hakin9

¿Cómo se envía un spam?

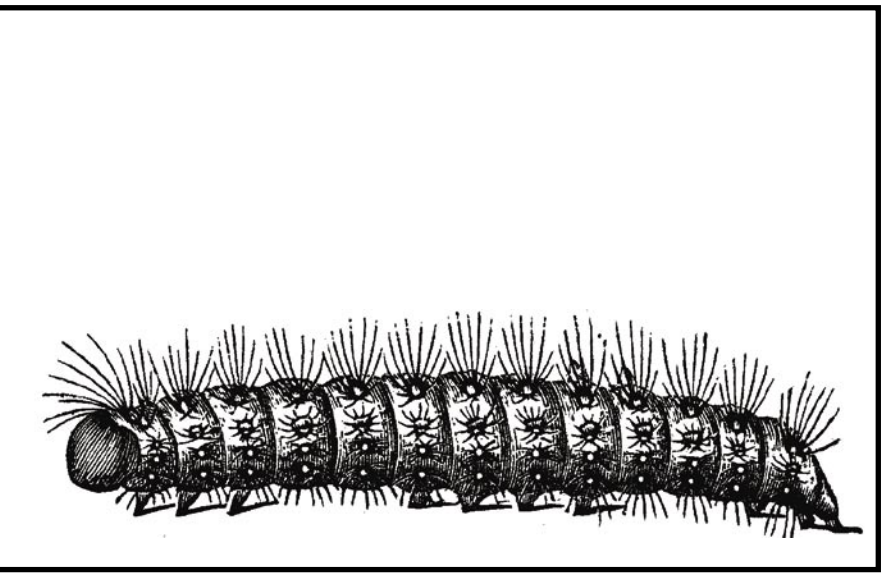
Tomasz Nidecki

Artículo publicado en el número 2/2004 de la revista “Hakin9”

Revista „Hakin9”, Wydawnictwo Software, ul. Lewartowskiego 6, 00-190 Warszawa, piotr@software.com.pl

¿Cómo se envía un spam?

Tomasz Nidecki



Con frecuencia los spammers se aprovechan de las vulnerabilidades de los sistemas. Los costes y los problemas relacionados con el envío, a menudo, de decenas o de cientos de miles de mensajes son trasladados a terceros. Enterémonos en qué se basan las técnicas empleadas por los spammers y cómo protegernos de ellos.

El envío masivo de correo electrónico absorbe muchísimos recursos. Para llevarlo a cabo, es imprescindible una conexión rápida y un ordenador dedicado. Incluso si el spammer dispone de tales recursos, el envío puede tardar varias horas. Los proveedores de internet no se entusiasman, por lo general, si de repente sus conexiones son utilizadas para el envío de spam y el spammer puede perder conexión con la red antes de poder enviar la mayor parte de los mensajes. Si se le detiene puede sufrir graves consecuencias jurídicas o financieras.

Los spammers con el objetivo de acelerar y perfeccionar el envío emplean dos métodos básicos. El primero está basado en la minimización del tiempo requerido para enviar el mensaje. Es conocido como *fire and forget*, es decir, ¨envía y olvida¨. Con este método el ordenador que envía los spams no espera respuesta de los servidores con los cuales se contacta. El segundo método se basa en el robo de los recursos de personas ajenas que por algún motivo han configurado mal sus sistemas o han sido víctimas de virus. La mayoría de los costes y muchas veces, también la responsabilidad por el envío de spams recae sobre ellos, dejando a los spammers impunes.

Protocolo SMTP

Para comprender los métodos utilizados por los spammers es necesario saber cómo funciona el protocolo más común para el envío del correo electrónico: SMTP. Igual que la mayoría de los protocolos empleados en Internet, está basado en simples comandos textuales.

En nuestro artículo aprenderás...

- de qué manera los spammers envían los spams (empleando ordenadores de personas inocentes),
- cómo proteger tu servidor ante spammers,
- cómo funciona el protocolo SMTP,
- qué es *open relay*, *open proxy* y *zombie*.

Qué deberías saber...

- cómo emplear las herramientas básicas del sistema Linux.

