Mini-HOWTO Term-Firewall

Barak Pearlmutter, `barak.pearlmutter@alumni.cs.cmu.edu` 22 Mai 1996

(Version française réalisée par Eric Dumas, `dumas@freenix.fr`, `dumas@Linux.EU.Org`, 1er Juillet 1997). Ce document décrit comme utiliser Term pour traverser un Firewall Internet, ce que vous n'êtes pas supposé pouvoir faire.

Contents

1 Avertissements (Important !)

Je décline sur le présent document toute responsabilité d'une quelconque application de ce qui va suivre. Si cela échoue de n'importe quelle manière, c'est votre problème. Ce n'est pas ma faute. Si vous ne comprenez pas les risques qui découlent de cette méthode, ne l'utilisez pas. Si vous employez cette méthode et que cela permet à des pirates vicieux de pénétrer dans votre système informatique et que cela vous coûte votre travail et à votre entreprise des millions de dollars, ce n'est que de votre faute. Ne venez pas pleurer.

2 Copyright

3 Introduction

Le programme `term` est normalement utilisé sur une ligne série ou modem, pour permettre à plusieurs services machine-à-machine de communiquer grâce à cette simple connexion série. Toutefois, il est assez utile quelquefois d'établir une connexion entre deux machines communiquant par `telnet` avec `term`.

L'utilisation la plus intéressante réside dans la connexion de deux machines séparées par des *firewalls* ou par des serveurs *socks*. Les *firewalls* permettent l'établissement de connexions à travers eux-mêmes, typiquement en utilisant le protocole *socks*. Ce dernier permet aux machines du réseau interne de se connecter à l'extérieur, et oblige les utilisateurs extérieurs à se connecter en premier sur la machine passerelle qui leur demande un mot de passe. Ces *firewalls* rendent impossible, par exemple, la communication entre un client X sur une machine intérieure et un serveur extérieur. Mais, en configurant une connexion `term`, ces restrictions peuvent être contournées assez facilement, au niveau de l'utilisateur.

4 Mise en oeuvre générale

Configurer une connexion `term` par-dessus une session `telnet` se fait en deux phases.

Dans un premier temps, votre client habituel `telnet` est utilisé pour configurer une connexion `telnet` et pour se connecter. Ensuite, le client `telnet` est mis en sommeil, et fait en sorte que la connexion `telnet` soit transmise à `term`.

5 Procédure détaillée

En détail, la marche à suivre est la suivante :

1. A partir d'une machine à l'intérieur du *firewall*, se connecter par `telnet` à l'extérieur de celui-ci et s'y loger.

2. Sauf si vous êtes sous **Linux** et que vous utilisez le système de fichiers */proc* (voir ci-dessous), vérifiez que votre shell est du genre `sh`. C.à.d. que votre shell par défaut soit une variante de `csh`. Appelez `telnet` par `(setenv SHELL /bin/sh; telnet machine.la-bas.dehors)`.

3. Après s'être logé, lancer la commande sur la machine de l'extérieur : `term -r -n off telnet`.

Maintenant revenez à l'invite `telnet` sur la machine locale, en utilisant le caractère d'échappement `^]` (ou celui que vous voulez), puis utilisez la commande de `telnet` pour exécuter une commande shell `!` pour lancer `term` :

```
telnet> ! term -n on telnet <&3 >&3
```

Et voilà !!! (Ndt : En français dans le texte).

(Si vous possédez une autre version de `telnet`, vous risquez d'avoir à utiliser d'autres descripteurs de fichiers que 3. C'est facile à déterminer en utilisant trace. Mais 3 semble fonctionner sur tous les `telnet` de type bsd que j'ai testés. J'ai également essayé sous Sun OS 4.x et les distributions **Linux** standard.)

Certains clients telnet ne possèdent pas de caractère d'échapement !. Par exemple, client telnet diffusé avec la Slackware 3.0 en fait partie. Les sources de ce client sont sensés provenir de ftp://ftp.cdrom.com:/pub/linux/slackware-3.0/source/n/tcpip/NetKit-B-0.05.tar.gz, paquetage qui contient le caractère d'échapement. Une solution assez simple est de récupérer ces sources et de les recompiler. Je n'y suis malheureusement pas arriver. De plus, si vous êtes à l'intérieur d'un firewall socks, vous devrez avoir un client telnet à la SOCKS. J'ai réussi à compiler un tel cient en utilisant ftp://ftp.nec.com/pub/security/socks.cstc/socks.cstc.4.2.tar.gz ou si vous êtes à l'extérieur des USA, ftp://ftp.nec.com/pub/security/socks.cstc/export.socks.cstc.4.2.tar.gz

Autrement, sous **Linux** version 1.2.13 ou précédentes, vous pouvez mettre `telnet` en sommeil avec `^]^z` , récupérer son pid et lancer :

```
term -n on -v /proc/ < telnetpid > /fd/3 telnet
```

Cela ne fonctionne plus avec les noyaux 1.3 et supérieur, qui ont vérouillé certaines failles de sécurité pour éviter les accès à des descripteurs de fichiers n'appartenant pas au propriétaire du processus ou à ses fils.

6 Sockets pour `term` multiples

C'est une bonne idée de donner un nom explicite à la socket de `term`. C'est l'argument donné à `telnet` dans la ligne de commande ci-dessus. A moins que vous n'ayez la variable d'environnement TERMSERVER positionnée à `telnet`, vous pouvez appeler les clients avec le paramètre `-t`, c'est-à-dire : `trsh -t telnet`.

