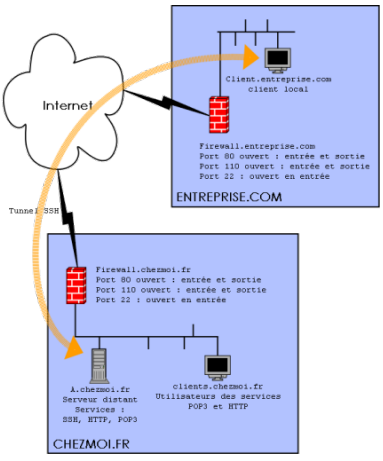
http://clx.asso.fr/spip/?Faire-passer-un-flux-HTTP-ou-autre

SSH

Faire passer un flux HTTP (ou autre) dans un tunnel SSH

- Documentations - Installation / Administration de base de Linux -



Date de mise en ligne : jeudi 29 août 2002

Le protocole SSH est un moyen trÚs pratique de se connecter à distance sur un serveur (ou une station), sans laisser à une tierce personne la chance de récupérer les données qui transitent. En somme, il s'agit d'un telnet crypté.
Mais il permet également de *forwarder* un *port* local vers un des ports de la machine distante de maniÚre à créer un VPN (Virtual Private Network ou Réseau Privé Virtuel [1].).
Dans cet article, nous allons :

1. Créer un tunnel ssh entre un serveur distant sur lequel vous avez la main et un client au sein d'une entreprise ;
2. Paramétrer le client pour "faire comme si" les services du serveur distant étaient hébergés en local. Ainsi depuis le poste client, vous pourrez naviguer sur le site du serveur distant, consulter vos mails, etc.) de maniÚre sécurisée et transparente.

[conventions typographiques]

Rappel : Qu'est que SSH ?

`DESCRIPTION` [nota]

```
ssh (SSH client) is a program for logging into a remote machine and for executing commands on a remote
machine.
 It is intended to replace rlogin and rsh, and provide secure encrypted communications between two
untrust-ed hosts over an insecure network.  X11 connections and arbitrary TCP/IP ports can also be forwarded
over the secure channel.

 ssh connects and logs into the specified hostname. The user must prove his/her identity to the remote
machine using one of several methods de-pending on the protocol version used.
```

Dans cet extrait de la page de man de la commande `ssh`, on comprend que le protocole SSH est un moyen de saisir des commandes à distance, en faisant transiter les données, entre le client et le serveur, de manière cryptée.
En pratique, ssh [2] utilise les libraires openssl, et un algorithme à clé publiques/privées RSA (ou DSA pour la version 2 de ssh).

`ssh` se comporte exactement comme `telnet`, mais avec une connexion cryptée.
Le client encrypte les données qu'il envoie au serveur, qui les décryptera, et réciproquement.
En plus, grâce au mécanisme de clés, il est tout à fait possible de se faire identifier, non par un couple login/password classique,
mais par une clé RSA (resp. DSA) présente dans le répertoire `.ssh` de l'utilisateur, sur l'ordinateur client.

Pour plus de détails sur les clés, reportez-vous à l'excellente doc sur GPG et au site UZINE.

Rappel 2 : qu'est ce qu'un port ?

Le protocole TCP permet d'établir des connexions entre machines, c'est-à-dire entre un client, qui demande une connexion, et un serveur, qui écoute les demandes de connexion et y répond.
Pour différencier les services (par exemple la messagerie électronique du serveur Web), on attribue de manière arbitraire [3] un *numéro de port* à chaque service.
De même, on différencie les clients par un couple adresse IP/port "source", avec un port pris dans une plage comprise entre 1025 et 32000, ce sont des ports dits *non privilégiés*, n'importe qui à le droit de les utiliser.
Cela permet d'avoir plusieurs connexions simultanées en provenance
d'un seul client vers un même service présent sur un serveur distant.

Dans le fichier `/etc/services`, vous trouverez une liste de ports couramment utilisés pour les services, à savoir le port 25 associé au démon SMTP, le port 110 au démon POP3, le port 80 au démon HTTP, 22 au démon SSH, etc.

PNG - 66.8Â ko

Sur ce schéma réalisé avec `dia`, le réseau d'entreprise s'appelle ENTREPRISE.COM, le réseau du serveur distant se nomme CHEZMOI.FR. *serveur.chezmoi.fr* désigne le PC familial par exemple et *a.entreprise.com* correspond au poste client.