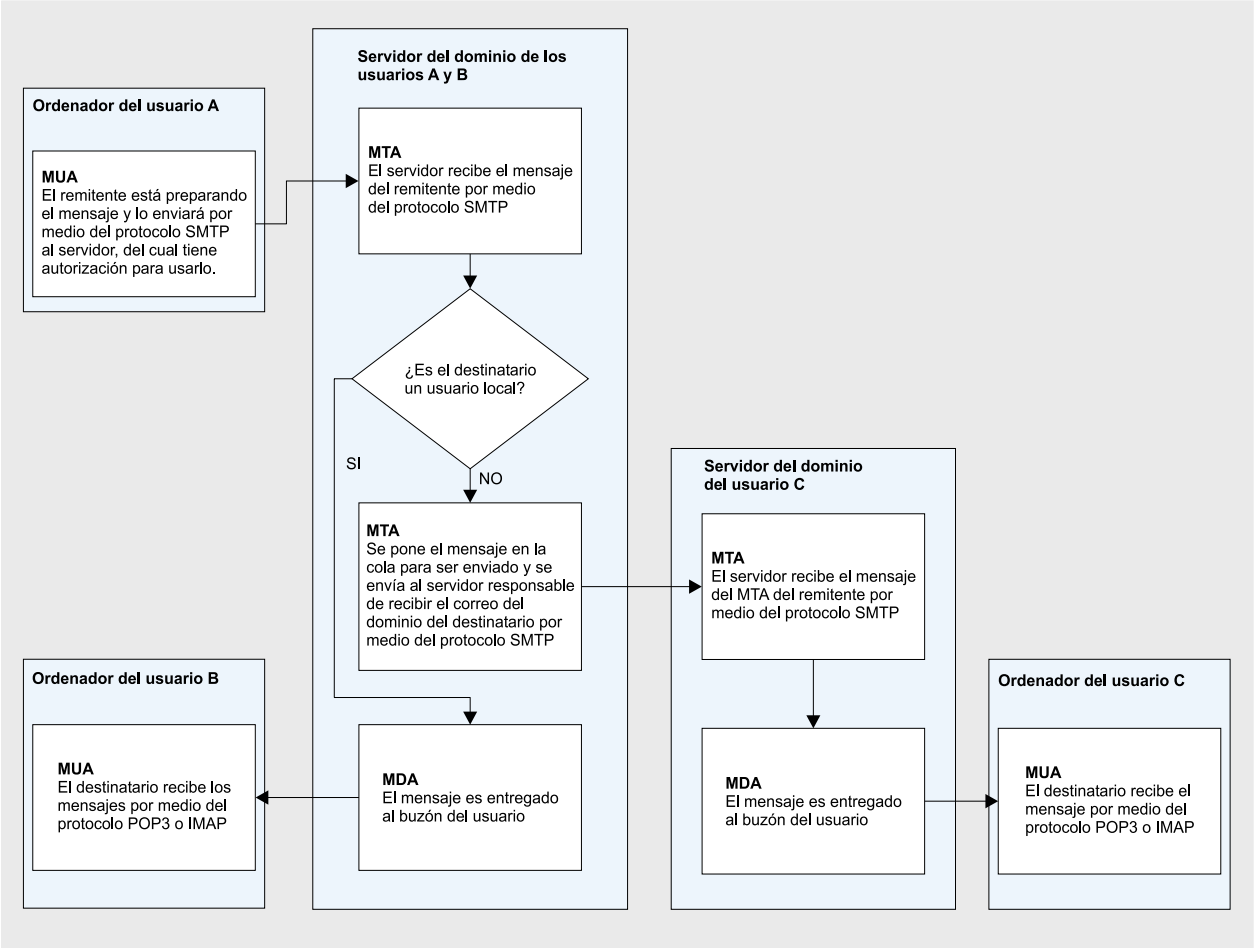


Figura 1. *Etapas del envío de correo electrónico*

Historia del SMTP

El programa SNDMSG (*Send Message*) es el precursor del SMTP, empleado en el año 1971 por Ray Tomlinson (junto con su propio proyecto *CYPNET*) para la creación de una aplicación que permitiera el envío del correo electrónico dentro de la red *ARPANET*. Un año más tarde, el programa utilizado en Arpanet para el envío de ficheros (FTP), fue ampliado con el comando `MAIL` y `MLFL`. Hasta el año 1980 el correo era enviado por medio de FTP. Fue en aquel año cuando nació el primer protocolo estándar de correo electrónico, MTP (*Mail Transfer Protocol*), descrito en el documento RFC 772. MTP sufrió varias modificaciones (RFC 780, 788) y en el año 1982, en el RFC 821, Jonathan B. Postel describió *Simple Mail Transfer Protocol*.

Desgraciadamente, el SMTP en su forma básica no cumplió todas esperanzas. De ahí que surgieron muchos documentos que describían la extensión del protocolo. Entre los más importantes podemos destacar:

- RFC 1123 – exigencias para los servidores de Internet (también abarca SMTP),
- RFC 1425 – introducción del estándar de las extensiones del protocolo SMTP – ESMTP,
- RFC 2505 – conjunto de sugerencias sobre protección antispam de los servidores,
- RFC 2554 – autorización de las conexiones – introducción del comando `AUTH`,

El actual estándar SMTP fue descrito en el año 2001 en RFC 2821. La colección RFC está en nuestro CD.

Etapas del envío del correo

El correo electrónico es enviado en varias etapas (véase Figura 1). Para comprenderlo imaginémonos que queremos enviar un mensaje de *hakin9@hakin9.org* a *nobody@example.com*. El usuario que envía el mensaje utiliza el programa *Mozilla Thunderbird* en una red local, el destinatario – *Outlook Express* por medio de la conexión telefónica tipo *dial-up*.

