracteres como \n (fin de línea) y
#    otros caracteres especiales que queremos limpiar como !, <, >,  etc
#-------------------------------------------------------------------------------------------
        $temp1 =~ s/\n/!/g;  # Quita control de fin de registro
        $temp1 =~ s/,//g;       # Quita comas
        $temp1 =~ s/^M/" "/g;  # Quita ^M
        $temp1 =~ s/!//g;        # Quita !
        $temp1 =~ s/,//g;        # Quita ,
        $temp1 =~ s/\</g;      # Quita <
        $temp1 =~ s/\>//g;     # Quita >
```

Ahora, viene algo interesante y un poco complejo y es que las líneas que en nuestro corto mensaje de "¿Hola cómo estás?" que como se ve en el listado mostrado son:

```
Date: Sun, 10 Dec 2006 20:55:17 -0500
Message-Id: <200612110155.kBB1tH2a002827@host23.webserver1010.com>
Content-Type: multipart/mixed; boundary="----------=_1165802117-2734-0"
Content-Transfer-Encoding: binary
MIME-Version: 1.0
X-Mailer: MIME-tools 5.411 (Entity 5.404)
From: madavila@e-logicasoftware.com
To: madavila@e-logicasoftware.com
Subject: Hola
```

Tienen la característica de separarse por el caracter dos puntos (:) como en **To:, Subject:, MIME-Version:, X-Mailer:, From:, To:, Subject:** y deben ser interpretadas para actualizar la base de datos de correos recibidos cuyos campos son los contenidos de estos nombres, es decir lo que va después de los dos puntos (:).

La idea es crear una cadena de datos separador por un caracter separador como | (pipe) que al final se le pasarán al programa que actualice el repositorio de Correos_Recibidos, es decir cambiar los dos puntos (:) por el carácter "pipe" (|) y encadenarlos.

Nuevamente las expresiones regulares vienen en nuestra ayuda. Veamos.

```
#-------------------------------------------------------------------------------------------------------
#          Regenera el registro separando por |  los campos claves del protocolo SMTP
#          Por ejemplo, la expresión s/To:/|To|/i lo que hace es buscar (s de search) el caracter To:
#          y lo cambia por |To y así para cada uno de los diferentes campos del SMTP
#          Observe algunos de los campos analizados
#-------------------------------------------------------------------------------------------------------
$temp1 =~ s/X-POP3-Rcpt:/|X-POP3-Rcpt|/i;
$temp1 =~ s/Return-Path:/|Return-Path|/i;
$temp1 =~ s/Received:/|Received1|/i;
$temp1 =~ s/Received:/|Received2|/i;
$temp1 =~ s/Date:/|Date|/i;
$temp1 =~ s/Message-Id:/|Message-Id|/i;
```

```
$temp1 =~ s/Message-ID:/ | Message-Id | /i;
$temp1 =~ s/Content-Type:/ | Content-Type | /;
$temp1 =~ s/Content-Transfer-Encoding:/ | Content-Transfer-Encoding | /i;
$temp1 =~ s/Mime-Version:/ | Mime-Version | /i;
$temp1 =~ s/MIME-Version:/ | Mime-Version | /i;
$temp1 =~ s/Cc:/ | Cc | /i;
$temp1 =~ s/X-Sent-Mail:/ | X-Sent-Mail | /i;
$temp1 =~ s/Reply-To:/ | Reply-To | /i;
$temp1 =~ s/X-Expiredinmiddle:/ | X-Expiredinmiddle | /i;
$temp1 =~ s/X-Mailer:/ | X-Mailer | /i;
$temp1 =~ s/X-Priority:/ | X-Priority | /i;
$temp1 =~ s/From:/ | From | /i;
$temp1 =~ s/To:/ | To | /i;
$temp1 =~ s/Subject:/ | Subject | /i;
$temp1 =~ s/X-Sender-Ip:/ | X-Sender-Ip | /i;
$temp1 =~ s/Organization:/ | Organization | /i;
$temp1 =~ s/Status:/ | Status | /i;
$temp1 =~ s/Content-Language:/ | Content-Language | /i;
$temp1 =~ s/Content-Transfer-Encoding:/ | Content-Transfer-Encoding | /i;
#-------------------------------------------------------------------------------------------------
# Una vez que lee el encabezado ahora lee el cuerpo de cada correo en $temp2
#          usando el método Body
#-------------------------------------------------------------------------------------------------
$temp2 = $pop->Body($i);
$temp1 =~ s/\ | \s/\ | /g;        # Quita caracteres en blanco a la derecha de |
@arr = split(/\ | /,$temp1);      # Crea un arreglo llamado @arr con los campos
                                  # separados por |
```


Al final, va dejando el registro de datos separados por | en la variable $registro11 y procede a hacerle los últimos afinamientos:

```
if ($registro11 =~ /iso-8859/ | | $registro11 =~ /ISO-8859/) {          # Mira si el Subject viene en
                                                                        # iso-8859
      $lineadecodificada = decode_mimewords($registro11);  # y lo decodifica
      $registro11 = $lineadecodificada;                               # sobre si mismo
};
#-------------------------------------------------------------------------------------------------|
# Borra el correo del buzón
#-------------------------------------------------------------------------------------------------|
$pop->Delete($i);
$pop->Close();
```

3.El *spam*

El *spam* o correo indeseado se ha convertido en un verdadero problema mundial puesto que se cuela dentro del correo normal haciendo perder mucho tiempo a los usuarios.