7 Le fichier d'initialisation .term/termrc.telnet

J'ai attendu que la ligne soit claire en utilisant un vérficateur de ligne sur ce média. J'espérais qu'il soit totalement transparent, mais cela semble impossible. Toutefois, le seul caractère perturbant semble être le 255. Le fichier `/.term/termrc.telnet` que j'emploie (le fichier `.telnet` est le nom de la connexion `term`, cf. supra) contient :

```
baudrate off
escape 255
ignore 255
timeout 600
```

Il peut être amélioré en trichant, j'ai un débit de seulement 30.000 cps (caractères par secondes) pour une connexion longue distance à-travers un *firewall* lent. FTP peut aller jusqu'à 100.000 cps en suivant le même chemin. Une vitesse en bps (bits par seconde) réaliste peut éviter quelques temps morts.

8 Administration

Manifestement, si vous attaquez de l'extérieur du *firewall*, et que vous employez une carte avec un identificateur sécurisé ou quelque chose de ce genre, vous voudrez sûrement inverser les rôles des serveurs de connexion et local (si vous ne comprennez pas ce que cela signifie, vous n'êtes peut-être pas assez familier avec `term` pour utiliser l'astuce décrite dans ce document d'une manière responsable).

9 Securité

Ce n'est rien moins qu'une faille que la possibilité d'avoir une connexion `telnet` détournée sur une machine non sécurisée de l'extérieur. Le premier risque supplémentaire provient des personnes capables d'utiliser la socket `term` que vous avez configurée sans que vous soyez au courant. Donc, soyez prudents (personnellement, je fais cela sur une machine externe que je sais être sécurisée. Pour être plus précis, un portable sous **Linux** que j'administre moi-même et qui n'accepte aucune connexion de l'extérieur).

Une autre possibilité est d'ajouter `socket off` dans `~/.term/termrc.telnet` ou ajouter `-u off`. Cela évide que la socket soit accessible du site distant, avec une perte de fonctionnalité assez mineure.

10 Mode `telnet`

Vérifiez que le démon `telnetd` distant n'est pas dans un mauvais mode sept bits. Si c'est le cas, vous devez l'indiquer à `term` lorsque vous le lancez en ajoutant un `-a` sur la ligne de commande (j'emploie de temps en temps un `^] telnet> set outbin` ou un `set bin` ou bien, je lance `telnet` avec l'option *-8* pour forcer la connexion en mode 8 bits).

11 Bugs et mes souhaits concerant term

Le programme de vérification de ligne a de temps en temps quelques problèmes pour contrôler la connexion `telnet`. Cela provient parfois du fait qu'il ne vérifie pas le code de retour de l'appel `read()`. Pour des connexions réseau, cet appel peut retourner le code d'erreur -1 avec *EINTR* (interrompu) ou *EAGAIN* (reéssayer). Manifestement, cela serait une bonne chose que cela soit vérifié.

Un certain nombre de caractéristiques pourraient faciliter l'utilisation de `term` sur `telnet`. Cela provient essentiellement d'une hypothèse qui a influencé le développement de `term`, qui est que la connexion dispose d'une largeur de bande faible, d'une latence réduite et qu'elle est quelque peu bruitée.

Une connexion `telnet` possède en général une bande passante assez importante, une grande latence et qui contient peu d'erreurs. Cela signifie que la connexion pourrait être mieux utilisée si :

1. la taille maximale de la fenêtre était augmentée, bien au-delà de la limite imposée par la formule $N\_PACKETS/2 = 16$ de `term`
2. une option pour désactiver l'envoi et la vérification du *checksum* des paquets était implémentée
3. de plus grands paquets étaient permis lorsque cela est approprié.

Egalement, pour améliorer la sécurité, il serait sympathique d'avoir une option dans `term` pour afficher la liste des connexions réalisées par la socket dans un fichier ou sur stderr, ou bien dans les deux. Cela permettrait de vérifier si une connexion `term` a été corrompue par des pirates situés du côté non sécurisé de la machine.

12 Trucs qui semblent ne pas fonctionner

Quelques clients et serveurs `telnet` acceptent d'encoder leurs communications pour tromper la surveillance sur réseau. Malheureusement, la méthode employée ci-dessus (en utilisant la connexion réseau que le client `telnet` a configuré pendant que l'autre client est en attente) ne fonctionne pas dans ce cas. Au lieu de cela, il doit réellement traverser le client `telnet`, donc ne peut réaliser l'encodage. Il semble qu'il faille modifier que le client, pour y a jouter une commande qui lance un processus avec leurs stdin et stdout connectés au `telnet` en cours. Cela serait également utile pour des processus de connexion automatiques, peut-être quelqu'un l'a-t-il déjà fait.

13 Sources

J'ai légèrement consulté la bibliothèque Term. Les détails ainsi que les patches à SOCKS sont disponibles en les demandant à Steven Danz (danz@wv.mentorg.com).

14 Remerciement

Mes remerciements à

- Gary Flake (flake@scr.siemens.com)
- Bill Riemers (bcr@physics.purdue.edu)
- Greg Louis (glouis@dynamicro.on.ca)

15 Copie supplémentaire des avertissements -Lisez-le !

Je décline sur le présent document toute responsabilité d'une quelconque application de ce qui a été exposé. Si cela échoue de n'importe quelle manière, c'est votre problème. Ce n'est pas ma faute. Si vous ne comprenez pas les risques qui découlent de cette méthode, ne l'utilisez pas. Si l'emploi de cette méthode permet à des pirates vicieux de pénétrer dans votre système informatique et que cela vous côte votre travail et à votre entreprise des millions de dollars, ce n'est que de votre faute. Ne venez pas pleurer.