En la primera etapa, el programa *Mozilla Thunderbird* contacta con el servidor SMTP señalado en las opciones de la cuenta del usuario *hakin9@hakin9.org – mail.software.com.pl*. El mensaje está enviado al servidor a través del protocolo SMTP. En la segunda etapa, *mail.software.com.pl* mira en los registros de los servidores DNS. Se entera de que el responsable

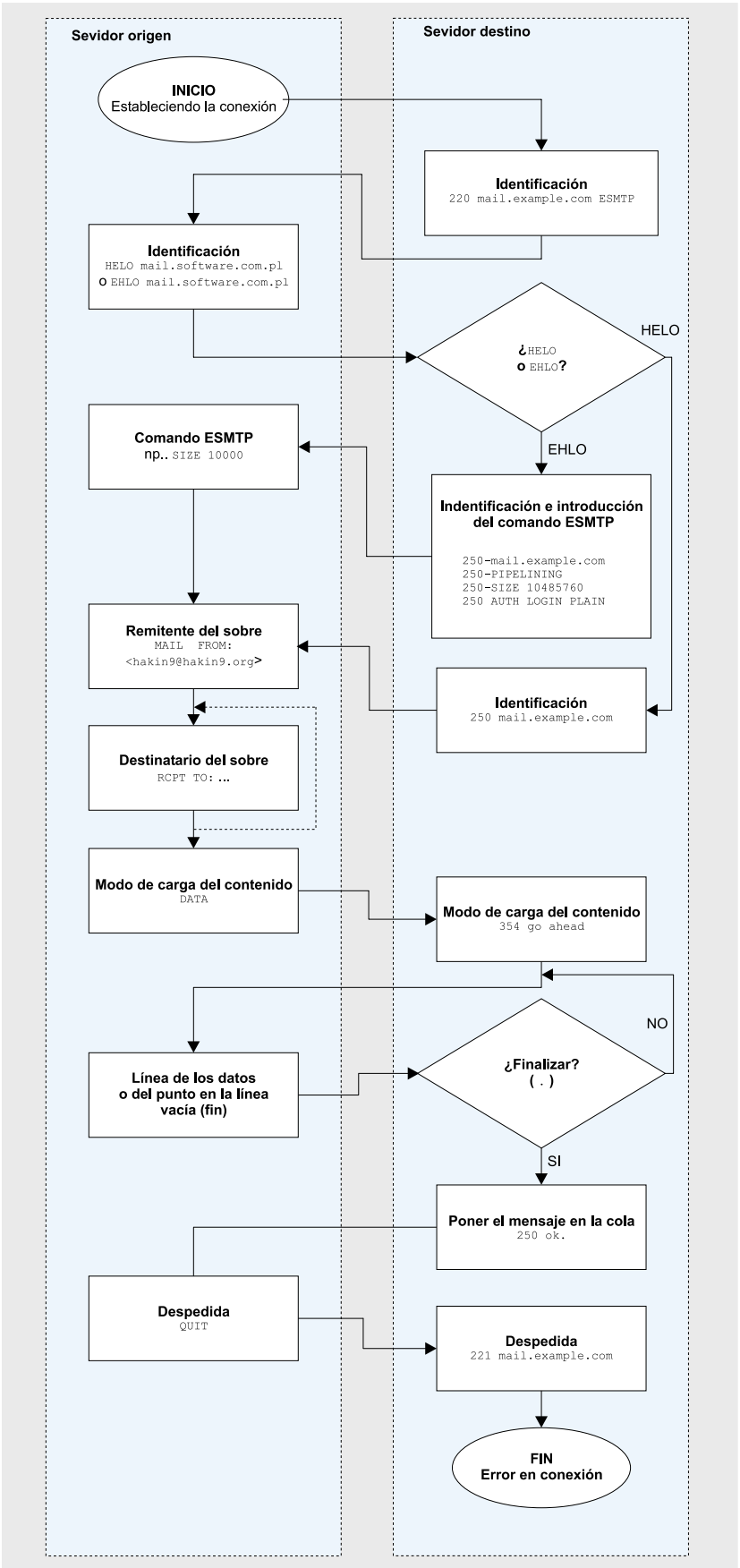


Figura 2. *Etapas de la comunicación por medio de SMTP*

de la recepción del correo del dominio *example.com* es *mail.example.com*. Toda esta información se encuentra en el registro MX (*Mail Exchanger*) publicado por la DNS responsable del domino *example.com* (podemos obtenerla por medio de los programas *host* o *dig*: `host -t mx example.com` o `dig example.com.mx`).

