Comprendre l’envoi des spams

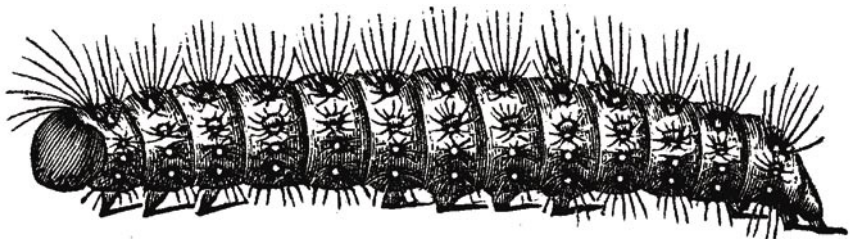
Tomasz Nidecki

Article publié dans le numéro 2/2004 du magazine *Hakin9*

Magazine *Hakin9*, Wydawnictwo Software, ul. Lewartowskiego 6, 00-190 Warszawa, hakin9@hakin9.org

Comprendre l'envoi des spams

Tomasz Nidecki



Les spammeurs profitent souvent des systèmes mal protégés. Les ennuis et les frais liés à l'envoi des dizaines ou des centaines de milliers de messages sont reportés sur des tiers. Nous allons apprendre en quoi consistent les techniques utilisées par les spammeurs et comment s'en protéger contre elles.

Un envoi massif de courrier électronique consomme beaucoup de ressources. Pour le mener à bien, il faut disposer d'une connexion rapide et d'un ordinateur dédié. Même si le spammeur dispose de ces ressources, l'envoi peut prendre plusieurs heures. Quant aux fournisseurs d'accès à Internet, ils ne sont pas ravis s'ils voient leurs bandes passantes utilisées pour l'envoi du spam et le spammeur peut perdre la connexion au réseau avant même de réussir à envoyer une partie importante des messages. S'il est identifié, il peut aussi subir de graves conséquences légales ou financières.

Pour accélérer l'envoi et le rendre plus efficace, les spammeurs utilisent deux méthodes de base. La première consiste à réduire le temps nécessaire à l'envoi du message. Elle est appelée *fire and forget*, c'est-à-dire « envoyer et oublier ». Lors de l'emploi de cette méthode, l'ordinateur qui envoie le spam n'attend pas la réponse des serveurs qu'il a contactés. La deuxième méthode consiste à s'approprier des ressources appartenant à des tiers qui ont mal configuré leur système d'exploitation ou bien ont été victimes de virus. La plupart des frais et souvent aussi la responsabilité pour l'envoi du spam retombe sur ces personnes, et le spammeur reste impuni.

Protocole SMTP

Pour pouvoir comprendre les méthodes utilisées par les spammeurs, il est nécessaire de connaître les principes de fonctionnement de SMTP : le protocole d'envoi de courrier électronique le plus répandu. Comme la plupart des protocoles utilisés dans le réseau, il est basé sur de simples commandes en mode texte.

Étapes de l'envoi du courrier

Le courrier électronique est envoyé en plusieurs étapes (voir Figure 1). Pour mieux les comprendre,

Cet article explique...

- comment les spammeurs envoient le spam (utilisant les ordinateurs des personnes qui n'y sont pour rien),
- comment protéger son serveur contre les spammeurs,
- comment fonctionne le protocole SMTP,
- ce que c'est que *open relay*, *open proxy* et *zombie*.

Ce qu'il faut savoir...

- comment utiliser les outils de base sous Linux.

