Petit guide du perçage de pare-feux

Version française du petit guide Firewall Piercing mini-HOWTO

François-René Rideau

<fare PLUS fwprc CHEZ tunes POINT org>

Arnaud Muller

Adaptation française<arnaud POINT muller CHEZ insa TIRET rouen POINT fr>

Yvon Benoist

Relecture de la version française<yvon POINT benoist CHEZ insa TIRET rouen POINT fr>

Jean-Philippe Guérard

Préparation de la publication de la v.f.<fevrier CHEZ tigreraye POINT org>

Version : 0.97

18 septembre 2006

Historique des versions		
Version 0.97.fr.1.0	2006-09-18	AM, YB, JPG
Première version française.		
Version 0.97	2001-11-24	FRR
Conversion au format SGML DocBook. *Conversion to DocBook SGML.*		

Résumé

Comment utiliser **ppp** par-dessus **ssh**, **telnet** ou autre, de façon à réaliser une connexion réseau transparente à travers un pare-feu. Valable aussi bien pour l'élaboration d'un VPN convivial que pour la perforation des pare-feu gênants.

Table des matières

Divers

Avis de non-responsabilité

Lisez cette section, elle est importante !!!!

Par la présente je décline toute responsabilité quant à l'utilisation que vous ferez de cette méthode d'effraction [hack]. Si ça se retourne contre vous d'une manière ou d'une autre, c'est pas de pot. Et c'est pas ma faute. Si vous ne comprenez pas les risques encourus, laissez tomber. Si vous utilisez cette méthode de piratage et qu'il permet à des vandales sans scrupules de pénétrer dans les ordinateurs de votre société et que ça vous coûte votre boulot et des millions d'euros à votre entreprise, eh bien c'est pas de pot. Ne venez pas pleurer chez moi.

Baratin juridique

Pour faciliter leur travail, les droits ont également été cédés aux responsables de la maintenance du LDP [Linux Development Project] sous la Licence de Documentation Libre GNU.

Recherche responsable de la maintenance

Je ne m'occupe plus activement de l'évolution de ce petit guide, bien que je continue d'en assurer la maintenance. Je recherche une personne qui pourrait prendre le relais pour la maintenance, qui l'étofferait pour en faire un petit guide à part entière en s'étendant davantage sur les solutions dont je ne fais que mentionner l'existence, et qui pourrait peut être développer des logiciels pour rendre la perforation de pare-feu plus facile. J'ai énormément d'idées pour étendre le contenu de ce petit guide et développer un logiciel adéquat, si quelqu'un est intéressé. J'avais également écrit une version française de ce guide pratique, mais personne n'en assure plus la maintenance depuis longtemps [NdT : ceci est une nouvelle traduction].

Remerciements

Même si tout ce qu'il en reste est le paragraphe sur la non responsabilité, ce document doit énormément au Term-Firewall mini-HOWTO de Barak Pearlmutter <bap CHEZ cs POINT unm POINT edu>. Le petit guide de Barak repose sur TermTM, ancien programme qui n'est plus supporté (excellent programme en son temps, et qui peut être encore utile dans certaines circonstances malheureuses), ainsi que certaines particularités d'une implémentation non standard de telnet, autrement dit beaucoup d'éléments obsolètes et non portables. Néanmoins, un petit guide sur la perforation des pare-feu était nécessaire, et malgré les limites des méthodes d'effraction [hacks] qu'il propose, son petit guide a été un modèle et un encouragement.

J'aimerais également féliciter Lars Brinkhoff <lars CHEZ nocrew POINT org> et Magnus Lundström <logic CHEZ gore POINT nocrew POINT org> pour leurs très bons tunnels http, messagerie et icmp.

Dernières versions

La dernière version LDP officielle de ce document est sur :
http://www.linuxdoc.org/HOWTO/mini/Firewall-Piercing.html

La source de ma dernière version officielle de ce document est sur : http://fare.tunes.org/files/fwprc/

La source de mon dernier brouillon de travail pour ce document est sur :
http://tunes.org/cgi-bin/cvsweb/fare/fare/www/articles/Firewall-Piercing.en.sgml

Introduction

Avant propos

Ce document a une morale. Et cette morale est : *un pare-feu ne peut pas protéger un réseau contre ses propres utilisateurs internes et ne devrait même pas essayer.*

Lorsqu'un utilisateur interne vous demande à vous, administrateur système, d'ouvrir un port sortant vers une machine externe, ou un port entrant vers une machine interne, vous devez le faire pour lui. Evidemment, vous devez aider l'utilisateur pour être sûr que ses transactions sont sécurisées, et que son logiciel est robuste. Mais refuser carrément de rendre ce service à l'utilisateur relève d'une

incompétence évidente. A moins que le pare-feu ne le coupe complètement du reste du monde extérieur, au point de se retrouver sans **ssh**, **telnet**, navigateur web, courrier électronique, dns, ligne téléphonique, radio, et sans pouvoir faire de **ping**, rien de rien, il n'en demeure pas moins que l'utilisateur peut utiliser les techniques de perforation de pare-feu pour accéder aux machines qu'il veut , et il le fera, et, résultat final au niveau sécurité, on aura une connexion non vérifiée avec le monde extérieur. Alors soit vous faites confiance à vos utilisateurs, après une formation et une sélection appropriées, soit vous leur refusez tout accès au réseau; mais là encore le rôle d'un administrateur réseau est habituellement de servir ses utilisateurs, alors c'est plutôt le premier cas de figure qu'il faut viser, pas le deuxième. Vous pouvez et vous devez les protéger contre le monde extérieur, vous pouvez et vous devez protéger vos services sensibles contre vos utilisateurs, mais vous ne les protégerez point contre eux-mêmes, et vous ne le pouvez pas.