Proviene del interés de extraños en captar correos electrónicos que alimenten las bases de datos comerciales vendibles a las empresas que hacen comercio electrónico y en ocasiones de jóvenes que les gusta jugar con la tecnología.

En e-Génesis he implementado un sistema de *Spam* propio, pero, en esta sección no se entrará en los detalles de su programación sino más bien en el análisis de los resultados.

La experiencia con e-Génesis consistió en añadir al subsistema de administración de los correos una manera de poder tomar un correo indeseado para actualizar un archivo o repositorio de datos llamado *Spam*, el cual guarda algunos datos relacionados con ellos. Estos datos son el correo electrónico del emisor y una palabra clave del tema o *subject* y que el usuario desea interpretarla como una palabra

maliciosa.

No se tomaría todo el *subject*, o tema, porque la probabilidad que este se repita completo entre correos de *spam* es mínima. Lo que se repite son palabras claves como porno, viagra, sexo y muchas más.

Luego, en el software que recibe los correos, se agregó un control al leerlo del agente de correo del servidor para que compare el correo recibido con el repositorio de *spam* de tal manera que si este correo lleva una palabra prohibida, o si el correo del emisor está catalogado como *spam*, no lo reciba.

La teoría inicial implementada en este sistema pretendía investigar la frecuencia de ocurrencias, estudiar si la contundencia del rechazo haría disminuir la recepción futura de estos correos y la observación analítica de los resultados para emitir algunas opiniones que pudieran ser constructivas con respecto de los patrones de este tipo de correo indeseado. Esa era la teoría. Ahora, ¿qué pasó en la práctica?.

Las estadísticas finales tienen una distribución muy aleatoria y para tratar de darles un sentido se presentan dos cuadros.

El primero muestra las frecuencias a través de una serie de variables que son::

Número de correos detectados como "*Spam* por correo": el "*spam* por correo" se refiere a un contador que muestra cuántas veces aparece en el *spam* un mismo correo de origen. Como se puede observar adelante, en el cuadro, el número de repeticiones del mismo correo es muy pequeño demostrando que el origen, es decir el correo de quien lo envía es engañoso y no necesariamente es un correo válido dentro del contexto de Internet, ocultando por tanto la identidad del emisor. Esto enseña que no vale la pena hacer esfuerzos por interpretar de dónde viene el *spam*, basados en el correo recibido.

El primero muestra las frecuencias a través de una serie de variables que son:

Número de correos detectados como "*Spam* por correo": el "spam por correo" se refiere a un contador que muestra cuántas veces aparece en el spam un mismo correo de origen. Como se puede observar adelante, en el cuadro, el número de repeticiones del mismo correo es muy pequeño demostrando que el origen, es decir el correo de quien lo envía es engañoso y no necesariamente es un correo válido dentro del contexto de Internet, ocultando por tanto la identidad del emisor. Esto enseña que no vale la pena hacer esfuerzos por interpretar de dónde viene el spam, basados en el correo recibido.

Número de correos detectados como "*Spam* por tema": se refiere al número de veces que se captaron correos de spam por palabras similares es decir que se repitieron en diversos correos de spam, mostrando una tendencia de uso de dichas palabras.

Número de palabras diferentes: número de palabras diferentes que aparecieron en todos los correos.

Número de palabras médicas, de sexo o financieras: se refieren a palabras clasificables dentro de esos géneros o significados. Como se puede observar, en estas categorías caen un alto porcentaje. La investigación deberá continuar analizando la sintaxis y otras posibles clasificaciones que nos indiquen tendencia para definir patrones.