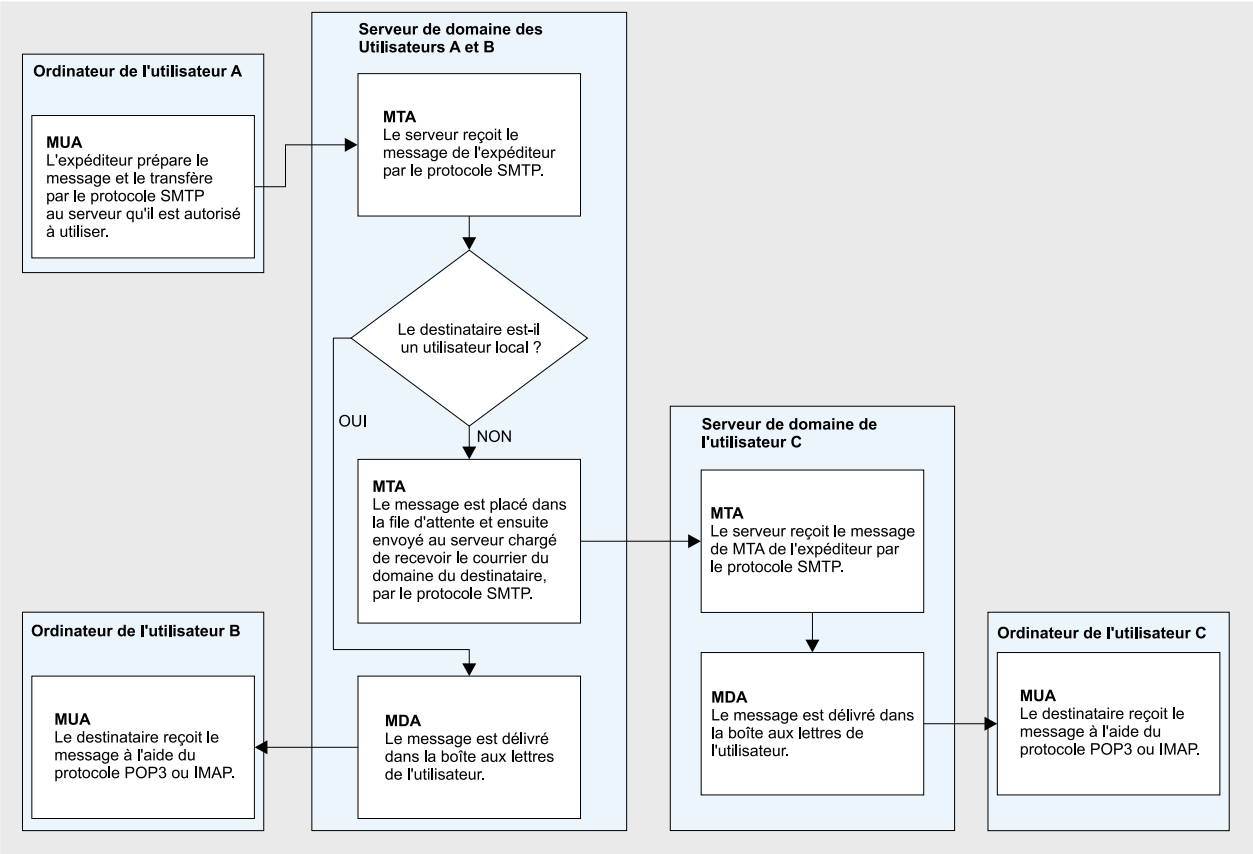


Figure 1. *Étapes de l'envoi du courrier électronique*

imaginons que nous voulons envoyer un message de *hakin9@hakin9.org* à *nobody@example.com*. L'expéditeur emploie le logiciel *Mozilla Thunderbird* en réseau local, et le destinataire utilise *Outlook Express* et il dispose de la connexion dial-up.

Lors de la première étape, le logiciel *Mozilla Thunderbird* entre en contact avec le serveur SMTP indiqué dans les paramètres du compte de l'utilisateur *hakin9@hakin9.org*, c'est-à-dire *mail.software.com.pl*. Le message est envoyé au serveur à l'aide du protocole SMTP. Lors de la seconde étape, *mail.software.com.pl* examine les enregistrements des serveurs DNS. Il apprend que c'est *mail.example.com* qui est responsable de la réception du courrier adressé au domaine *example.com*. On trouve cette information dans l'enregistrement MX (*Mail Exchanger*) publié par le DNS responsable du domaine *example.com* (nous pouvons l'obtenir à l'aide des logiciels *host* ou *dig* : `host -t mx example.com` ou `dig example.com mx`).