Les pare-feux (sur *firewall.entreprise.com* dans notre exemple) ne filtrent les connexions TCP que sur un nombre restreint de ports, en fonction des besoins des utilisateurs et des services de base (webmail, accès distants...) mis en place par l'entreprise pour ses collaborateurs.
En général, les ports 80, 110, et parfois 25 sont ouverts pour les connexions du réseau local (ou LAN) vers l'extérieur.
Pour les connexions entrantes, sur les équipements de filtrage destinés à la protection du LAN, un bon administrateur ne laissera ouverts depuis l'extérieur que les ports utilisés par les ordinateurs qui doivent être accessibles depuis Internet. Il s'agit, généralement, là aussi des ports 80, 25 et 110, ainsi que le port 22 pour l'administration à distance.

`sshd` peut écouter sur un port arbitraire

On vient de le voir, les administrateurs qui font bien leur travail ne laissent pas sortir les connexions à destination des ports SSH. Mais le port HTTP (port 80) reste ouvert.

Si, depuis un réseau (*entreprise.com*) configuré ainsi, vous désirez accéder à une machine distante [4] via SSH, vous devrez utiliser un autre moyen qu'une connexion directe sur le port 22.
La solution consiste à forcer le démon SSH du poste client (*a.entreprise.com*) à écouter sur un des ports accessibles du serveur distant (*serveur.chezmoi.fr*), généralement le port 80 ou 110.
Vous trompez ainsi le pare-feu de votre entreprise (*firewall.entreprise.com*) sur le type de connexion que vous tentez d'établir. Il croit laisser passer une requête HTTP ou POP.

En pratique, il faut modifier le fichier `/etc/ssh/sshd_config`. La
syntaxe de ce dernier est simple et rappelle celle du fichier de configuration d'Apache. Repérez les lignes suivantes :

```
# What ports, IPs and protocols we listen for
 Port 22
```

Ajoutez le port 110 ou 80, ou les deux afin d'obtenir le résultat suivant :

```
Port 22,110,80
```

Redémarrez le démon `sshd`, soit par un `kill -HUP` *PID de* `sshd`, soit par un `/etc/init.d/sshd restart` (ou encore /etc/rc.d/init.d/sshd restart ou `rcsshd restart`).
Attention, il n'y a pas d'espace entre la virgule et le nombre qui la suit.

Attention également de ne pas couper la branche sur laquelle vous êtes assis. Il est bien évident que vous ne POUVEZ PAS redémarrer `sshd` si vous êtes connecté sur la machine distante via `ssh` (!).
[5].

La commande `screen` ouvre la voie d'une troisième solution, pour redémarrer le démon à distance.
`screen` est une commande qui permet de dissocier le *shell* du *terminal* ; le `sh` du `ssh`. [6].
Grâce à `screen`, si la connexion vient à être coupée, les commandes saisies et les programmes lancés continueront à s'exécuter et le démon `sshd` sera redémarré.

Vérifiez ensuite les erreurs éventuelles dans le fichier `/var/log/messages` par un

```
tail -f /var/log/messages
```

(sur serveur.chezmoi.fr).

Si la configuration de *firewall.entreprise.com* et *firewall.chezmoi.fr*, laissent passer les demandes de connexion sur le port 80 ou 110, vous pouvez vous connecter à *serveur.chezmoi.fr* avec la commande `ssh -p 80 nom@machine.domaine` (resp. 110). Un client SSH pour Windows, putty, permet également d'établir une telle connexion en modifiant le numéro de port dans la fenêtre principale de session (remplacer 22 par 80, resp. 110 dans le champ adéquat) [7].

Changer les ports utilisés par les services

PNG - 12.8Â ko
Comment se connecter au port 110 ou 80 de *serveur.chezmoi.fr* avec SSH, alors que cet ordinateur héberge déjà un serveur POP3 et un
serveur HTTP ?
Il y a une astuce pour empêcher plusieurs services d'écouter les mêmes ports (ici `sshd` et `httpd` écoutent sur le même port : le port 80).

