Se connecter chez soi depuis partout et sans encombre grâce à SSH et Tor

- Objet : Utiliser SSH avec Tor pour se connecter facilement à des serveurs distants
- Niveau requis : Intermédiaire
- Débutant, à savoir : Utiliser GNU/Linux en ligne de commande, tout commence là !.
- Suivi :
  à-placer
  - Création par vakuy 03/04/2018
  - Testé par <...> le <...> Fix Me!
- Commentaires sur le forum : Lien vers le forum concernant ce tuto [1] Fix Me!

Introduction

But

Le but de ce tutorial est d'apprendre à configurer un serveur et un client SSH fonctionnant avec Tor.

A quoi ça sert

Si vous vous intéressez à ce tutoriel, vous connaissez et utilisez certainement déjà SSH, un protocole de communication sécurisé pour se connecter à une autre machine Linux (ou MacOS), que ce soit en réseau local ou à distance. En réseau local, aucun problème: il suffit d'indiquer l'adresse IP interne associé à la machine à laquelle on souhaite se connecter, exemple:

```
ssh toto@192.168.1.100
```

Les choses se compliquent quand on souhaite se connecter à distance. Imaginez que vous avez un serveur chez vous et que vous souhaitez l'administrer depuis l'extérieur. Vous ne pouvez évidemment pas indiquer son adresse locale. Pour vous y connecter, vous devez donc:

- Activer le port forwarding sur votre routeur pour vous assurer que les connexions SSH soient redirigées sur votre serveur
- Sécuriser votre serveur avec des programmes comme fail2ban suite aux centaines de tentatives de connexion que vous allez immanquablement recevoir une fois le port forwarding activé
- En cas d'adresse IP dynamique (cas de la plupart des clients lambdas des FAI), s'inscrire à un service de DNS dynamique

En plus de ces inconvénients, il se peut que pour une raison ou une autre vous ne souhaitiez pas que votre adresse IP publique soit associée à un service SSH (recherche d'anonymat).

En revanche, une connexion SSH via Tor procure les avantages suivants:

- Pas besoin d'ouvrir des ports sur le routeur et d'IP forwarding
- Connexion directe au service en contournant tous les firewalls (pratique par exemple pour se connecter à une machine virtuelle)
- Anonymat garanti: personne ne sait où est hébergé votre service ni même que vous vous y

connectez
- Risques d'attaque ou de tentative de bruteforcing limités, voire nuls suivant la configuration

Au chapitre des inconvénients, on notera tout de même:

- Une vitesse de connexion en général nettement en dessous d'une connexion classique
- Des interruptions de service ou time out assez fréquentes

Comment ça marche

Vous avez sans doute déjà entendu parler du darknet et de ces sites en .onion qui ne sont accessibles que par Tor (en général son navigateur, Tor Browser). Ces sites sont en fait des *hidden services*, des services cachés générés très facilement en modifiant quelques lignes dans le fichier de configuration de Tor sur une machine Linux. C'est exactement ce que nous allons faire ici, sauf qu'au lieu d'écouter sur le port 80 ou 443 (http/https), nous allons écouter sur le port 22, port par défaut de SSH. Puis nous allons configurer le client pour qu'il puisse se connecter à cette adresse .onion par Tor.

Ce n'est probablement pas pour vous si

- Vous ne souhaitez pas ou ne pouvez pas utiliser Tor
- Vous avez besoin d'un service 100% fiable 24/24 (il arrive que ça ne fonctionne pas)
- Vous ne supportez pas les lenteurs occasionnées par Tor

Installation

Matériel requis

- Une connexion internet
- Deux machines Linux (ici debian 9) avec le logiciel Tor installé et mis à jour

Note: pas besoin d'avoir des machines distantes, il est tout à fait possible d'obtenir le même résultat avec des machines virtuelles sur le même ordinateur

Première partie: installation de Tor sur le serveur et configuration

On commence par installer Tor et ssh **sur le serveur**:

```
sudo apt install tor ssh
```

Note: vérifiez que l'installation comprend les paquets openssh-client et openssh-server, le métapaquet ssh les installe les deux sur debian 9, sur une autre distribution il est possible qu'il faille les préciser séparément

Création du hidden service

On se rend dans le fichier de configuration de tor:

```
sudo nano /etc/tor/torrc
```

On cherche la section pour les hidden services (############### This section is just for location-hidden services ###) et on modifie comme suit (=on décommente la ligne HiddenServiceDir /var/lib/tor/other_hidden_service/ et HiddenServicePort 22 127.0.0.1:22):

```
############### This section is just for location-hidden services ###

## Once you have configured a hidden service, you can look at the
## contents of the file ".../hidden_service/hostname" for the address
## to tell people.
##
## HiddenServicePort x y:z says to redirect requests on port x to the
## address y:z.

#HiddenServiceDir /var/lib/tor/hidden_service/
#HiddenServicePort 80 127.0.0.1:80

HiddenServiceDir /var/lib/tor/other_hidden_service/
#HiddenServicePort 80 127.0.0.1:80
HiddenServicePort 22 127.0.0.1:22
```

Note: on peut indiquer toute autre valeur à la place de “other_hidden_service”. C'est simplement le nom du répertoire, j'ai ici pris celui par défaut.