Lors de la troisième étape, *mail.software.com.pl* se connecte à *mail.example.com* et lui transmet le message. Pendant la quatrième étape, *mail.example.com* délivre le message reçu dans la boîte aux lettres locale de l'utilisateur *nobody*. La cinquième étape consiste en une connexion dial-up établie par l'utilisateur de la boîte aux lettres *nobody* avec le serveur *mail.example.com*, à

Histoire de SMTP

Le logiciel *SNDMSG* (*Send Message*) est considéré comme le précurseur de SMTP. Il a été utilisé en 1971 par Ray Tomlinson (en liaison avec son propre projet, *CYPNET*) pour créer une application permettant d'envoyer le courrier électronique dans le réseau *ARPANET*. Une année plus tard, les commandes `MAIL` et `MLFL` ont été ajoutées à *FTP* , le logiciel utilisé dans *Arpanet* pour l'envoi des fichiers. *FTP* a été utilisé pour l'envoi du courrier jusqu'en 1980, où le premier standard du protocole pour le courrier électronique a été créé : il s'agit de MTP (*Mail Transfer Protocol*), décrit dans le document RFC 772. MTP a été modifié à plusieurs reprises (RFC 780, 788) et en 1982 Jonathan B. Postel a présenté *Simple Mail Transfer Protocol* dans RFC 821.

Cependant, SMTP sous sa forme de base n'a pas satisfait toutes les attentes. Plusieurs documents décrivant les extensions du protocole ont été créés. Sont considérées comme les plus importantes :

- RFC 1123 – obligations concernant les serveurs Internet (concernant aussi SMTP),
- RFC 1425 – implémentation du standard d'extension du SMTP, à savoir ESMTP,
- RFC 2505 – suggestions concernant la protection anti-spam pour les serveurs,
- RFC 2554 – authentification des connexions : implémentation de la commande `AUTH`.

Le standard courant de SMTP a été décrit en 2001 dans RFC 2821. Sur notre CD-ROM, vous trouverez la totalité des RFC.