Parce que des administrateurs passifs, ou absents, ou surchargés, ou carrément incompétents, ou irresponsables, ou simplement dirigés par des personnes incompétentes, ça existe, il se trouve parfois qu'un utilisateur puisse se retrouver derrière un pare-feu qu'il peut traverser, mais ce ne sera pas très commode. Ce petit guide fournit un moyen générique et portable de percer des tunnels dans des pare-feu, en transformant les rares petits bits qui arrivent au compte-gouttes en une véritable autoroute de l'information, de sorte que l'utilisateur puisse utiliser d'une façon cohérente des outils standards pour accéder aux ordinateurs de l'autre côté du pare-feu. La même technique peut être utilisée par les administrateurs réseaux compétents pour faire des réseaux privés virtuels (VPN [Virtual Private Networks]).

Les problèmes de sécurité.

Bien entendu si votre administrateur système a installé un pare-feu, il a sûrement une bonne raison et vous avez sûrement signé un engagement à ne pas le contourner. D'un autre côté, le fait que vous puissiez utiliser telnet, le web, la messagerie électronique, ou tout autre flux quel qu'il soit d'information bidirectionnel avec l'extérieur du pare-feu (ce qui est une condition sine qua non pour que les méthodes présentées puissent fonctionner) signifie que vous êtes autorisé à accéder à des systèmes externes, et le fait que vous puissiez ouvrir une session sur un système externe particulier signifie que vous êtes aussi autorisé à le faire.

Il s'agit donc ici tout simplement d'utiliser de façon *commode* les failles légales d'un pare-feu, et d'autoriser les programmes génériques à fonctionner à partir de là avec les protocoles génériques, plutôt que d'avoir recours à des programmes spéciaux ou modifiés (et recompilés) passant par de nombreux proxies à usage particulier qui ont été mal configurés par un administrateur système négligent ou incompétent, ou d'installer de nombreux convertisseurs à usage particulier pour accéder à chacun de vos services habituels (comme la messagerie électronique) en passant par des chemins supportés par le pare-feu (comme le web).

De plus, l'utilisation d'un émulateur d'IP niveau utilisateur tel que SLiRPTM devrait permettre de maintenir la protection contre les attaques extérieures par perforation du pare-feu, à moins que vous ne le permettiez explicitement (ou alors les agresseurs sont astucieux et malicieux, et root, ou, sinon, capables de vous espionner sur le serveur).

L'un dans l'autre, la méthode présentée devrait être relativement sécurisée. Cependant, ça dépend des conditions particulières dans lesquelles vous faites l'installation, et je ne peux donner aucune garantie sur cette méthode. De nombreuses choses sont intrinséquement non sécurisées au niveau des connections internet, que ce soit avec cette méthode ou pas, donc n'imaginez surtout pas que telle ou telle chose est sécurisée à moins que vous n'ayez de bonnes raisons, et/ou utilisez un procédé de chiffrage tout le temps.

Répétons les bases de la sécurité réseau : *vous ne pouvez faire confiance à rien du tout au niveau d'une connexion, pas plus que vous ne faites confiance aux hôtes qui peuvent manier les données non cryptées*, y compris les hôtes des deux côtés de la connection, et tous les hôtes qui peuvent intercepter la communication, à moins que la communication soit cryptée correctement avec des clefs secrètes. Si vous placez mal votre confiance, vos mots de passe peuvent être volés et utilisés contre vous, votre numéro de carte de crédit peut être volé et utilisé contre vous, et vous pouvez être viré de votre boulot pour avoir mis en danger toute l'entreprise. Tant pis pour vous.

Pour résumer, n'utilisez pas cette méthode sauf si vous savez ce que vous faites. Relisez l'avis de non-responsabilité plus haut.

Autres conditions requises

On suppose que vous savez ce que vous faites, que vous savez comment configurer une connexion au réseau, qu'en cas de doute vous aurez lu toute la documentation qu'il faut (guides pratiques, manuels, pages web, archives de listes de diffusion, RFC, cours, tutoriels).

On suppose que vous avez des comptes console des deux côtés du pare-feu, que vous pouvez d'une façon ou d'une autre transmettre des paquets d'information dans les deux sens à travers le pare-feu (avec **telnet**, **ssh**, le courriel, et le web comme moyens les plus couramment utilisés pour travailler), et que vous pouvez laisser un démon en fonctionnement comme tâche en arrière-plan sur le serveur (ou bénéficier d'un démon existant, **sshd**, **telnetd**, ou **sendmail**/**procmail**).

On suppose que vous savez ou que vous êtes disposé à apprendre comment configurer un émulateur d'IP (**pppd**, **slirp**) ou un accès à internet démon et la bibliothèque qui va avec (SOCKSTM, TermTM) des deux côtés, en fonction de vos besoins en terme de connectivité et de vos droits d'accès, en recompilant certains logiciels si besoin est.

Enfin, et c'est également important, pour que vous puissiez utiliser les méthodes décrites dans ce document, on suppose que vous êtes root du côté du pare-feu qui a besoin d'un accès IP transparent complet vers l'autre côté. En effet, il vous faudra lancer le démon PPP de ce côté, ce qui permet l'utilisation de l'installation normale de routage du paquet kernel. Au cas où vous n'êtes pas root de ce côté, votre cas n'est pas désespéré malgré tout : en effet le Term-Firewall mini-HOWTO de Barak Pearlmutter décrit comment utiliser TermTM, programme entièrement fait pour l'utilisateur, en vue du perçage de pare-feu. Bien qu'il n'y ait aucun petit guide, je soupçonne SOCKSTM de pouvoir également être utilisé comme un moyen de percer les pare-feu sans avoir le privilège root; j'accepterai volontiers vos contributions à ce Guide pratique sur cette méthode pour percer les pare-feu.