En la tercera etapa, *mail.software.com.pl* conecta con *mail.example. com* y le pasa el mensaje. En la siguiente etapa, *mail.example.com* entrega el mensaje recibido al buzón local de correo del usuario *nobody*. En la última etapa, el usuario del buzón *nobody* conecta por medio de la conexión *dial-up* con el servidor *mail.example.com* a través del protocolo POP3 (o IMAP) con ayuda del programa *Outlook Express* y recoge el mensaje. Puede ocurrir que el mensaje recorra un camino más largo. El remitente podría utilizar servidores de correo separados, p. ej. *recepcion.software.com.pl* y *envío.software.com.pl*. De este modo el mensaje pasaría de los usuarios a través de *recepcion.software.com.pl*, entregado a *envío.software.com.pl* y, a continuación, enviado a *mail.example.com*. Del mismo modo en el caso de *mail.example.com* en la recepción y entrega del correo al usuario puedenestar implicados varios servidores.

¿El sucesor de SMTP?

Dan Bernstein, autor del *qmail*, desarrolló el protocolo de nombre QMTP (*Quick Mail Transfer Protocol*), que debería sustituir a SMTP. QMTP elimina muchos de los problemas que se presentan en SMTP, pero es incompatible con su antecesor. Desgraciadamente, no se ha implementado en ninguna parte, a parte del propio *qmail*.

Más información sobre QMTP: *http://cr.yp.to/proto/qmtp.txt*

Programas que toman parte en el envío del correo

En el envío del correo participan varios programas:

- El programa usado por el usuario final, que sirve no sólo para recibir y enviar, sino también para leer y escribir emails, es conocido como MUA – *Mail User Agent*. Ejemplos: *Mozilla Thunderbird, Outlook Express, PINE, Mutt;*
- La parte del servidor responsable de la comunicación con los usuarios (recepción del correo) y del envío y recepción del correo de otros servidores es conocida como MTA – *Mail Transfer Agent*. Los más conocidos son: *Sendmail, qmail, Postfix, Exim;*
- La parte del servidor responsable de la entrega del correo al usuario local se llama MDA – *Mail Delivery Agent*. Ejemplos de MDA autónomos: *Maildrop, Procmail*. La mayoría de los MTA poseen mecanismos propios de entrega del correo local a los usuarios, por tanto no siempre es necesario emplear MDA adicionales.

Etapas de la comunicación en SMTP

El envío de un mensaje con la ayuda del SMTP está dividido en varias etapas. He aquí un ejemplo de una sesión SMTP entre los servidores *mail.software.com.pl* y *mail.example.com*. Los datos enviados por *mail.software.com.pl* están marcados con el símbolo `>` y los datos recibidos de *mail.example.com*: `<`. Tras el establecimiento de la conexión, *mail.example.com* se presenta así:

```
< 220 mail.example.com ESMTP Program
```

informando que su nombre completo de host (FQDN) es *mail.example.com*. Asimismo, nos enteramos de que podemos utilizar los comandos ESMTP (del SMTP extendido: véase el cuadro) y que el MTA utilizado es *Program*. El nombre del programa es opcional, algunos MTA, por ej.: *qmail*, no lo ofrecen.

Nos presentamos así:

```
> HELO mail.software.com.pl
```

como respuesta obtenemos:

```
< 250 mail.example.com
```

esto significa que *mail.example.com* está listo para recibir el correo. A continuación introducimos la dirección de correo electrónico del remitente del email a la que podremos recibir eventuales mensajes devueltos:

```
> MAIL FROM:<hakin9@hakin9.org>
< 250 ok
```

Introducimos las direcciones de los destinatarios:

```
> RCPT TO:<test1@example.com>
< 250 ok
> RCPT TO:<test2@example.com>
< 250 ok
> RCPT TO:<test3@example.com>
< 250 ok
```

Luego, tras el comando `DATA` transferimos las cabeceras y el contenido del mensaje. Separamos las cabeceras del contenido con una línea vacía y terminamos el mensaje con un punto en la siguiente línea:

```
> DATA
< 354 go ahead
> From: nadie@hakin9.org
> To: todos@example.com
> Subject: Nada
>
> Esto es una prueba
> .
< 250 ok 1075929516 qp 5423
```

Después de enviar el mensaje, podemos terminar la conexión:

```
> QUIT
< 221 Bye
```

El servidor no siempre es capaz de llevar a cabo nuestras ordenes. Si obtenemos un código que comience con la cifra 4 (código de la serie 4xx) eso significa que el servidor temporalmente rechaza la recepción del mensaje.

Tabla 1. *Lista de los comandos del protocolo SMTP utilizados con más frecuencia:*

Comando	Descripción
HELO <FQDN>	Presentarse al servidor
EHLO <FQDN>	Presentarse al servidor conectado con la petición de darnos un listado de los comandos ESMTP disponibles
MAIL FROM:<dirección>	Introducir la dirección de correo electrónico del remitente del email a la que podremos recibir eventuales mensajes devueltos
RCPT TO:<dirección>	Introducir la dirección del destinatario del mensaje
DATA	Pase al modo de recepción del contenido del mensaje
AUTH <mecanismo>	Autorización de la conexión (ESMTP); entre los mecanismos más conocidos tenemos: *LOGIN, PLAIN* y *CRAM-MD5*