Número de correos recibidos que en el Tema traen incluido el nombre del correo del receptor: este descubrimiento es interesante ya que muestra que si en el tema o subject aparece el nombre del usuario que recibe el correo es casi siempre un spam. Por ejemplo, si el correo del receptor es juan@dominio.com y en el tema viene la palabra juan, es un spam.

Palabras you, your, new, low: son palabras que tienen el 21.67 % de apariciones en el spam.

ver tabla 1 en la siguiente página

Veamos los resultados en la tabla 1:
Nota: se usa el punto para indicar miles y la coma para los decimales.

Concepto medido	Frecuencia	% sobre el total de correos
Número de correos detectados como spam por correo	112	
Número de correos detectados como spam por tema	18.725	
Número de palabras diferente clasificadas como spam	574	
Spam de palabras médicas	1.090	5.8211%
Spam de palabras de sexo	897	4.7904%
Spam de palabras financieras	817	4.3632%
Spam por tener el nombre del correo en el tema	233	1.2443%
Palabra you, our	2.977	15.8985%
Palabra new	568	3.2334%
Palabra low	512	2.7343%
Total		37.8852%

Tabla Nº 1. Frecuencias.

El siguiente cuadro, tabla 2, presenta análisis de los resultados tratando de buscar algunas conclusiones:

Referencia	Palabras	% de las ocurrencias	No. de ocurrencias
1	you	7,00%	1.351
2	your	5,00%	975
3	our, new	3,00%	1.219
4	low	2,00%	512
5	Get, pharmeceutical, with, deliver, pharm, viagra, madavila, now, stock, more, watch, here, weight (17,62%)	1,00%	3.299
6	Otras	Entre 0,0053% y 1,0%	11.369
		Total	18.725

Tabla Nº 2. Distribución porcentual de apariciones.

Para mostrar la disparidad de las apariciones de palabras de "Spam por tema" se muestra el cuadro anterior en donde los conceptos son:

Porcentaje de las ocurrencias: se trata de medir de alguna manera las palabras cuyas frecuencias son más significativas, por ejemplo en la primera línea se puede deducir que la palabra you aparece 1.351 veces, con un 7% sobre el total.

En la segunda línea se observa que la palabra your tiene 975 apariciones, conformando el rango del 5% al 6% de apariciones sobre el total.

En la quinta línea se puede deducir que palabras dentro del rango del 3% al 4% de apariciones sobre el total con 1.219 apariciones son las palabra *our y new*.

En la sexta línea se puede deducir que la palabra *low* tiene 512 apariciones sobre el total conformando el rango del 2% al 3%.

En la séptima línea se puede deducir que las trece palabras *Get, Pharmaceutical, With, Deliver, Pharm, Viagra, madavila, Now, Stock, More, Watch, Here, Weight* conforman el rango entre el 1% al 2% de apariciones sobre el total, con 3.299 apariciones para todas es decir el 17,62%.

En la octava línea se puede deducir que palabras dentro del rango del 0% al 1% de apariciones sobre

el total con 11.369 apariciones son 556 palabras diferentes.

Antes de desarrollar el sistema de *spam* de e-Génesis y de hacer estas mediciones las expectativas sobre los resultados no eran predecibles y solamente existía en interés de trabajar en el manejo de estas tecnologías y de estos problemas de seguridad informática.

3.1 Conclusiones sobre el patrón de spam personal

Estas conclusiones nacen de la experiencia del análisis de las diez y ocho mil setecientas veinte y cinco (18.725) apariciones de palabras catalogadas como spam y no se pretende hacer generalizaciones contundentes pero sí mostrar la existencia de patrones, como éste, que tipifican a un usuario que puede representar a muchos equivalentes.

Caracterización del usuario (es decir el autor):

Ingeniero de sistemas, directivo, académico, empresarial con un altísimo manejo de su gestión a través del Internet y con el uso del correo como herramienta básica. Entorno centrado mayormente con interlocutores de habla hispana.

El 37,88 % del spam por tema viene por el uso de las 18 palabras *You, Your, Our, New, Low, Get, Pharmaceutical, With, Deliver, Pharm, Viagra, madavila, Now, Stock, More, Watch, Here, Weight.*

El 62,11% del spam viene por 556 palabras diferentes que aparecen 11.369 veces y ocupan individualmente entre el 0% al 1% de apariciones sobre el total dando una muestra demasiado dispersa como para que sobre ellas se haga un análisis puntual que ofrezca una ayuda más específica, como las anteriormente mencionadas.