Logiciels à télécharger

La plupart des logiciels mentionnés dans ce Guide pratique devrait être disponible dans votre distribution de Linux standard, éventuellement parmi les contributions. Au moins, les quatre premiers ci-dessous sont disponibles en tant que paquets `.rpm` et `.deb` Au cas vous voudriez récupérer les dernières versions (après tout, un des deux bouts de la connexion peut ne pas fonctionner sous Linux), utilisez les adresses ci-dessous :

- **SLiRP** se trouve sur http://blitzen.canberra.edu.au/slirp ou sur ftp://www.ibc.wustl.edu/pub/slirp_bin/.

- **zsh** se trouve sur http://www.zsh.org/.
- **ppp** se trouve sur ftp://cs.anu.edu.au/pub/software/ppp/.
- **ssh** se trouve sur http://www.openssh.com/.
- **fwprc**, **cotty** et **getroute.pl** se trouvent sur http://fare.tunes.org/files/fwprc/.
- **httptunnel** se trouve sur http://www.nocrew.org/software/httptunnel/.
- **mailtunnel** se trouve sur http://www.detached.net/mailtunnel/.

Compréhension du problème

Quand vous comprenez un problème, vous avez fait la moitié du chemin vers la solution.

Donner un nom aux choses

Si vous voulez que cette méthode fonctionne, vous devrez comprendre comment elle fonctionne pour que, si quelque chose ne marche pas, vous sachiez où chercher.

La première étape pour comprendre le problème est de donner un nom aux concepts appropriés.

Comme d'habitude, nous allons ci-après appeler « client » la machine qui décide d'initialiser la connexion, ainsi que les programmes et les fichiers sur cette machine. Réciproquement, nous appelerons « serveur » celui qui attend les connexions et les accepte, ainsi que les programmes et fichiers sur cette machine. Le perçage de pare-feu est utile lorsque les deux machines sont séparées par un pare-feu, de telle sorte qu'il est possible pour le serveur d'accepter certaines connexions, alors qu'il n'est pas certain que le client puisse en accepter. Un tunnel sera créé entre les deux machines, ce qui permet un trafic IP complet malgré le pare-feu.

Habituellement, lorsque l'on perce un pare-feu, le client est la machine derrière le pare-feu : il a un accès limité à internet, mais peut d'une façon ou d'une autre ouvrir une connexion ou une autre sur le serveur. Le serveur est une machine avec un accès complet à internet, qui va servir de proxy pour le client afin qu'il accède à internet. Dans un VPN, les rôles peuvent être inversés, avec le client étant sur internet et le serveur servant de proxy au client afin d'accéder à certains réseaux privés.

Le problème principal

Le problème principal pour le perçage de pare-feu est de créer un tunnel : une connexion continue d'une machine cliente vers une machine serveur de l'autre côté du pare-feu, qui permet un échange bidirectionnel d'informations. Optionnellement, cette connexion devrait être sécurisée. Le problème annexe est de transformer cette connexion en un accès IP complet pour une utilisation transparente par les programmes normaux.

Pour le problème principal, nous considérerons que soit (1) vous pouvez établir des connexions TCP/IP normales du côté client du pare-feu vers un port sur une machine serveur où un sshd tourne ou peut être mis en fonctionnement, ou (2) vous pouvez d'une façon ou d'une autre établir une connexion telnet à travers un proxy telnet. Au cas où vous ne pourriez pas, nous allons vous diriger vers un autre logiciel qui permet de percer un tunnel à travers un pare-feu. Bien que nous ne donnions qu'une solution sécurisée dans le premier cas, vous pouvez bidouiller votre propre solution sécurisée dans les

autres cas, si vous comprenez le principe (si vous ne le comprenez pas, quelqu'un, par exemple moi, peut le faire pour vous contre rémunération).

Le problème annexe

Pour le problème annexe, les émulateurs d'IP (**pppd** ou SLiRPTM) sont lancés de chaque côté du tunnel.

Du côté qui veut un accès IP complet vers l'autre côté, il vous faudra lancer **pppd**. De l'autre côté, vous devez lancer **pppd** si vous voulez un accès IP complet dans l'autre sens, ou SLiRPTM si vous voulez empêcher tout accès. Consultez votre documentation **pppd** ou SLiRPTM habituelle pour plus d'informations, si vous avez des besoins spécifiques qui ne sont pas traités dans les exemples ci-dessous.

Bien qu'il s'agisse d'un concept banal, ça nécessite néanmoins quelques astuces toutes bêtes afin de fonctionner, car (a) si vous utilisez une quelconque session shell programmée interactive pour démarrer l'émulateur d'IP du serveur de n'importe quel côté, il vous faut synchroniser correctement le démarrage de l'émulateur d'IP de l'autre côté, afin de ne pas envoyer des saletés dans la session shell, et (b) les émulateurs d'IP sont conçus pour être lancés sur une interface « tty » : vous devez donc convertir votre interface tunnel en une tty.

Le point (a) ne représente rien de plus que le problème de synchronisation habituel, et n'existe même pas si vous utilisez **ssh**, qui s'occupe de manière transparente du lancement de commande du serveur.

Le point (b) requiert l'utilisation d'un simple utilitaire extérieur. Nous en avons fait un, **cotty** juste dans ce but.