Podéis encontrar una lista extensa de comandos SMTP y ESMTP en la siguiente dirección: *http://fluffy.codeworks.gen.nz/esmtp.html*

Listado 1. *El open relay más sencillo*

```
$ telnet lenox.designs.pl 25
< 220 ESMTP xenox
> helo hakin9.org
< 250 xenox
> mail from:<hakin9@hakin9.org>
< 250 Ok
> rcpt to:<nobody@example.com>
< 250 Ok
> data
< 354 End data with
 <CR><LF>.<CR><LF>
> Subject: test
>
> Esto es una prueba
> .
< 250 Ok: queued as 17C349B22
> quit
< 221 Bye
```

Deberíamos intentar enviarlo un poco más tarde. Si obenemos un código que comience con la cifra 5 eso significa que el servidor definitivamente rechaza la recepción de nuestro correo e intentos posteriores carecen de sentido. Un listado con los comandos más importantes y los códigos obtenidos del servidor SMTP se ilustra en las Tablas 1 y 2.

Servidores *open relay*

Cuando fue creado el protocolo SMTP, no existía el problema del spam y cada usuario tenía la capacidad de utilizar cualquier servidor para enviar sus mensajes al mundo. Ahora, cuando los spammers buscan la oportunidad de aprovecharse de algún servidor y enviar miles de mensajes, este método no es aconsejable. Los servidores que permiten el envío de correo al mundo sin autorización son conocidos como *open relay.*

Cada servidor que permite a usuarios no autorizados el envío de correo, tarde o temprano cae en manos de los spammers, lo que puede acarrear consecuencias muy serias. Primero: porque puede provocar una reducción de eficacia del servidor, el cual en vez de recibir y entregar mensajes a usuarios autorizados, enviará spam. Segundo: el proveedor de internet puede anular el contrato con motivo del uso del servidor para fines ilegales o inmorales. Tercero: la dirección IP del servidor terminará en las listas negras y muchos servidores dejarán de recibir sus mensajes (la eliminación de la IP de muchas listas negras es muy difícil, incluso a veces imposible).

Empleo del *open relay*

Comprobemos lo fácil que es emplear el *open relay* para el envío de spam. Como ejemplo tomaremos uno de los servidores polacos que está mal configurado, utilizado por los spammers: *lenox.designs.pl.* Como podemos observar en el Listado 1, en este caso no tuvimos que realizar pasos complicados para enviar un mensaje. El servidor considera a todos que conectan con él como usuarios autorizados para enviar correspondencia. Es el tipo del servidor *open relay* más peligroso, puesto que es el más fácil de aprovecharse.

Existen otros *open relay,* un poco más difíciles de utilizar por los spammers. Como ejemplo podemos tomar uno de los servidores de correo que está mal configurado del portal polaco *O2: kogut.o2.pl.*

Listado 2. *Servidor open relay que permite el envío sólo a usuarios existentes hakin9@hakin9.org*

```
$ telnet kogut.o2.pl 25
< 220 o2.pl ESMTP Wita
> helo hakin9.org
< 250 kogut.o2.pl
> mail from:<ania@o2.pl>
< 250 Ok
> rcpt to:<
< 250 Ok
> data
< 354 End data with
 <CR><LF>.<CR><LF>
> Subject: test
>
> Esto es una prueba
> .
< 250 Ok: queued as 31B1F2EEA0C
> quit
< 221 Bye
```

Como observamos en el Listado 2, bastó con adivinar el nombre del usuario para hacerse pasar por él y enviar el mensaje. En el caso de algunos servidores es suficiente dar el nombre del dominio local, incluso no tiene que existir el usuario por el que nos hacemos pasar.

Observamos una situación similar en el Listado 3: otra vez nos

Tabla 2. *Lista de los códigos de respuesta más importantes:*

Código	Descripción
220	Servicio activo: el servidor nos da la bienvenida informando que le podemos enviar el comando
250	Comando aceptado
354	Se puede empezar a introducir el contenido del mensaje
450	El buzón del usuario está ocupado por el momento (p. ej.: bloqueado por otro proceso)
451	Error local durante el procesamiento del correo
452	No hay espacio en el disco por el momento
500	No existe el comando
501	Error en el comando o en sus parámetros
502	El comando no se implementó
550	Buzón del usuario inaccesible
552	Se ha excedido el límite de espacio en el disco

Podéis encontrar una lista completa de códigos y de normas de crearlos en RFC 2821 (en nuestro CD).

¿Cómo no llegar a ser el *open relay*?

El protocolo SMTP permite:

- recibir el correo del usuario (MUA) y enviarlo a otro servidor (MTA),
- recibir el correo de otro servidor (MTA) y enviarlo al usuario local (MUA),
- recibir el correo de un servidor (MTA) y enviarlo a otro servidor (MTA).

No hay ninguna diferencia en la manera de enviar los mensajes por medio de MUA y MTA.

Lo más importante es si la dirección IP del destinatario es de confianza (por ej.: en la red local) y si el remitente está en el dominio local o externo.