Changer le port sur lequel le démon SSH écoute
Plutôt que laisser écouter le démon `sshd` sur toutes les adresses locales de la machine, c'est-à-dire l'adresse de votre
LAN, l'adresse délivrée par votre fournisseur d'accès Internet (FAI) [8] et l'adresse localhost, nous allons affiner un peu la configuration de `sshd`, en associant les ports à des adresses précises.
Dans le `/etc/ssh/sshd_config` de *serveur.chezmoi.fr*, supprimez les modifications précédentes de manière à obtenir `Port 22`. Insérez ensuite les lignes suivantes :

```
ListenAddress 127.0.0.1:22
```

```
ListenAddress [VotreAdresseProvider]:110
ListenAddress [VotreAdresseProvider]:80
```

Que disent ces lignes ? Que le démon `sshd` va écouter sur le port 22 si les demandes entrent par l'adresse localhost (127.0.0.1), et sur les ports 110 et 80 si les requêtes parviennent à l'adresse publique *VotreAdresseProvider*, fournie par votre FAI .

Pour trouver *VotreAdresseProvider*, saisissez la commande `ifconfig ppp0` par exemple (avec une liaison RNIS, il s'agit de ippp0). Repérez ensuite *inet adr* qui correspond à l'adresse attribuée par votre FAI.

L'astuce, c'est que le démon SSH n'écoute pas sur les ports 80 et 110 sur l'adresse localhost, cela sera utilisé par la suite.

Configuration du serveur Apache

Il faut maintenant configurer le serveur HTTP Apache pour qu'il *n'écoute pas* sur le port 80 associé à l'adresse publique de votre FAI, mais uniquement sur le port 80 de l'adresse localhost. En premier lieu, nous allons le forcer à écouter sur un autre port : 8081.

Ouvrez le fichier `/etc/apache/httpd.conf` (ou `/etc/httpd/conf/httpd.conf` ou `/etc/httpd/httpd.conf` en fonction de votre distribution (la commande `locate httpd.conf` devrait vous renseigner sur l'emplacement exact). Remplacez 80 par 8081 dans la ligne :

```
Port 80
```

Vous obtenez alors :

```
Port 8081
```

Nous verrons plus tard comment préciser l'adresse sur laquelle les serveurs HTTP écoutent les demandes de connexion.

Redémarrez ensuite apache à l'aide de `kill -HUP` *PID du* `httpd` qui a le plus petit numéro ou de `/etc/init.d/apache restart`, et vérifiez qu'il écoute bien sur le port 8081 en vous connectant via telnet :

```
# telnet localhost 8081
 Trying 127.0.0.1...
 Connected to localhost.
 Escape character is '^]'.
```

Tapez

```
GET /
```

Ajoutez une ligne vide (appuyez sur la touche Entrée), et la page index.html devrait apparaître (enfin son code source, bien sûr).

Forwarder un port avec le protocole SSH

La partie *serveur.chezmoi.fr* est maintenant configurée, `ssh` écoute sur le port 80, est joignable depuis votre client distant, Apache écoute sur le port 8081, et sur localhost:80

Une autre astuce va être employée afin de créer le tunnel SSH. Nous allons dire au client SSH, sur *a.entreprise.com* d'écouter sur le port 80, et de le *binder* (c'est-à-dire *relier*) à un port donné de *serveur.chezmoi.fr*.

Votre client distant (a.entreprise.com) enverra toutes les demandes de connexion le port 80 qui lui seront faites au serveur distant, *serveur.chezmoi.fr*, qui se chargera de faire parvenir la connexion à l'adresse IP et le port que vous avez précisé à l'ouverture de la connexion SSH. C'est, en quelque sorte, une téléportation de connexions IP. Un peu compliqué à comprendre, mais très très pratique à l'usage. D'ailleurs, d'autres ont fait beaucoup mieux que moi, en terme de pédagogie sur SSH.

Nous allons demander de *forwarder* [9] le port 80 de localhost, vers le port 8081 du *serveur.chezmoi.fr*.
C'est exactement comme si on faisait un lien symbolique entre les deux ports, en utilisant le tunnel SSH établi entre les serveurs [10].

Dans le man, ils expliquent le *forwarding* (en anglais sur ma machine) de la sorte :

```
# man ssh
 ...
 -L port:host:hostport
 Specifies that the given port on the local (client) host is to be forwarded to the given host and port on the remote side.
 This works by allocating a socket to listen to port on the local side, and whenever a connexion is made to this port, the connection is forwarded over the secure channel, and a connection is made to host port hostport from the remote machine.  Port forwardings can also be specified in the configuration file.  Only root can for-ward privileged ports.
```

Attention, il est bien précisé que seul l'utilisateur **root** a le droit de forwarder le port 80 local vers une machine distante via SSH.