< PIQUAGE DE CRISE>

Entre autres problèmes débiles dûs à l'étroitesse d'esprit des concepteurs de **pppd** (ceci n'est plus le cas dans les versions récentes de Linux), on peut seulement le lancer soit par un dispositif dans `/dev` ou par le tty courant. On ne peut pas le lancer par une paire de tunnels (ce qui serait la conception évidente). C'est parfait pour le **pppd** du serveur s'il y en a un, puisqu'il peut utiliser le `tty` des sessions **telnet** ou **ssh**; mais pour le **pppd** du client, cela entraîne un conflit en cas d'utilisation de **telnet** pour établir une connexion.

En effet, **telnet** veut, également, être sur un tty, il se comporte *presque* correctement avec deux tunnels, à part qu'il insistera encore pour faire des iotctl au tty courant, avec lequel il va interférer ; l'utilisation de **telnet** sans un tty impose également un régime tel que toute la connexion échouera sur des ordinateurs « lents » (**fwprc** 0.1 fonctionnait parfaitement sur un P/MMX 233, un délai d'attente de 6 sur un 6x86-P200+, et aucun sur un 486dx2/66). L'un dans l'autre, lors de l'utilisation de **telnet**, vous avez besoin de **cotty** comme démon pour copier la sortie d'un tty sur lequel fonctionne pppd sur un autre tty sur lequel fonctionne **telnet**, et inversement.

Si je trouve l'abruti (probablement un gars de MULTICSTM bien que il a dû y avoir des gens d'UNIXTM assez bêtes pour copier cette idée) qui a inventé le principe des dispositifs « tty » grâce auxquels on lit et on écrit à partir d'un « même » pseudo fichier, au lieu d'avoir des couples de tunnels propres, je l'étrangle !

</JE ME CALME>

La solution sécurisée : percer en utilisant ssh

Principe

Considérons que votre administrateur de pare-feu autorise les connexions TCP transparentes vers un port quelconque sur un serveur de l'autre côté du pare-feu (que ce soit le port du ssh normal, le 22, un autre port de destination, tel que le port http, le 80, ou autre), ou que vous vous débrouillez d'une façon ou d'une autre pour qu'un port quelconque d'un côté du pare-feu soit redirigé vers un port de l'autre côté (en utilisant **httptunnel**, **mailtunnel**, un tunnel sur le **telnet**, ou autre).

Vous pouvez alors lancer un **sshd** sur le port côté serveur, et vous y connecter avec un **ssh** sur le port côté client. Des deux côtés de la connexion **ssh** vous lancez des émulateurs d'IP (**pppd**), et là vous avez votre VPN, réseau privé virtuel, qui évite les restrictions stupides du pare-feu, avec un bonus en plus : la confidentialité grâce au cryptage (faites attention, l'administrateur du pare-feu connaît tout de même l'autre bout du tunnel, et toute information d'authentification quelle qu'elle soit que vous pouvez avoir envoyée avant de lancer le **ssh**).

Exactement la même technologie peut être utilisée pour construire un VPN, réseau privé virtuel, qui permet de regrouper de façon sécurisée des sites physiques en un seul réseau logique sans sacrifier la sécurité au niveau du réseau de transport entre les sites.

Exemple de session

Ci-dessous se trouve un exemple de script que vous pouvez adapter à vos besoins. Il utilise le système de rangée de **zsh**, mais vous pouvez l'adapter facilement à votre shell favori. Utilisez l'option **-p** pour que **ssh** essaie un autre port que le port 22 (mais à ce moment-là, veillez à bien lancer **sshd** sur le même port).

Notez que le script suppose que **ssh** peut s'ouvrir sans que vous ayez à taper interactivement votre mot de passe (en effet, son tty de contrôle sera connecté à **pppd**, alors s'il vous demande un mot de passe, c'est raté). Ceci peut se faire soit avec les clefs ssh dans votre `~/.ssh/authorized_keys` pour lesquelles un mot de passe n'est pas nécessaire, ou que l'on peut débloquer en utilisant **ssh-agent** ou **ssh-askpass**. Regardez votre documentation sur ssh. En fait vous pourriez aussi utiliser un script de chat pour entrer votre mot de passe, mais ce n'est assurément *pas* la chose à faire.

Si vous n'êtes pas **root** ou simplement si vous voulez protéger le réseau de votre client des connexions sortantes, vous pouvez utiliser **slirp** au lieu de **pppd** comme émulateur PPP du serveur. Il n'y a qu'à décommenter la ligne appropriée.

```
#!/bin/zsh -f
SERVER_ACCOUNT=root@server.fqdn.tld
SERVER_PPPD="pppd ipcp-accept-local ipcp-accept-remote"
#SERVER_PPPD="pppd" ### Ceci suffit normalement si c'est dans /usr/sbin/
#SERVER_PPPD="/home/joekluser/bin/slirp ppp"
CLIENT_PPPD=( pppd
        silent
        10.0.2.15:10.0.2.2
        ### Si vous voulez tester décommentez les lignes suivantes:
        # updetach debug
        ### Une autre option potentiellement utile (allez voir la section sur le routage) :
        # defaultroute
)
$CLIENT_PPPD pty "ssh -t $SERVER_ACCOUNT $SERVER_PPPD"
```

Notez que les options par défaut de votre `/etc/ppp/options` ou `~/.slirprc` peuvent casser ce script, enlevez donc toute option non désirée.

Notez également que `10.0.2.2` est le paramétrage par défaut pour **slirp**, ce qui peut ne pas fonctionner avec votre installation particulière. En tout cas, vous devriez de préférence utiliser une adresse dans l'une des catégories réservées par la RFC-1918 pour les réseaux privés : `10.0.0.0/8`, `172.16.0.0/12` ou `192.168.0.0/16`. Il se pourrait que le réseau local protégé par pare-feu utilise certaines d'entre elles et il est de votre responsabilité d'éviter les conflits. Pour une plus grande personnalisation, lisez la documentation appropriée.