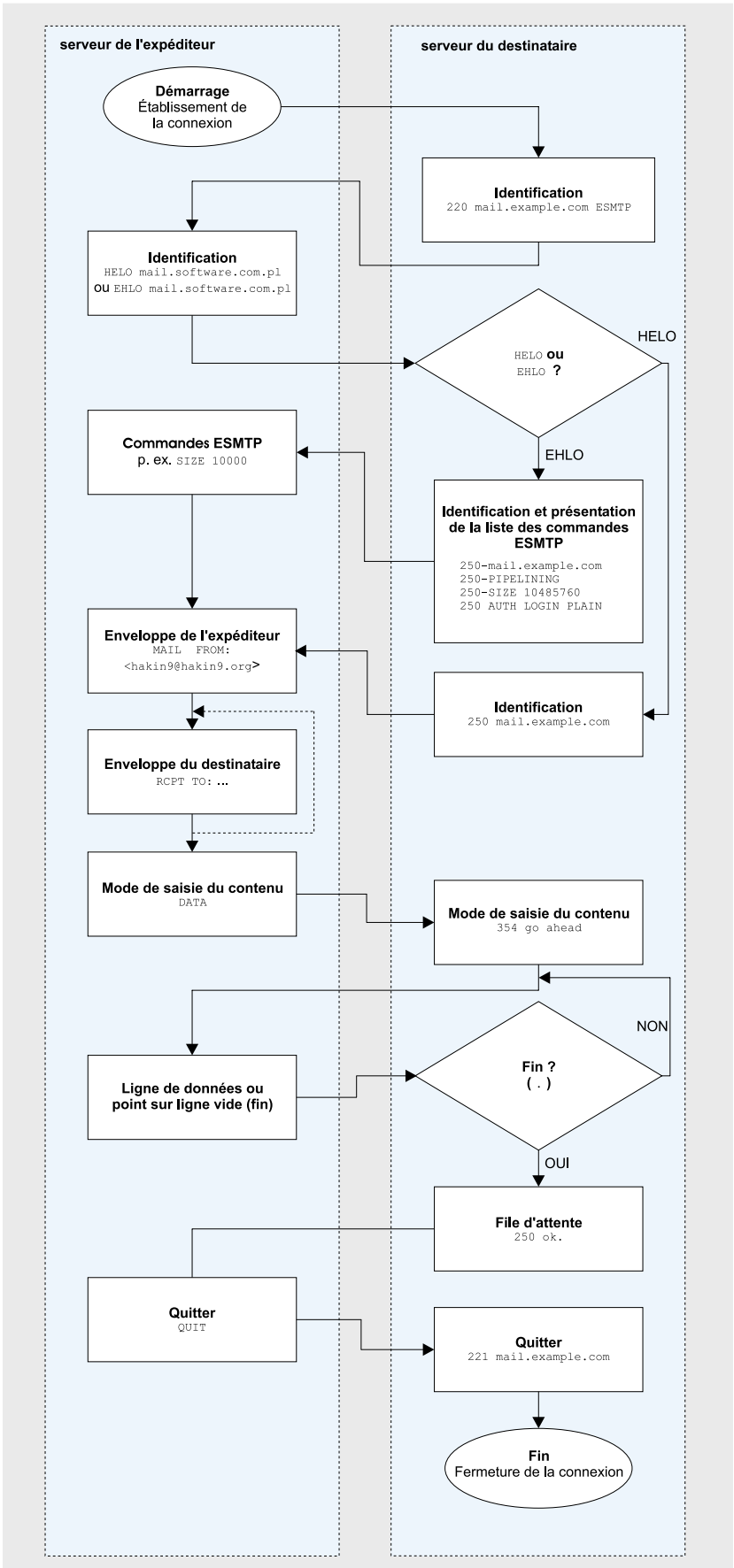


Figure 2. *Étapes de la communication à l'aide de SMTP*

l'aide du protocole POP3 (ou IMAP) et du logiciel *Outlook Express* et en récupération du message.

Il arrive que le chemin du message soit un peu plus long. L'expéditeur pourrait utiliser des serveurs de messagerie (SMTP) séparés, p. ex. *reception.software.com.pl* et *envoi.software.com.pl*. Le message serait alors reçu par *reception.software.com.pl*, transféré vers *envoi.software.com.pl* et ensuite envoyé vers *mail.example.com*. Il en est de même dans le cas de *mail.example.com* des serveurs différents peuvent être responsables de la réception et de l'envoi des messages à l'utilisateur.

Logiciels participant à l'envoi du courrier

Plusieurs logiciels participent à l'envoi du courrier :

- Le logiciel employé par l'utilisateur final pour la réception, l'envoi, la lecture et la rédaction des messages est appelé MUA – *Mail User Agent*. Des exemples de MUA : *Mozilla Thunderbird*, *Outlook Express*, *PINE*, *Mutt*.
- La partie du serveur chargée de la communication avec les utilisateurs (réception du courrier), ainsi que de l'envoi et de la réception du courrier des autres serveurs est appelée MTA – *Mail Transfer Agent*. Les MTA les plus répandus sont : *Sendmail*, *qmail*, *Postfix*, *Exim*.
- La partie du serveur chargée de délivrer le courrier à l'utilisateur local est appelée MDA – *Mail Delivery Agent*. Les exemples des MDA autonomes : *Maildrop*, *Procmail*. La plupart des MTA disposent de leurs propres mécanismes pour délivrer le courrier aux utilisateurs locaux, donc souvent il n'est pas nécessaire d'utiliser des MDA supplémentaires.

Étapes de communication dans le SMTP

L'envoi du message à l'aide de SMTP est divisé en plusieurs étapes. Voici un exemple de session SMTP entre les serveurs *mail.software.com.pl*

Tableau 1. *Présentation des commandes du protocole SMTP le plus souvent utilisées*

Commande	Description
HELO <FQDN>	Se présenter au serveur
EHLO <FQDN>	Se présenter au serveur et demander la liste des commandes ESMTP disponibles
MAIL FROM:<l'adresse >	Spécifier l'enveloppe : l'adresse de l'émetteur pour un retour éventuel du message qui ne peut pas être délivré
RCPT TO:<l'adresse >	Spécifier l'adresse du destinataire du message
DATA	Passage en mode de réception du contenu du message
AUTH <méthode>	Authentification de la connexion (ESMTP). Les méthodes le plus souvent utilisées : LOGIN, PLAIN et CRAM-MD5

Vous trouverez une liste étendue des commandes SMTP et ESMTP à l'adresse *http://fluffy.codeworks.gen.nz/esmtp.html*

et *mail.example.com*. Les données envoyées par *mail.software.com.pl* ont été marquées du caractère >, et celles reçues de *mail.example.com* par le symbole <.