El envío del correo fuera de nuestro servidor se conoce como *relaying*. No podemos permitir el *relaying* sin autorización para que los spammers no puedan usar nuestro servidor para llevar a cabo sus actividades. Por lo tanto, durante la configuración del servidor SMTP hay que tener en cuenta lo siguiente:

- Si el mensaje está destinado a uno de los dominios atendidos por nuestro servidor tiene que ser recibido sin autorización.
- Si el mensaje es enviado por un usuario local (desde el MUA en el servidor) en la red local o desde una IP permanente autorizada y el remitente es usuario externo, el mensaje puede ser recibido sin autorización (sin embargo, se recomienda su autorización).
- Si el mensaje es enviado por un usuario externo (por ej.: de una IP dinámica) y el destinatario es un usuario externo, el mensaje puede ser recibido sin autorización.

Listado 3. *Servidor multistage open relay que permite el envío sólo a los usuarios existentes*

```
$ telnet smtp.poczta.onet.pl 25
< 220 smtp.poczta.onet.pl ESMTP
> helo hakin9.org
< 250 smtp.poczta.onet.pl
> mail from:<ania@buziaczek.pl>
< 250 2.1.0 Sender syntax Ok
> rcpt to:<hakin9@hakin9.org>
< 250 2.1.5 Recipient address ↵
 syntax Ok; ↵
 rcpt=<hakin9@hakin9.org>
> data
< 354 Start mail input; ↵
 end with <CRLF>.<CRLF>
> Subject: test
>
> Esto es una prueba
> .
< 250 2.6.0 Message accepted.
> quit
< 221 2.0.0 smtp.poczta.onet.pl
  Out
```

encontramos con el servidor de correo de uno de los portales polacos más grandes, esta vez *Onet* (lo curioso es que *Onet* declara ser un combatiente activo de los spams...). Es lo que se conoce como *multistage open relay,* es decir, el mensaje es recibido por una IP y enviado por otra.

Para enterarnos de esto tenemos que analizar las cabeceras `Received` (véase el cuadro) del mensaje recibido. Como observamos en el Listado 4, el mensaje fue recibido por *ps8.test.onet.pl* (213.180.130.54) y enviado al destinatario por *smtp8.poczta.onet.pl* (213.180.130.48). Esto dificulta la detección de que el servidor fue configurado como *open relay*, pero no molesta en absoluto utilizarlo para el envío de spams.

Otro tipo de *open relay* son los servidores con mal configurada autorización del remitente (SMTP-AUTH). Permiten el envío de correo tras introducir cualquier login y contraseña. Esto a menudo les sucede a los administradores principiantes del *qmail* que por algún motivo no leyeron la documentación del parche SMTP-AUTH e invocaron de modo incorrecto el *qmail-smtpd*.

El programa *qmail-smtpd* con el parche incorporado requiere tres argumentos: el FQDN, el programa que verifica la contraseña (compatible con *checkpassword*) y el parámetro adicional del programa que verifica la contraseña (por ejemplo, `qmail-smtpd hakin9.org /bin/checkpassword /bin/true`). Un error frecuente es poner `/bin/true` como segundo parámetro, ya que entonces la verificación de la contraseña es siempre exitosa (independientemente del login y contraseña introducidos). De la misma manera, el spammer puede intentar un ataque de diccionario (*ing. dictionary attack*), por lo tanto recomendamos que las contraseñas de los usuarios utilizadas para la autorización del SMTP no sean triviales.

Cabeceras *Received*

Las cabeceras `Received` son elementos obligatorios de cada mensaje. Determinan el camino desde el remitente hasta el destinatario (cuanto más arriba esté colocada la cabecera, más cerca estará del servidor del destinatario). Las cabeceras son añadidas automáticamente por los servidores de correo. Sin embargo, el spammer puede agregar sus propias cabeceras, tratando de borrar su identidad y echar la culpa a otra persona. Las únicas cabeceras siempre verdaderas son las que coloca el servidor del destinatario (las de más arriba). Las demás pueden ser falsas. Gracias a las cabeceras `Received` podemos identificar la IP del remitente real del mensaje y también reconocer si el mensaje fue enviado por medio de *open relay* u *open proxy*. No obstante, el análisis de las cabeceras no es una tarea fácil, ya que no existe un método estándar para su formulación y cada servidor de correo muestra la información a su manera.

Servidores *open proxy*

Otro tipo de servidores mal configurados que pueden ser empleados por los spammers son los *open proxy*, o sea, los servidores proxy a los que pueden conectar usuarios

Listado 4. *Cabeceras Received del correo recibido del servidor multistage open relay*

```
Received: from smtp8.poczta.onet.pl (213.180.130.48)
 by mail.hakin9.org with SMTP; 23 Feb 2004 18:48:11 -0000
Received: from mail.hakin9.org ([127.0.0.1]:10248 "helo hakin9.org")
 by ps8.test.onet.pl with SMTP id <S1348420AbUBWSrW>;
 Mon, 23 Feb 2004 19:47:22 +0100
```

no autorizados. Los servidores *open proxy* pueden funcionar en base a una gama extensa de programación y protocolos. El protocolo más frecuente es HTTP-CONNECT, pero hay algunos *open proxy* que permiten la conexión a través de los protocolos HTTP-POST, SOCKS4, SOCKS5 y otros.