Si le **pppd** de votre client est vieux ou non-linux (par exemple BSD) et n'a pas d'option **pty**, utilisez :

```
cotty -d -- $CLIENT_PPPD -- ssh -t $SERVER_ACCOUNT $SERVER_PPPD
```

Pièges : ne mettez pas les commandes données à cotty entre guillemets, car elles s'exécutent **exec()**telles quel, et n'oubliez pas de spécifier le chemin complet pour le **pppd** du serveur s'il n'est pas dans le chemin standard installé par **ssh**.

On laisse au lecteur la reconnexion automatique (conseil : l'option **nodetach** de **pppd** pourrait être utile pour ça).

La solution non sécurisée : percer en utilisant telnet

Principe

Si vous ne pouvez faire que du **telnet** (à cause d'un proxy **telnet**), cette solution pourrait bien vous convenir.

Le programme de perçage de pare-feu, **fwprc**, utilisera un proxy tty, **cotty**, qui ouvre deux dispositifs pseudo-tty, lance des commandes sur chacun des esclaves de ces dispositifs, et copie systématiquement chaque caractère que l'on sort sur le tty qui sert d'entrée pour l'autre commande. Une commande sera une connexion telnet vers le site serveur, et l'autre sera le **pppd**. **pppd** peut alors ouvrir et contrôler la session telnet avec un script de chat comme d'habitude.

En fait, si votre proxy telnet permet une connexion vers un port arbitraire, et si vous pouvez lancer un démon de manière sûre sur l'hôte serveur (avec relance planifiée [cron] en cas de plantage), vous feriez mieux d'écrire un programme qui connectera juste un port côté client au port côté serveur par le proxy, afin de pouvoir utiliser la solution sécurisée ci-dessus, en utilisant éventuellement une variante de

```
ssh -t -o "ProxyCommand ..."
```

(Soumettez moi une solution et je l'intégrerai volontiers à la distribution **fwprc**).

Remarque : si vous devez utiliser la solution non sécurisée basée sur le telnet, assurez vous que, dans votre session cible, il ne se trouve rien de confidentiel ou qui ne doive être bidouillé, puisque le mot de passe sera envoyé en clair sur internet. Si c'est à votre portée, un système ne demandant le mot de passe qu'une seule fois, ou un système de cryptographie explicite augmentera votre sécurité, ce qui, toutefois, rendra les scripts de connexion automatique bien plus complexes.

fwprc

J'ai écrit un script qui décrit de façon très détaillée comment percer les pare-feu, **fwprc**, disponible sur mon site, avec également **cotty** (qui est requis à partir de la version `0.2` de **fwprc**). Au moment où j'ai écrit ces lignes, la version la plus avancée de **fwprc** est `0.3e` et pour **cotty** `0.4c`.

Le nom « fwprc » est volontairement illisible et imprononçable, afin d'embrouiller votre administrateur système incompétent et paranoïaque sans doute responsable de ce pare-feu qui vous casse les pieds (bien sûr il peut y avoir des pare-feu légitimes également, et parfois même, ils sont indispensables ; la sécurité n'est qu'un problème de configuration *correcte*). Si vous devez le lire à voix haute, prononcez le de la pire façon possible et imaginable.

GRAND CONCOURS ! Envoyez moi un fichier audio avec un enregistrement audio numérique de votre prononciation de « fwprc ». Le prix pour la prononciation la plus mauvaise est une mise à niveau gratuite et le nom du gagnant sur la page du **fwprc** `1.0`!

J'ai testé le programme sur différentes configurations, en le configurant avec des fichiers ressources. Mais bien entendu, selon la loi de l'emmerdement maximum, ça plantera pour vous. N'hésitez pas à proposer les améliorations qui faciliteront la vie de ceux qui configureront après vous.

.fwprcrc

fwprc peut être personnalisé grâce au fichier `.fwprcrc` fait pour être le même des deux côtés du pare-feu. Avoir plusieurs configurations de rechange parmi lesquelles choisir est certes possible (par exemple, *moi* je le fais), et on laisse ça comme exercice pour le lecteur.

Pour commencer, copiez la section appropriée de **fwprc** (l'avant-dernière) dans un fichier nommé `.fwprcrc` dans votre répertoire home. Remplacez ensuite les valeurs des variables par des trucs qui correspondent à votre configuration. Enfin, copiez le sur l'autre hôte et testez.

Le comportement par défaut est d'utiliser **pppd** sur le client, et **slirp** sur le serveur. Pour modifier ceci, vous pouvez redéfinir la fonction appropriée dans votre `.fwprcrc` avec une ligne telle que :

```
remote_IP_emu () { remote_pppd }
```

Notez que SLiRPTM est plus sûr que **pppd**, et plus facile d'accès, puisqu'il ne requiert pas d'être root sur la machine serveur, et n'a pas besoin d'une configuration supplémentaire du pare-feu pour empêcher les connexions du monde extérieur sur le réseau derrière un pare-feu. La fonctionnalité de base dans SLiRPTM fonctionne plutôt bien, mais je n'ai pas réussi à faire marcher certains des plus qu'il est censé proposer (tel que le contrôle du temps d'exécution). Bien entendu, puisqu'il s'agit d'un logiciel libre, n'hésitez pas à aller dans la source pour carrément implémenter ou réparer n'importe quel dispositif dont vous aurez besoin.

Routage

Il ne suffit pas de percer le pare-feu. Il faut également router les paquets du côté client du pare-feu vers le côté serveur. Dans cette section, j'aborde les paramètres de base spécifiques au routage à travers un tunnel. Pour plus d'explications détaillées sur le routage, lisez les guides pratiques correspondants et les pages de manuel sur les réseaux, le routage et l'usurpation d'identité.