Après l'établissement de la connexion, *mail.example.com* se présente :

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

en informant que le nom complet de l'hôte (FQDN) est *mail.example.com*. Nous apprenons aussi qu'il est possible d'utiliser les commandes d'ESMTP (SMTP étendu : voir l'encadré) et que le MTA employé est *Program*. Le nom du logiciel est facultatif, certains MTA, par exemple *qmail*, ne l'indiquent pas.

Nous nous présentons :

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

comme réponse, nous recevons :

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

ce qui signifie que *mail.example.com* est prêt à recevoir du courrier. Ensuite, nous spécifions l'enveloppe d'expéditeur, c'est-à-dire l'adresse à laquelle un retour éventuel du message non délivré doit être dirigé :

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

Nous indiquons les adresses auxquelles le message doit être envoyé :

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

Ensuite, après la commande `DATA`, nous transmettons les en-têtes et le contenu du message. Les en-têtes sont séparés du contenu par une ligne vide et le message est terminé par un point sur une ligne vide :

```
> DATA
< 354 go ahead
> From: personne@hakin9.org
> To: tous@example.com
> Subject: Rien
>
> C'est un test
> .
< 250 ok 1075929516 qp 5423
```

Après avoir envoyé le message, nous pouvons terminer la connexion :

```
> QUIT
< 221 Bye
```

Le serveur n'est pas toujours en état d'exécuter nos commandes. Si nous recevons un code qui commence par le chiffre 4 (code de série 4xx), cela signifie que le serveur refuse temporairement de recevoir le message. Nous devrions essayer de réenvoyer le message un peu plus tard. Si nous recevons un code qui commence par le chiffre 5, cela signifie que le serveur refuse définitivement de recevoir notre courrier et que les nouvelles tentatives resteront sans effet. La liste des commandes et des codes les plus importants envoyés par le

Tableau 2. *Présentation des codes de réponse SMTP*

Code	Description
220	Service actif : le serveur informe qu'il est disponible et prêt à recevoir les commandes
250	Commande acceptée
354	Début du contenu du message
450	Boîte aux lettres de l'utilisateur temporairement non disponible (p. ex. bloquée par un autre processus)
451	Erreur locale lors du traitement du courrier
452	Manque de ressources système
500	Commande non reconnue
501	Erreur de syntaxe dans la commande ou dans les paramètres
502	Commande non implémentée
550	Boîte aux lettres de l'utilisateur non disponible
552	Manque de ressources de stockage

Vous trouverez la liste complète des codes et les règles de leur création dans RFC 2821 (sur notre CD-ROM).

Listing 1. *Open relay le plus simple*

```
$ 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
>
> C'est un test
> .
< 250 Ok: queued as 17C349B22
> quit
< 221 Bye
```

Listing 2. *Serveur open relay permettant l'envoi du courrier seulement par les utilisateurs existants*

```
$ 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:<hakin9@hakin9.org>
< 250 Ok
> data
< 354 End data with
  <CR><LF>.<CR><LF>
> Subject: test
>
> C'est un test
> .
< 250 Ok: queued as 31B1F2EEA0C
> quit
< 221 Bye
```

Listing 3. *Serveur multistage open relay permettant l'envoi des messages seulement par les utilisateurs existants*

```
$ 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
>
> C'est un test
> .
< 250 2.6.0 Message accepted.
> quit
< 221 2.0.0
  smtp.poczta.onet.pl Out
```

serveur SMTP a été présentée dans les tableaux 1 et 2.

Serveurs relais ouverts (*open relay*)

Lorsque le protocole SMTP a été créé, le problème de spam n'existait pas encore et chaque utilisateur pouvait employer un serveur quelconque pour envoyer son message dans le monde. Maintenant, quand les spammeurs cherchent l'occasion d'utiliser des serveurs pour envoyer des milliers de messages, cette approche n'est plus valable. Les serveurs permettant d'envoyer du courrier sans authentification dans le monde entier sont appelés relais ouverts *(open relay)*.