Une fois cela fait, on redémarre le service Tor:

```
sudo service tor restart
```

Le hidden service nouvellement créé se trouve ici, comme indiqué dans le fichier torrc, dans le répertoire /var/lib/tor/other_hidden_service/hostname. On récupère son adresse .onion:

```
serverssh@server-ssh:~$ sudo cat /var/lib/tor/other_hidden_service/hostname
wn2na4obqlp73cwk.onion
```

On note bien cette adresse car c'est celle qu'on utilisera pour se connecter à cette machine depuis le client.

Deuxième partie: installation de Tor sur le client et configuration de ssh

On commence par installer Tor et ssh **sur le client**:

```
sudo apt install tor ssh
```

Note: vérifiez que l'installation comprend les paquets openssh-client et openssh-server, le métapaquet ssh les installe les deux sur debian 9, sur une autre distribution il est possible qu'il faille les préciser séparément

On installe également nmap, dont l'outil ncat servira à connecter notre client ssh au serveur en .onion:

```
sudo apt install nmap
```

C'est tout pour ce qui est de l'installation, le reste se gère dans la partie “Utilisation” ci-dessous.

Optionnel: d'avantage de sécurité avec un hidden service “stealth”

Avec la configuration actuelle, n'importe qui peut tenter de se connecter à notre serveur SSH s'il en connaît l'adresse .onion (hautement improbable, celle-ci étant générée cryptographiquement et localement). Bien sûr, la connaissance du mot de passe (ou de la clé) reste nécessaire pour pouvoir se connecter. Si vous voulez néanmoins être sûr d'être le seul à pouvoir “parler” à votre serveur SSH, vous pouvez ajouter une option “stealth” à votre hidden service sur le serveur. De cette manière, seuls les clients étant également en possession d'une clé spécifique pourront initier une connexion à votre serveur SSH. Toutes les autres tentatives aboutiront à un timeout.

1. configuration du hidden service “stealth” sur le serveur ssh

Comme pour la création d'un hidden service normal, on se rend dans le fichier de configuration de tor:

```
sudo nano /etc/tor/torrc
```

Et on cherche la section pour les hidden services (############### This section is just for location-hidden services ###) qu'on modifie comme précédemment, mais avec une ligne en plus (HiddenServiceAuthorizeClient stealth):

```
############### This section is just for location-hidden services ###

## Once you have configured a hidden service, you can look at the
## contents of the file ".../hidden_service/hostname" for the address
## to tell people.
##
## HiddenServicePort x y:z says to redirect requests on port x to the
## address y:z.

#HiddenServiceDir /var/lib/tor/hidden_service/
#HiddenServicePort 80 127.0.0.1:80

HiddenServiceDir /var/lib/tor/other_hidden_service/
#HiddenServicePort 80 127.0.0.1:80
HiddenServicePort 22 127.0.0.1:22

HiddenServiceDir /var/lib/tor/hidden_ssh_stealth/
HiddenServicePort 22 127.0.0.1:22
HiddenServiceAuthorizeClient stealth root
```

Remarque: je crée ici un deuxième service ssh, que j'appelle “hidden_ssh_stealth”

Comme précédemment, on récupère l'adresse .onion du service:

```
sudo service tor restart
sudo cat /var/lib/tor/hidden_ssh_stealth/hostname
```

Comme on a ajouté l'option stealth, on a cette fois une courte clé en plus:

```
erzi6casesvznend.onion rXc/eL2wK2pTNbYcdGjulh  # client: root
```

On retient l'ensemble de ces informations dont on aura besoin pour se connecter au serveur depuis notre client.

2. configuration du fichier /etc/tor/torrc sur le client

Sur le client, éditer le fichier /etc/tor/torrc:

```
sudo nano /etc/tor/torrc
```

Aller tout à la fin du fichier et ajouter ces lignes:

```
HidServAuth erzi6casesvznend.onion rXc/eL2wK2pTNbYcdGjulh # client: root
```

Bien entendu, remplacez ces valeurs par les vôtres!

Puis redémarrer le service tor:

```
sudo service tor restart
```

Voilà, c'est tout! Pour les options SSH à utiliser pour se connecter au service, se référer au chapitre “Installation” ci-dessous.

Optionnel: utiliser un hidden service V3 plutôt que V2

Si vous avez une version de Tor récente (0.3.2.1 minimum), vous pouvez également profiter de la nouvelle génération de hidden services, appelée V3.