Il y a un truc

Le truc, c'est que, même si votre administrateur réseau vous demandait d'installer un routeur de votre côté client comme route par défaut, (ceci peut être utile si on veut une route spécifique vers les réseaux sur le client du pare-feu), il faudra installer PPP link comme route vers les réseaux sur le côté serveur.

En d'autres termes, votre route par défaut devrait pointer vers un routeur de n'importe quel côté du tunnel qui vous donne accès à internet.

Ce qui est primordial, c'est que les paquets envoyés au serveur hôte en tant qu'élément de fonctionnement du tunnel doivent être routés à travers votre réseau habituel (par exemple votre routeur ethernet par défaut), autrement, votre kernel aura des problèmes, vu qu'il essaie de router à l'intérieur du tunnel les paquets qui, précisément, devraient constituer l'extérieur du tunnel.

Ainsi donc, vous devrez paramétrer correctement les routes dans votre configuration de démarrage du réseau. L'emplacement précis de vos données de configuration de routage dépend de votre distribution, mais c'est généralement sous `/etc/init.d/network` ou `/etc/network/`; de même, votre configuration PPP se trouve généralement dans `/etc/ppp/`, et l'endroit normal pour configurer ces routes se trouve habituellement dans `ip-up` ou `ip-up.d/`. (Astuce : pour identifier les emplacements de fichier spécifiques à votre distribution, vous devez lire la documentation de votre distribution ou encore RTFM; sinon, utilisez **grep** récursivement dans votre `/etc`; au pire, repérez ce qui se passe au moment du démarrage, comme configuré dans votre `/etc/inittab`.)

Lorsque vous percez un tunnel dans un réseau protégé à partir d'un portable sur internet, le script **getroute.pl** (disponible sur la distribution **fwprc**) donne la route utilisée vers le serveur hôte qui représente l'autre bout du tunnel.

Une fois que vous pouvez router les paquets vers le côté serveur du tunnel, ça vous intéressera peut-être de configurer votre machine comme routeur pour tous vos copains du côté client du pare-feu ; vous aurez ainsi réalisé un véritable VPN partagé. Ceci n'est pas spécifique au perçage de pare-feu, alors lisez donc les guides pratiques correspondants sur les réseaux, le routage et l'usurpation d'identité. Egalement, pour des raisons de sécurité, veillez à bien installer un bon pare-feu sur votre machine, surtout si vous devez être routeur pour d'autres personnes.

Enfin, souvenez vous que si vous utilisez **pppd** sur le côté serveur du tunnel (par opposition au mode utilisateur **slirp**), vous devrez également configurer les routes qu'il faut et des règles de pare-feu du côté serveur du tunnel.

Exemple de routage

Dans cet exemple, votre machine cliente est connectée à un réseau local avec pare-feu grâce au dispositif ethernet `eth0`. Son adresse IP est `12.34.56.78`; son réseau est `12.34.56.0/24`; son routeur est `12.34.56.1`.

Votre administrateur réseau peut vous avoir dit d'utiliser `12.34.56.1` comme routeur par défaut, mais n'en tenez pas compte. Vous devez seulement l'utiliser comme route vers le côté client du pare-feu.

Supposons que le côté client du pare-feu est composé des réseaux `12.34.0.0/16` et `12.13.0.0/16`, et de l'hôte `11.22.33.44`. Pour les rendre accessibles par votre routeur client, ajoutez ces routes au script de démarrage de votre réseau global :

```
route add -net 12.34.0.0 netmask 255.255.0.0 gw 12.34.56.1
route add -net 12.13.0.0 netmask 255.255.0.0 gw 12.34.56.1
route add -host 11.22.33.44 gw 12.34.56.1
```

Vous devez également garder la route vers le réseau local du client, nécessaire pour le kernel 2.0 de Linux , mais pas pour le kernel 2.2 et suivants de Linux (ceci l'ajoute implicitement pendant le **ifconfig**):

```
route add -net 12.34.56.0 netmask 255.255.255.0 dev eth0
```

Par contre, vous devez *impérativement* enlever toute route par défaut de vos scripts. Supprimez ou mettez en commentaire une ligne telle que :

```
route add default gw 12.34.56.1
```

Remarquez qu'il est également possible d'enlever la route de la configuration du kernel en marche sans redémarrer, grâce à la commande suivante :

```
route del default gw 12.34.56.1
```

Vous pouvez ensuite obtenir de **pppd** l'installation automatique d'une route par défaut lorsqu'il démarre en utilisant son option **defaultroute** Sinon, vous pouvez l'ajouter plus tard :

```
route add default gw 10.0.2.2
```

Si vous ne voulez pas de **pppd** comme route par défaut, parce que l'accès internet est disponible de votre côté du pare-feu, et si vous voulez plutôt que le réseau `98.76.48.0/20` soit routé par le tunnel, sauf au départ de l'hôte `98.76.54.32` qui représente l'autre bout du tunnel, ajoutez les lignes suivantes à votre `/etc/ppp/ip-up`:

```
route add -host 98.76.54.32 gw 12.34.56.1
route add -net 98.76.48.0 netmask 255.255.240.0 gw 10.0.2.2
```

Si vous êtes sur un portable et que vous changez de réseau local, mais que vous voulez quand même garder votre route actuelle vers `98.76.54.32`, utilisez **getroute.pl** comme suit pour trouver automatiquement la bonne passerelle dans la commande **route add -host** :

```
$(getroute.pl 98.76.54.32)
```

Remarquez que si vous les avez dans votre `/etc/hosts`, vous pouvez utiliser des noms symboliques au lieu des adresses IP numériques (et vous pouvez même utiliser des FQDN, si vous pensez que le DNS ne plante jamais).