Chaque serveur qui permet aux utilisateurs non authentifiés d'envoyer des messages sera tôt ou tard utilisé par les spammeurs, ce qui peut avoir de graves conséquences. Premièrement, cela peut réduire les performances du serveur qui va envoyer du spam au lieu de recevoir et d'envoyer les messages des utilisateurs authentifiés. Deuxièmement, le fournisseur d'accès à Internet peut résilier le contrat, vu que le serveur est utilisé pour un procédé illégal et immoral. Troisièmement, l'adresse IP du serveur sera sur des listes noires et de nombreux serveurs n'accepteront pas son courrier (effacer l'adresse IP de plusieurs listes noires est difficile, et parfois même impossible).

Utilisation de relais ouverts

Nous allons voir qu'il est très simple d'utiliser les relais ouverts pour l'envoi du spam. L'un des serveurs polonais mal configurés et utilisés par les spammeurs nous servira d'exemple : *lenox.designs.pl*. Comme on peut voir sur le Listing 1, aucune démarche spéciale n'a été nécessaire pour envoyer le message. Le serveur considère chaque utilisateur qui s'y connecte comme autorisé à envoyer du courrier. C'est le type de serveur relais ouvert le plus dangereux, car le plus facile à utiliser.

Il existe d'autres serveurs relais ouverts, plus difficiles à utiliser par les spammeurs. A titre d'exemple, nous pouvons citer l'un de ces certains serveurs de messagerie mal configurés du portail polonais *O2* : *kogut.o2.pl*. Comme on peut le voir sur le Listing 2, il a suffi de deviner le nom d'un utilisateur pour usurper son identité et envoyer le message. Dans le cas de certains serveurs, il suffit de donner le nom du domaine local et l'utilisateur dont nous usurpons l'identité ne doit même pas exister.

Le Listing 3 présente une situation similaire : il s'agit du serveur de messagerie de l'un des portails polonais les plus importants, à savoir *Onet* (curieusement, *Onet* prétend lutter activement contre le spam...). Il s'agit d'un *multistage open relay*. Cela signifie que le message est reçu par une IP et envoyé par une autre.

Pour s'en rendre compte, il faut examiner les en-têtes `Received` (voir l'encadré) du message reçu. Comme on peut voir sur le Listing 4, le message a été reçu par *ps8.test.onet.pl* (213.180.130.54) et envoyé au destinataire par *smtp8.poczta.onet.pl* (213.180.130.48). Cela rend plus dif-