Necesidad de ejercer un control de rechazo sobre esta muestra dispersa de manera particular para cada una de ellas pues si bien cada una está dentro de un valor cercano al 0% hasta el 1%, la suma de todos ellos da 62,11%

El porcentaje de protección obtenido se refleja en esas cifras: diariamente llegan a este buzón cerca de 100 correo de los cuales el 6% contiene palabras nuevas de spam que van aumentando los filtros y disminuyendo esa frecuencia cada vez más.

Esta dispersión determina que, para lograr un alto nivel de protección, se necesita disponer de una base de datos de palabras filtradas basadas en prototipos completos, como este.

No basta con saber que unas pocas palabras son *spam* sino usar todo el diccionario de palabras para asegurar una protección aceptable.

Se demuestra también que la gran mayoría de palabras *spam* son en inglés y que el origen del no es de usuarios locales sino extranjeros o, por lo menos, a ellos va dirigido el mayor ataque. Si embargo, debe tenerse en cuenta que el análisis de idioma no se ha tratado en esta investigación de manera formal.

Se demuestra que el *spam* no es un problema de fácil protección ya que los algoritmos implementables son determinísticos.

Se determina que el uso de expresiones regulares es de gran utilidad ya que un filtro del estilo v(a-z)*i(a-z)*a(a-z)*g(a-z)*r(a-z)*a(a-z)* filtrará todas las formas de la palabra viagra con vocales repetidas como viiagra, viiagraaa, etc. facilitando el filtro con una sola expresión. Válida mientras no sea amigo de ningún señor de apellido Villagrán.....

Que expresiones como ^re$ o ^hi$ rechazarán todos los correos cuyo Tema lleve sólo la palabra re o hi lo cual es frecuente.

Para aquellos interesados en estudiar mas detalles pueden acceder a la base de datos de spam completa sobre la cual se basó esta investigación en:
http://e-logicasoftware.com/tutoriales/base_de_datos_spam/Base_de_datos_spam_investigacion_e-genesis.xls

Tomando un correo de *spam* al azar de los que han llegado a este buzón, el encabezado se ve así:

```
X-POP3-Rcpt: madavila@e-logicasoftware.com
Received: from i141231.upc-i.chello.nl (i141231.upc-i.chello.nl (62.195.141.231))
    by host23.webserver1010.com (8.12.11.20060614/8.12.10) with ESMTP id kBC34Oii026157
    for <madavila@e-logicasoftware.com>; Mon, 11 Dec 2006 22:04:24 -0500
Received: from (106.195.189.163) (port=14352 helo=znfkgdi)
    by baelsgx.frgudf.r.12hs.com with psmtp
    id uULZoa-66A5f8-00
  for <madavila@e-logicasoftware.com>; Tue, 12 Dec 2006 04:04:47 +0100
Message-ID: <000b01c71d9a$38ab0690$00000000@gindrapassage9>
From: "Culture" <arrange@baelsgx.frgudf.r.12hs.com>
To: madavila@e-logicasoftware.com
Subject: filtering kind builtin ProtoWall
Date:  Tue, 12 Dec 2006 04:04:20 +0100
MIME-Version: 1.0
Content-Type: multipart/related;
type="multipart/alternative";
boundary="----=_NextPart_000_0007_01C71DA2.9A0D9F80"
X-Priority: 3
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook Express 6.00.2900.2180
X-MimeOLE: Produced By Microsoft MimeOLE V6.00.2900.2180
```

Lo anterior permitirá hacer otro tipo de análisis como por ejemplo el estudio de los IP's del origen que bien podría determinar una manera de saber de dónde provienen los correos spam así como un análisis usando los métodos Bayesianos, tan usado en las metodología para este tipo de análisis, pero esto será parte de otra investigación futura.

Si esto fuera una novela de misterio, terminaría diciendo: ¡Continuará!

Bibliografía

(1) David Till, Teach yourself Perl 5, Sams publishing, 1996
(2) Ben Laurie & Peter Laurie, Apache: The Definitive Guide, Second Edition, O'Reilly,1999
(3) Thomas A. Powell, Manual de Referencia HTML, McGrawHill, 1998
(4) Robert McDaniel, CGI Manual of Style, Macmillan Computer Publishing USA,1996
(5) Sriram Srinivasan, Advanced Perl Programming, O'Reilly,1997
(6) Clinton Wong, Client Programming with Perl Automating Tasks on the Web 1st Edition, O'Reilly, 1997
(7) Shishir Gundavaram, CGI Programming on the World Wide Web , O'Reilly,1996
(8) Wikipedia. http://es.wikipedia.org
(9) Bryan Costales, sendmail, O'Reilly, 1997