Perçage inverse

La logique

Des fois, seul un côté du pare-feu peut lancer des sessions telnet vers l'autre côté, cependant, un moyen de communication est possible (en général par le courrier électronique). Percer un pare-feu est toujours possible, en déclenchant, grâce à n'importe quel moyen de transmission de messages disponible, une connexion telnet du « bon » côté du pare-feu vers l'autre côté.

fwprc inclut du code pour déclencher de telles connexions à partir d'un courriel authentifié par OpenPGP ; il suffit d'ajouter **fwprc** comme filtre **procmail** aux messages utilisant ce protocole (instructions contenues dans **fwprc** lui-même). Remarquez cependant que si vous devez lancer **pppd** avec les privilèges appropriés, il vous faudra peut-être créer votre propre suid wrapper pour devenir root. Instructions incluses dans **fwprc**.

En outre, déclenchement authentifié ne signifie absolument pas connexion sécurisée. Il faut vraiment utiliser **ssh** (peut être en plus de telnet) pour des connexions sécurisées. Et puis méfiez vous de ce qui se passe entre le déclenchement d'une connexion telnet, et le moment ou **ssh** prend en main cette connexion. Toute contribution à ce sujet sera la bienvenue.

Obtenir le message de déclenchement

Si vous êtes sous un pare-feu, votre messagerie peut tout-à-fait être dans un serveur de messagerie central qui ne fait pas de filtrage procmail ou qui n'autorise pas les sessions telnet. Aucun problème! Vous pouvez lancer **fetchmail** en mode démon (ou comme tâche cron) pour interroger votre serveur de messagerie et distribuer le courrier à votre système linux qui, lui-même, aura été configuré pour utiliser **procmail** à la réception. Remarquez que si vous lancez **fetchmail** comme démon en arrière plan, il bloquera tout autre fetchmail que vous voudriez simplement lancer à d'autres moments, comme lorsque vous ouvrez un **fwprc**; bien entendu, si c'est possible, lancez également un démon fetchmail en tant qu'utilisateur bidon. Des scrutations trop fréquentes ne seront pas bonnes pour le serveur de messagerie ou pour l'hôte. Si elles sont trop peu fréquentes, vous devrez attendre avant que le message ne soit lu et que la connexion inverse soit établie. J'utilise une fréquence de scrutation de deux minutes.

Autres outils automatiques pour le perçage inverse

Un autre moyen d'interroger un serveur pour voir les messages, quand on n'a pas de boîte de messagerie, mais qu'on a bien un accès FTP vers l'extérieur, est d'utiliser un tunnel FTP.

Comme outil pour maintenir une connexion permanente entre un hôte sous pare-feu et un proxy externe, afin d'exporter des services depuis l'hôte vers l'extérieur, il y a le tunnel pare-feu.

Remarques finales

Autres réglages

Je n'ai aucune idée sur la manière de percer des pare-feu avec des systèmes d'exploitation de niveau inférieur, mais vous pouvez prendre un de ces vieux ordinateurs hors d'usage (quasiment tout ce qui a 8 Mo de RAM et une carte ethernet devrait aller), y installer Linux ou BSD, et percer le pare-feu avec,

tout en servant de routeur pour les autres machines fonctionnant avec des SE de niveau inférieur. Lisez les guides pratiques correspondants sur le routage, l'acheminement IP, NAT, etc.

Je ne connais pas les détails, mais il existe un outil prometteur pour percer les pare-feu : le Bouncer de Chris Mason, qui joue le rôle de proxy SOCKS par-dessus SSL.

Il y a d'autres sortes de pare-feu que ceux qui autorisent les connexions ssh ou telnet directes. Tant qu'un flot continu de paquets peut transmettre des informations à travers un pare-feu dans les deux directions, il est possible de le percer ; seulement, le prix pour écrire le perforateur peut être plus ou moins élevé.

Cela peut être très facile : ainsi, on a remarqué qu'il suffit de lancer **ssh** sur un maître pty et faire du **pppd** sur l'esclave tty. Si on le souhaite, on peut même le faire sans qu'il soit question de pare-feu, juste pour construire un VPN sécurisé (Virtual Private Network). Le VPN mini-HOWTO donne tous les détails nécessaires à ce sujet. Comme exercice, nous vous invitons, à modifier **fwprc** pour utiliser cette technique, ou peut-être même pour l'utiliser à l'intérieur d'une précédente session **fwprc** non sécurisée.

Maintenant si le seul chemin à travers le pare-feu est un proxy WWW (ce qui est habituellement un minimum pour un réseau connecté à internet), on pourra utiliser le script ssh-https-tunnel de Chris Chiappa.

Autre programme prometteur pour percer à travers http : le httptunnel de Lars Brinkoff, une combinaison serveur et client http permettant une connexion TCP/IP par tunnel grâce au protocole http. On devrait ensuite pouvoir lancer **fwprc** (de préférence par-dessus **ssh**) sur cette connexion, bien que je n'aie pas encore essayé. Quelqu'un pourrait-il tester et faire un rapport ? Remarquez que **httptunnel** est toujours en cours de développement, vous pouvez donc aider à implémenter les fonctionnalités qui lui manquent actuellement, telles que les connexions multiples, et/ou servir des pages bidon pour leurrer les administrateurs méfiants de pare-feu excessivement hermétiques.

Quel que soit ce qui passe à travers votre pare-feu, que ce soit telnet, http ou autres connexions TCP/IP, ou des choses vraiment bizarres comme les requêtes DNS, les paquets ICMP, le courrier électronique (lisez mailtunnel, icmptunnel), ou que sais-je encore, vous pouvez toujours écrire une combinaison tunnel client/serveur, et lancer un **ssh** et/ou une connexion PPP à travers celui-ci. La performance pourrait ne pas être très bonne, en fonction de la vitesse effective de communication de l'information après avoir payé les charges dûes au codage de l'ensemble des filtres et proxies, mais un tel tunnel est toujours intéressant tant qu'il peut au moins servir à utiliser **fetchmail**, **suck**, et autres programmes non interactifs.