Listing 4. *En-têtes Received du courrier reçu du serveur 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
```

Comment éviter de devenir un relais ouvert (*open relay*) ?

Le protocole SMTP permet de :

- recevoir le courrier d'un utilisateur (MUA) et de l'envoyer à un autre serveur (MTA),
- recevoir le courrier d'un autre serveur (MTA) et de l'envoyer à un utilisateur local (MUA),
- recevoir le courrier d'un serveur (MTA) et de l'envoyer à un autre serveur (MTA).

Il n'y a aucune différence entre la manière d'envoyer le message par MUA et par MTA. Le plus important est de savoir si on peut faire confiance à l'adresse IP de l'expéditeur (p. ex. dans le réseau local) et si le destinataire se trouve dans le domaine local ou extérieur.

L'envoi du courrier en dehors de notre serveur est appelé réacheminement (*relaying*). Pour empêcher les spammeurs d'utiliser notre serveur pour leurs activités, nous ne devons pas autoriser le réacheminement sans authentification. Pour cela, lors de la configuration du serveur SMTP, il faut prendre en compte les hypothèses suivantes :

- Si le message est destiné à l'un des domaines gérés par notre serveur, il doit être accepté sans authentification.
- Si le message est envoyé par un utilisateur local (de MUA sur le serveur) dans le réseau local ou bien d'une adresse IP fixe et authentifiée, et que le destinataire soit un utilisateur non local, le message peut être accepté sans authentification (cependant, il est conseillé d'exiger l'authentification).
- Si le message est envoyé par un utilisateur non local (p. ex. d'une adresse IP dynamique) et que le destinataire soit un utilisateur non local, le message ne peut pas être accepté sans authentification.

ficile de découvrir que le serveur est configuré comme un relais ouvert, mais cela n'empêche pas de l'utiliser pour l'envoi du spam.

Les serveurs dont l'authentification de l'expéditeur (SMTP-AUTH) a été incorrectement configurée constituent un autre type de relais ouvert. Ils permettent d'envoyer le courrier après avoir indiqué un login et un mot de passe quelconques. Cela arrive le plus souvent lorsque des administrateurs débutants de *qmail* qui n'ont pas lu la documentation du patch SMTP-AUTH appellent *qmail-smtpd* d'une manière incorrecte.

Le logiciel *qmail-smtpd* avec le patch exige trois arguments : FQDN, le logiciel vérifiant le mot de passe (compatible avec *checkpassword*) et un paramètre additionnel pour le logiciel vérifiant le mot de passe. Exemple : `qmail-smtpd hakin9.org /bin/checkpassword /bin/true`. L'erreur commise le plus souvent consiste à indiquer `/bin/true` comme le second paramètre : la vérification du mot de passe sera toujours réussie, indépendamment du login et du mot de passe saisis. Le spammeur peut aussi tenter une attaque par dictionnaire : il serait donc préférable que les mots de passe des utilisateurs pour l'authentification SMTP ne soient pas trop banals.

En-têtes Received

Les en-têtes `Received` sont un élément obligatoire de chaque message. Chaque serveur de messagerie par lequel passe le courrier ajoute son propre en-tête `Received` au-dessus des autres, affectés par d'autres serveurs. Comme ça, en lisant les en-têtes du bas vers le haut, nous pouvons connaître le chemin du courrier entre l'expéditeur et le destinataire. Le spammeur peut ajouter ses propres en-têtes pour dissimuler son identité et faire attribuer la faute à une autre personne. Les en-têtes situés dans la partie supérieure, insérés par le serveur du destinataire, sont certainement vrais. Les autres peuvent être faux.

Uniquement grâce aux en-têtes `Received`, nous pouvons identifier l'adresse IP réelle de l'expéditeur du message et souvent vérifier si le message a été posté par l'intermédiaire de relais ouvert *(open relay)* ou de proxy ouvert *(open proxy)*. Cependant, l'analyse des en-têtes n'est pas facile, car il n'existe pas de standard fixé pour leur définition et l'ordre des informations dans l'en-tête varie en fonction du serveur de messagerie.

Serveurs *open proxy*

Les *open proxy* sont un autre type de serveurs mal configurés qui peuvent être utilisés par les spammeurs. Il s'agit de serveurs proxy permettant l'établissement de la connexion par les utilisateurs non autorisés. Les serveurs *open proxy* peuvent fonctionner avec divers logiciels et divers protocoles. Le plus souvent, il s'agit du protocole HTTP-CONNECT, mais on trouve aussi des *open proxy* permettant la connexion à l'aide des protocoles HTTP-POST, SOCKS4, SOCKS5 et d'autres.

Open proxy peut être utilisé par un spammeur de la même manière qu'un relais ouvert : pour envoyer des messages non autorisés. De plus, plusieurs d'entre eux permettent de dissimuler l'adresse IP. Un tel proxy sera une aubaine alléchante pour chaque spammeur.

Utilisation du serveur *open proxy*

Le Listing 6 présente un exemple d'utilisation du serveur open proxy permettant la connexion par le protocole HTTP-CONNECT sur le port 80. La communication avec le relais ouvert constitue la plus grande partie de la connexion (les mêmes commandes que dans le Listing 2). Mais avant de nous connecter au serveur SMTP, nous établissons le contact avec *open proxy* et c'est avec son aide que nous nous connectons au MTA. Pendant la connexion, nous déclarons que la communication sera réalisée à l'aide du protocole HTTP/1.0, mais nous ne sommes pas obligés de l'utiliser.

Pour un spammeur, la situation idéale serait de trouver un serveur *open proxy* sur lequel est aussi installé un serveur de messagerie local. Dans la plupart des cas, MTA accepte les connexions du proxy local sans authentification, en des considérant comme les utilisateurs locaux. Dans

Listing 5. *Serveur open relay avec SMTP-AUTH configuré de manière incorrecte*