En pratique, ça se passe comme ça :

```
ssh -p 80 nom@serveur.chezmoi.fr -L 80:localhost:8081
```

Ce qui signifie que le client SSH enverra toutes les demandes de connexion qui lui seront faites, à *serveur.chezmoi.fr* , qui se chargera de les acheminer sur localhost:8081 (c'est à dire lui même).

Une fois votre mot de passe tapé, si vous n'utilisez pas de clé RSA, la connexion SSH est établie. Vérifiez que le tunnel est bien fonctionnel (à partir d'une autre console) :

```
# telnet localhost 80
 Trying 127.0.0.1...
 Connected to localhost.
 Escape character is '^]'.
```

Tapez

```
GET /
```

suivi d'une ligne vide, la page index.html devrait apparaître. Vous êtes sûr que le tunnel SSH fonctionne, et que le port 80 est bien *forwardé* sur votre serveur.

```
# telnet localhost 8081
 Trying 127.0.0.1...
```

```
Connected to localhost.
Escape character is '^]'.
```

Tapez

```
GET /
```

suivi d'une ligne vide, et la page index.html devrait apparaître.

J'attire votre attention sur ce point. Vous avez, grâce à la commande précédente, *forwardé* un service, pour *faire comme si* le service Web hébergé sur *serveur.chezmoi.fr* était hébergé sur votre client *a.entreprise.com*. Si vous ouvrez votre navigateur web, et si vous vous connectez sur "localhost", la pseudo-interface locale, vous accédez aux services Web de votre serveur.
Il est tout à fait possible de faire de même avec l'ensemble des services de votre serveur distant, afin de les utiliser depuis votre client local. Il vous suffit d'ajouter autant de fois le paramètre `-L` que nécéssaire, lorsque vous lancez `ssh`.

Utiliser Putty

Si vous n'avez pas de poste sous Linux à disposition, mais une machine sous Windows, la version de développement du logiciel Putty vous permettra d'utiliser les fonctions de *port forwarding* de `ssh`.
Téléchargez la version de développement, et affichez les paramètres **SSH / Tunnels** afin de définir une translation du port 80 local, vers l'adresse `localhost:8081`. La capture d'écran ci-dessous vous montre les paramètres à préciser :

La page tunneling de putty
Puis cliquez sur ajouter pour créer le tunnel.
N'oubliez pas de sauvegarder la configuration que vous venez de définir afin de la réutiliser lors de vos prochaines connexions sur *serveur.chezmoi.fr*.
Je ne vais pas ici vous donnner le mode d'emploi de Putty, pour cela reportez-vous au site de putty. Ah ben oui, c'est en anglais. Eh, oh, je suis pas votre père, hein ?

Notez bien qu'il est là aussi possible de *forwarder* plusieurs services, pour que le client accède par exemple, au serveur Web, à la messagerie, à sa base de données mysql, à un serveur vnc (pour de la prise de contrôle à distance) vers un PC sous Windows, sous Linux, etc.

Apache ne marche plus très bien...

Votre tunnel SSH est effectif. Vous vous connectez à votre serveur sur le port 80, et le port 80 local du poste client *a.entreprise.com* est redirigé vers le port 8081 local de *serveur.chezmoi.fr*.

Le serveur distant fait partie d'un LAN, et le serveur Web Apache ne marche plus... Les utilisateurs de *chezmoi.fr* ne sont pas contents de rajouter le numéro de port :8081 dans la barre d'adresse de leur fureteur Web à chaque fois qu'ils tentent de se connecter au serveur Web de *serveur.chezmoi.fr*. C'est compréhensible. Pour les calmer, il existe une solution du même tonneau que celle que nous avons vue pour `ssh`. Dans le fichier `/etc/apache/httpd.conf` de *serveur.chezmoi.fr*, ajoutez les lignes :

```
Listen 127.0.0.1:80
```

```
Listen serveur.chezmoi.fr:80
```

Redémarrez Apache, et hop, plus de problèmes depuis le réseau local de votre serveur distant (*serveur.chezmoi.fr*).

Et mon cucipop ?

Bon. Ben dans le script de conf de `cucipop`, y'a pas un truc comme "Ecouter sur le port nÂ°XX", des fois ?
Allez, *man* et *google* sont vos amis. J'ai des mails à lire, moi. Je l'ai déjà dit, je suis pas otre père, non mais des fois !