Si vous devez traverser une ligne 7 bits, vous devrez utiliser SLIP au lieu de PPP. Je n'ai jamais essayé, parce que les lignes sont plus ou moins 8 bits maintenant, mais ça ne devrait pas être difficile. Si nécessaire, rabattez vous sur le Term-Firewall mini-HOWTO.

Si vous avez une connexion 8 bits propre et que vous êtes root sur linux des deux côtés du pare-feu, vous pouvez utiliser ethertap pour de meilleures performances, en encapsulant des communications ethernet brutes en plus de votre connexion. David Madore a écrit sur les tunnels ethertap-sur-TCP et ethertap-sur-UDP ftp://quatramaran.ens.fr/pub/madore/misc/. Il reste à traiter de ethertap-sur-tty pour combiner avec des outils comme fwprc.

Si vous cherchez une performance supérieure à celle obtenue en payant un tunnel à communication séquentielle, niveau espace utilisateur, à travers lequel lancer PPP, vous êtes dans la situation très difficile où vous devrez rebidouiller une drôle de pile IP, en utilisant (par exemple) les fonctions de

type ’functor’ des protocoles par paquets du projet Fox. Vous obtiendrez alors une IP sur HTTP, une IP sur DNS, une IP sur ICMP, ou autre, directe, qui requiert non seulement un protocole élaboré, mais également une interface vers un noyau de SE, qui sont tous deux chers à implémenter.

Pour finir, si vous n’êtes pas en train de vous battre contre un pare-feu excessivement hermétique, mais que vous faites juste votre propre VPN, il y a un large éventail d’outils VPN, et bien que les trucs que je présente soient simples, qu’ils fonctionnent bien, et qu’ils peuvent être suffisants pour vos besoins, ça serait peut-être pas mal de rechercher dans cette offre évolutive (dont je ne connais pas grand-chose) une solution qui correspond à vos besoins au niveau performance et maintenabilité.

Le suivi de ce petit guide

J’ai pensé qu’il était nécessaire de l’assurer, mais je n’ai pas beaucoup de temps pour ça, donc ce petit guide est très sommaire. Il va donc rester ainsi jusqu’à ce que j’obtienne assez de retours d’information pour savoir quelle section améliorer, ou mieux, jusqu’à ce que quelqu’un vienne se charger du suivi de ce petit guide. Tout retour d’information est le bienvenu. Toute aide est la bienvenue. La prise en charge du suivi du petit guide est la bienvenue.

En tout cas, les sections ci-dessus montrent que de nombreux problèmes peuvent être facilement résolus si quelqu’un (vous ?) y consacre un peu de temps (ou d’argent, en engageant quelqu’un d’autre) ; il n’y a qu’à s’asseoir et écrire : le concept n’est pas difficile, bien que, dans le détail, cela puisse ne pas être simple ou évident.

N’hésitez pas à proposer d’autres problèmes, et, espérons-le, d’autres solutions, pour ce petit guide.

Documents sur le sujet

Le LDP publie de nombreux documents en rapport avec ce petit guide. tout particulièrement Linux Security Knowledge Base, le HOWTO VPN et le petit guide VPN. Pour des questions plus générales sur le réseau, le routage et les pare-feu, commencez avec le Networking Overview HOWTO. Regardez également le Linux Firewall and Security site.

Là encore, lorsque vous rencontrez un problème avec un programme, le réflexe pour tout utilisateur de Linux devrait être de Lire le Pstain [sic] de Manuel [RTFM : Read The Fscking Manual] sur les programmes en question.

Le mot de la fin

Je suis arrivé à la conclusion que, de la même façon que le besoin en Design Patterns venait directement du fait que les gens utilisaient des langages inférieurs tels que le C++TM ou le JavaTM qui ne permettent pas d’exprimer directement des constructions de programmation de plus haut niveau (alors que de bons langages tels que le LISPTM permettent de les exprimer), on peut dire que le besoin en guides pratiques vient directement du fait que les systèmes LinuxTM et UNIXTM sont des systèmes d’exploitation inférieurs qui ne permettent pas d’exprimer directement les tâches que les gens essayent de faire avec, qui sont pourtant simples.

Si vous pensez que tout ce bidouillage de scripts stupides et de guides pratiques idiots est trop compliqué et qu’un système d’ordinateur convenable devrait nous automatiser tout ça, alors bienvenu au club des allergiques comme moi à UNIX (UNIX haters) et de ceux qui détestent les systèmes de bas niveau actuels, et aspirent à des systèmes informatiques déclaratifs qui prennent en charge tous les détails sans intérêt et nous laissent nous concentrer sur les choses importantes (jetez un ½il, si ça vous

dit, à mon propre projet TUNES).

Copie supplémentaire de l’avis très important de non responsabilité - croyez le !!!

« *Par la présente je décline toute responsabilité quant à l’utilisation que vous ferez de cette méthode d’effraction [hack]. Si ça se retourne contre vous d’une manière ou d’une autre, c’est pas de pot. Et ce n’est pas ma faute. Si vous ne comprenez pas les risques encourus, laissez tomber. Si vous utilisez cette méthode et qu’elle permet à des vandales sans scrupules de pénétrer dans les ordinateurs de votre société et que ça vous coûte votre boulot et des millions d’euros à votre entreprise, eh bien c’est pas de pot. Ne venez pas pleurer chez moi.* »