```
$ 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
> test
< 334 UGFzc3dvcmQ
> test
< 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
>
> C'est un test
> .
< 250 ok 1077563277 qp 13947
> quit
< 221 mail.example.com
```

ce cas-là, le spammeur n'est obligé de connaître aucun serveur relais ouvert et peut confortablement et de manière anonyme réaliser ses activités aux dépens et à la responsabilité d'un autre, tout en restant difficile à découvrir.

Zombie

La méthode appelée « zombie » est la plus récente et la plus agressive utilisée par les spammeurs pour faire retomber les frais et la responsabilité sur des tiers. Cette technique est un mélange de virus (ver) et de cheval de Troie. Son but est de créer un *open proxy* sur l'ordinateur infecté par le virus. De cette manière, on crée un réseau très étendu de serveurs d'*open proxy* anonymes utilisés par les spammeurs du monde entier.

Les virus de la série *Sobig* sont l'exemple le plus connu de zombie. Voici comment fonctionne la variante *Sobig.E* :

- Le premier élément, après avoir infecté l'ordinateur de l'utilisateur (après l'ouverture de la pièce jointe par celui-ci), est envoyé à toutes les adresses trouvées dans les fichiers .txt et .html sur le disque dur.
- Entre 19h et 23h UTC, il se connecte à l'une des 22 adresses IP contenues dans le code du virus, sur le port 8998 UDP, pour obtenir l'adresse URL d'où il peut télécharger son deuxième élément.
- Après le téléchargement du deuxième élément (cheval de Troie), celui-ci est installé et exécuté ; l'adresse IP de l'ordinateur infecté est envoyée à l'auteur du zombie et ensuite le troisième élément est téléchargé.
- Le troisième élément c'est le logiciel *Wingate* modifié qui s'installe automatiquement et crée un *open proxy* sur l'ordinateur de l'utilisateur.

Vous trouverez plus d'informations sur le fonctionnement des virus de la série *Sobig* à l'adresse *http://www.lurhq.com/sobig.html*

La seule méthode efficace de protection contre les zombies consiste à utiliser les logiciels anti-virus et les systèmes IDS (*Intrusion Detection System*, par exemple *Snort*) qui nous aideront à détecter un open proxy dans notre réseau.

Listing 6. *Serveur open proxy utilisé pour un envoi anonyme du courrier par open relay*

```
$ telnet 204.170.42.31 80
> CONNECT kogut.o2.pl:25 HTTP/1.0
>
< HTTP/1.0 200S
  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 withS
  <CR><LF>.<CR><LF>
> Subject: test
>
> C'est un test
> .
< 250 Ok: queued as 5F4D41A3507
> quit
< 221 Bye
```

Mieux vaut prévenir que guérir

Comme nous l'avons vu, utiliser les serveurs mal protégés n'est pas difficile. Pour l'administrateur du serveur « troué », les conséquences peuvent s'avérer graves, tandis que le spammeur n'assumera vraisemblablement aucune responsabilité. ■

Où les spammeurs trouvent-ils les adresses de relais ouverts et d'*open proxy* ?

Il peut être assez difficile de trouver soi-même un serveur mal protégé. Il suffit cependant de recevoir du spam envoyé par relais ouvert ou par *open proxy* pour pouvoir les utiliser soi-même. Pour vérifier si un relais ouvert se trouve sous l'adresse IP suspecte, on peut utiliser le script *rlytest* (*http://www.unicom.com/sw/rlytest/*), tandis que pour découvrir un *open proxy* on peut se servir de *pxytest* (*http://www.unicom.com/sw/pxytest/*). Vous trouverez les deux sur notre CD-ROM.

Les spammeurs qui tiennent à obtenir de meilleures performances dans leur activité, utilisent surtout les bases de données payantes d'adresses de relais ouverts et d'*open proxy*. Il n'est pas difficile d'en trouver : il suffit de saisir les mots *open proxy* ou *open relay* dans n'importe quel moteur de recherche et cliquer sur les premiers liens qui apparaissent (p. ex. : *http://www.openproxies.com/* – 20 USD pour un mois, *http://www.openrelaycheck.com/* – 199 USD pour six mois).

Une autre méthode pour obtenir des adresses consiste à récupérer la zone contenant les adresses de relais ouverts ou d'*open proxy* à partir de l'un des serveurs DNSBL (voir l'article *Protection contre le spam sur le serveur*). Une liste de ces serveurs peut être trouvée, par exemple, à l'adresse *http://www.declude.com/junkmail/support/ip4r.htm*. Pour récupérer la zone mentionnée, on peut utiliser l'application *host* : `host -l <nom de la zone> <adresse DNSBL>`. Toutefois, de nombreux serveurs DNSBL rendent impossible la récupération des zones entières.