Post-scriptum :

Si vous souhaitez aller plus loin, un support de cours TCP/IP est disponible sur le site Web du LAISSUS.

Vous pouvez consulter le trÚs bon et trÚs classique bouquin de **Tannenbaum**, ou celui de **Pujolles**. Attention, c'est technique (et sauvage, accrochez-vous). Pointez-vous dans n'importe quelle bonne librairie scientifique, demandez le **Tannenbaum**, vous aurez un pavé de 500 Pages.

Sinon, l'article Construire son réseau d'entreprise, nettement plus abordable, techniquement parlant, vous donnera de sérieuses bases en réseau.

Pour en savoir plus sur les VPN, vous pouvez consulter VPN AccÚs securisé à des pages web privées et transferts via ftp.

Enfin, les INCOUTOURNABLES *HOWTO* :

- VPN HOWTO
- NET4 HOWTO
- Network Administrator Guide (NAG) qui existe aussi en version Française, chez O'Reilly, dans toutes les libraires dignes de ce nom voire moins, puique je l'ai même vu à Carrefour (!).

---

[1] Il s'agit d'une solution pour le télétravail et la sécurité.

[conventions typographiques] :

- Un terme en majuscules (SSH) représente le nom du protocole,
- Un terme en police fixe (`ssh`) représente la commande à taper telle quelle,
- Un nom en *italique* (*serveur.chezmoi.fr*) représente un nom de machine, de domaine, ou un terme anglais pour lequelle une traduction nuirait à la comprehension (déjà sérieusement hypothéquée ;-).

[nota] toujours commencer une présentation par une page de man. Ça forge le caractère.

[2] Du moins la version que nous utilisons au CLX, c'est à dire Openssh.

[3] L'IANA recense tous les services officiellement utilisés et les numéros de port associés. On apelle d'ailleurs ces ports (entre 1 et 1024) des ports privilégiés, seul root a le droit d'y faire écouter un démon. La liste n'est pas exhaustive, vous comprendrez pourquoi plus bas.

[4] le PC familial, par exemple, sur lequel sont installés un certain nombre de services tels que POP SMTP ou HTTP. Sur notre schéma, c'est serveur.chezmoi.fr.

[5] En effet, le redémarrage du démon `sshd` se fait en deux étapes.

1. arrêt du démon

2. démarrage du démon
   Si vous perdez la connexion â€” ce qui est logique â€” après la première commande, tous les process fils de votre terminal seront tués, dont le process de démarrage de `sshd` ; `sshd` ne sera pas redémarré.

Si vous devez redémarrer, à distance, le démon `sshd` et que le -HUP ne fonctionne pas, et si votre `/etc/ssh/sshd_config` ne contient pas d'erreur, vous pouvez programmer un redémarrage de `sshd` dans la *crontab*. Vous perdrez la connexion avec le serveur, c'est vrai, mais le processus de redémarrage ne sera pas tué â€” puisque le père de ce process, c'est `cron` â€ contrairement à un `/etc/init.d/sshd` restart tapé depuis une console `ssh`.

[6] Un `ssh` sur un client distant ouvre une fenêtre d'affichage, une connexion sur le serveur, et exécute un shell pour exécuter les commandes.
`screen` ouvre un second *shell* un peu particulier. La commande `screen -r` permettra de retrouver votre session `ssh` initiale ; vous pouvez même forcer `screen` à détacher la session et la rattacher au terminal que vous utilisez, avec un `screen -rd`, par exemple pour récupérer une session ouverte sur l'un des terminaux virtuels de votre serveur distant, et vice-versa. `man screen` pour en savoir plus.

[7] Vous comprenez maintenant qu'il est possible de faire tourner n'importe quel service sur n'importe quel port, l'usage, et l'IANA, veulent pourtant que l'on respecte au maximum les fonctions associées aux différents ports â€” pour éviter de s'y perdre â€.

[8] d'autant plus si votre machine est connectée à Internet via l'ADSL

[9] à ne pas confondre avec *translater*, qui désigne une autre opération en réseau.

[10] Il est tout à fait possible, par ce même procédé, de *forwarder* n'importe quoi n'importe où. Dans les deux sens. La seule limite étant l'utilisation, à un moment donné d'un des port de l'un des deux bout du tunnel ; mais on peut très bien faire un tel lien entre un serveur Web, chez *entreprise.com*, et un des ports de *serveur.chezmoi.com* (puisque c'est l'un des bouts du tunnel ssh)