Un *open proxy* puede ser usado por el spammer de manera idéntica que *open relay* para enviar correspondencia no autorizada. Además, muchos *open proxy* permiten ocultar su dirección IP. Un proxy de esta índole es un bocado exquisito para los spammers.

Empleo del *open proxy*

En el Listado 6 está ilustrado el empleo de un *open proxy* que permite conexión por medio de HTTP-CONNECT en el puerto 80. La mayor parte de la conexión es la comunicación con *open relay* (los mismos comandos que en el Listado 2). Sin embargo, antes de conectarnos con el servidor SMTP, establecemos contacto con *open proxy* y con su ayuda nos conectamos con MTA. Durante la conexión declaramos que la comunicación se llevará a cabo por medio del protocolo HTTP/1.0; sin embargo, no lo utilizamos.

Para el spammer lo más confortable es encontrar un servidor *open relay* que al mismo tiempo tenga instalado un servidor de correo local. En la mayoría de los casos el MTA sin autorización acepta las conexiones del proxy local tratándolas del mismo modo que las de los usuarios locales. En estas situaciones el spammer no tiene que conocer ningún servidor *open relay* y puede tranquilamente llevar a cabo su actividad a costa y respo-

Listado 5. *Servidor open relay con configuración errónea SMTP-AUTH*

```
$ telnet mail.example.com 25
< 220 mail.example.com ESMTP
> ehlo hakin9.org
< 250-mail.example.com
< 250-PIPELINING
< 250-8BITMIME
< 250-SIZE 10485760
< 250 AUTH LOGIN PLAIN CRAM-MD5
> auth login
< 334 VXNlcm5hbWU6
> cualquier cosa
< 334 UGFzc3dvcmQ
> cualquier cosa
< 235 ok, go ahead (#2.0.0)
> mail from:<hakin9@hakin9.org>
< 250 ok
> rcpt to:<nobody@nowhere.com>
< 250 ok
> data
< 354 go ahead
> Subject: test
>
> Esto es una prueba
> .
< 250 ok 1077563277 qp 13947
> quit
< 221 mail.example.com
```

Listado 6. *Servidor open proxy empleado para el envío anónimo de correspondencia por open relay*

```
$ telnet 204.170.42.31 80
> CONNECT kogut.o2.pl:25 HTTP/1.0
>
< HTTP/1.0 200↵
 Connection established
< 220 o2.pl ESMTP Wita
> helo hakin9.org
< 250 kogut.o2.pl
> mail from:<ania@o2.pl>
< 250 Ok
> rcpt to:<hakin9@hakin9.org>
< 250 Ok
> data
< 354 End data with↵
 <CR><LF>.<CR><LF>
> Subject: test
>
> Esto es una prueba
> .
< 250 Ok: queued as 5F4D41A3507
> quit
< 221 Bye
```

¿De dónde toman los spammers las direcciones *open relay* y *open proxy*?

Buscar por cuenta propia servidores mal asegurados puede resultar bastante problemático. Sin embargo, basta con recibir un spam enviado por *open relay* u *open proxy* para poder utilizarlo. Para verificar si bajo la dirección IP sospechosa hay un *open relay,* podemos usar el script *rlytest* (*http://www.unicom.com/sw/rlytest/*), en cambio, para detectar un *open proxy*: *pxytest* (*http://www.unicom.com/sw/pxytest/*). Ambos se encuentran en nuestro CD.

Los spammers que están interesados en perfeccionar su actividad utilizan, sobre todo, bases de direcciones comerciales *open relay* y *open proxy.* Es muy fácil encontrarlas, basta con escribir en cualquier buscador la palabra *open proxy* u *open relay* y hacer click en los primeros vínculos (p. ej.: *http://www.openproxies.com/* – 20 USD mensuales, *http://www.openrelaycheck.com/* – 199 USD semestrales).

Otro método para obtener direcciones es descargar la zona que contenga direcciones *open relay* u *open proxy* de uno de los servidores DNSBL (véase el artículo *Protección contra spams en el servidor*). De la página web *http://www.declude.com/junkmail/support/ip4r.htm* podéis bajar un listado con todos los servidores de este tipo. Para descargar la zona podemos emplear la aplicación *host*: `host -l <nombre de la zona> <dirección DNSBL>`. Desgraciadamente, muchos servidores DNSBL no permiten bajar zonas completas.

nsabilidad de otra persona, dificultando en gran medida su detección (la IP del spammer estará únicamente en los logs del servidor proxy y el destinatario de la correspondencia podrá recibirla solamente con ayuda del administrador del proxy). Si el spammer está muy interesado en ocultar su IP puede emplear varios *open proxy* en cascada (conectando el primero con el segundo y así sucesivamente hasta llegar al servidor de correo).