Sur debian stable, la version actuelle (avril 2018) est 0.2.9.14, il faudra donc installer le paquet tor depuis les backports pour pouvoir utiliser un onion en v3 (sur le serveur comme sur le client):

ajouter la source backports dans /etc/apt/sources.list, puis:

```
sudo apt update
sudo apt -t stretch-backports install tor
```

Sur le serveur, le fichier /etc/tor/torrc s'édite comme suit pour générer un hidden service en V3:

```
HiddenServiceDir /var/lib/tor/sshv3/
HiddenServiceVersion 3
HiddenServicePort 22 127.0.0.1:22
```

Comme on peut le constater, l'adresse obtenue est beaucoup plus longue:

```
tyo4x3tmv6dvsqdg2fxr7koqolil4iujpg6kvoig3ybd6unryo5ferad.onion
```

Pour le reste, rien ne change.

Utilisation

Paramétrage de SSH sur le client

Sur le client, si vous tentez une connexion directement à l'adresse onion, vous recevrez immanquablement un message d'erreur:

```
clientssh@client-ssh:~$ ssh serverssh@wn2na4obqlp73cwk.onion
ssh: Could not resolve hostname wn2na4obqlp73cwk.onion: Name or service not
known
```

En effet, il faut utiliser des options spécifiques pour indiquer au client SSH comment se connecter à cette adresse. Essayez cette fois-ci avec les arguments suivants:

```
ssh -o VerifyHostKeyDNS=no -o CheckHostIP=no -o IdentitiesOnly=yes -o
ProxyCommand="ncat --proxy 127.0.0.1:9050 --proxy-type socks5 %h %p"
serverssh@wn2na4obqlp73cwk.onion
```

Vous devriez alors pouvoir vous connecter à votre serveur en attente, tapi dans les tréfonds de Tor:

```
The authenticity of host 'wn2na4obqlp73cwk.onion (<no hostip for proxy
command>)' can't be established.
ECDSA key fingerprint is SHA256:IJfKWAyoQzCTEUXurgIcwfmQaT0A2fxCQnX0y+X3mvU.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'wn2na4obqlp73cwk.onion' (ECDSA) to the list of
known hosts.
serverssh@wn2na4obqlp73cwk.onion's password:
Linux server-ssh 4.9.0-4-amd64 #1 SMP Debian 4.9.51-1 (2017-09-28) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Apr  4 08:38:10 2018
```

Explication des arguments de la commande:

- -o ProxyCommand)“ncat –proxy 127.0.0.1:9050 –proxy-type socks5 %h %p”

En réalité, seule cette option est strictement nécessaire. Elle indique à SSH d'utiliser ncat pour ouvrir un proxy qui fera router la connexion SSH, ici Tor.

Les trois commandes suivantes ne sont pas nécessaires mais ajoutent un degré de sécurité:

- -o VerifyHostKeyDNS=no

En indiquant non à cette option, on dit à SSH de ne pas faire une requête DNS pour vérifier la clé du

host.

- -o CheckHostIP=no

En indiquant non à cette option, on dit à SSH de ne pas faire une requête DNS pour résoudre le hostname du serveur (ici ynlna4obqlp73ewk.onion).

- -o IdentitiesOnly=yes

En indiquant oui à cette option, on dit à SSH de ne pas essayer toutes les clés dont il a connaissance au moment de la connexion, mais seulement celle indiquée dans IdentityFile.

Automatisation de la commande

Comme on l'imagine aisément, devoir écrire tous ces arguments à chaque connexion est très fastidieux. Il est heureusement possible de les indiquer dans un fichier que SSH va charger à chaque nouvelle connexion à notre serveur sur Tor. Pour ce faire, modifier ou créer le fichier ~/.ssh/config:

```
nano ~/.ssh/config
```

si le dossier .ssh n'existe pas, le créer

L'éditer comme suit:

```
Host Onion
Hostname wn2na4obqlp73cwk.onion
User serverssh
proxyCommand ncat --proxy 127.0.0.1:9050 --proxy-type socks5 %h %p
VerifyHostKeyDNS no
CheckHostIP no
IdentitiesOnly yes
```

Note: vous pouvez remplacer “Onion” par ce que bon vous semble. “User” doit par contre correspondre au compte d'utilisateur sur le serveur auquel vous vous connectez, dans mon exemple “serverssh”.

Vous pouvez maintenant vous connecter à votre serveur en tapant simplement ssh Onion. Les options seront automatiquement chargées:

```
clientssh@client-ssh:~$ ssh Onion
serverssh@wn2na4obqlp73cwk.onion's password:
Linux server-ssh 4.9.0-4-amd64 #1 SMP Debian 4.9.51-1 (2017-09-28) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Apr  4 12:40:45 2018 from 127.0.0.1
```

Astuce: pour faciliter le debbuging en cas d'erreur, lancer ssh avec l'option de verbosité maximum:

ssh -vvv Onion

1)

N'hésitez pas à y faire part de vos remarques, succès, améliorations ou échecs !