Zombie

Zombie es un método nuevo y a la vez el más invasor, empleado por los spammers para traspasar los costes y las responsabilidades a terceros. Esta técnica está basada en la unión de un virus (worm) con un troyano. El objetivo es crear un *open proxy* en el ordenador infectado por el virus. De esta manera se crea una red enorme de *open proxy* anónimos, de los que se aprovechan los spammers de todo el mundo.

El ejemplo más conocido de zombie son los virus de la serie *Sobig*. He aquí como funciona el variante *Sobig.E*:

- El primer elemento, después de infectar el ordenador del usuario (al abrir el adjunto de un email), se envía a todas las direcciones encontradas en los archivos .txt y .html del disco duro.
- Entre las 19:00 y las 23:00, hora UTC, se conecta con una de las 22 direcciones IP que contiene el código del virus) en el puerto 8998 UDP para obtener la dirección URL de la que podrá tomar el segundo elemento.
- Tras haber tomado el segundo elemento (el troyano) se instala y se ejecuta; la dirección IP del ordenador infectado es enviada al autor del zombie. A continuación, se toma el tercer elemento.
- El tercer elemento es el programa *Wingate* modificado, que después de su instalación automática pone en marcha el *open proxy* en el ordenador del usuario.

Podéis encontrar más información sobre los virus de la serie *Sobig* en la URL *http://www.lurhg.com/sobig.html*.

El único método seguro contra los zombies es el uso de programas antivirus y sistemas IDS (*Intrusion Detection System,* por ejemplo, *Snort*) que nos ayudarán a detectar *open proxy* en nuestra red.

Mejor prevenir que curar

Como podemos ver, no es difícil aprovecharse de los servidores mal configurados. Para el administrador de un servidor vulnerable las consecuencias pueden ser muy serias y el spammer saldrá probablemente impune. Por tanto, no hay que menospreciar la problemática relacionada con la seguridad de los servidores ante los spammers.

Así que antes de arrancar nuestro propio servidor proxy, asegurémonos de que solamente los usuarios de la red local están autorizados a utilizarlo. Nuestro servidor de correo debe exigir autorización, a pesar de que los portales polacos más grandes nos dan un mal ejemplo en este campo. Probablemente se vea afectada la comodidad de nuestros usuarios, pero el objetivo de este funcionamiento no tiene que convencer a nadie. ■

La historia de spam

La palabra spam proviene del nombre de jamón con especias enlatado producido por la empresa *Hormel Foods* – SPAM. Las siglas proceden de *Shoulder Pork and hAM* o *SPiced hAM*. ¿Cómo se empezó a asociar jamón con especias con el correo indeseado? Parcialmente la culpa por ello tienen los autores del show *Monty Python's Flying Circus*. En uno de sus sketches se representa un restaurante cuya dueña recomienda SPAM, un componente de cada plato. A las mesas está sentado un grupo de vikingos cantando *spam, spam, spam, lovely spam, wonderful spam* (*spam, spam, spam, rico spam, maravilloso spam*), con lo cual de vez en cuando ensordece a la dueña.

Es difícil determinar quién fue primero en usar la palabra spam para denominar los abusos en la red. Puede ser que los pioneros eran los MUDs (de *Multi-User Dungeon),* los usuarios de juegos RPG de texto quienes solían aplicar este término a envíos que contentían demasiados datos en una entrega (hoy día es más frecuente llamarlo *flooding)*. Es posible que la palabra spam se haya empleado por primera vez en los chats *Bitnet Relay*, que eran los precursores de IRC.

Por el primer caso de spam en el correo se considera una carta enviada en 1978 por *Digital Equipment Corporation*. Dicha empresa mandó un anuncio sobre el nuevo ordenador DEC-20 a todos los usuarios de *Arpanet* de la costa occidental de los E.E.U.U. Sin embargo, la palabra *spam* no se adoptó antes de 1994, cuando en *Usenet* apareció un anuncio del despacho de los abogados Lawrence Cantera y Martha Siegiel. Informaban en él de su servicio de rellenar formularios de la lotería para visado americano. El anuncio fue enviado mediante un script a todos los grupos de discusión de entonces.

Actualmente, con la noción *spam* se determina correo electrónico enviado a propósito, en grandes cantidades, a personas que no desean recibir tales mensajes. Pueden (pero no tienen que) ser ofertas comerciales, propaganda política, etc. Desde hace poco se suele llamar ham (del ing.: jamón) el correo que no es spam.

Más sobre la historia de spam en:
http://www.templetons.com/brad/spamterm.html,
http://www.geocities.com/SiliconValley/Way/4302/spam.html
Sketch del spam (Monty Python's Flying Circus – 1969): *http://www.rompecadenas.com.ar/sketch.htm*