Ma configuration WireGuard (client et serveur) avec IPv6 NATé

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 25 mars 2026

https://www.bortzmeyer.org/wireguard.html

Si vous connaissez déjà, même superficiellement, WireGuard, vous n'apprendrez rien dans cet article, je n'ai rien fait d'original avec WireGuard mais, comme je l'ai récemment utilisé intensivement, voici mon expérience et ma configuration.

WireGuard est un logiciel permettant de créer un VPN, ce qui permet plein de choses, comme d'échapper à certains types de censure, ou comme dissimuler vos activités au fournisseur d'accès Internet que vous utilisez. Par exemple, en ce moment, j'écris cet article depuis le Wifi gratuit, non sécurisé, d'un aéroport dans un pays étranger, donc le risque est important. Bien sûr, je ne fais que des communications chiffrées de bout en bout (SSH, HTTPS, etc) mais le FAI voit quand même les machines avec lesquelles je communique. Autant lui dissimuler un peu plus de choses.

Il y a plusieurs façons de faire du VPN. Déjà, il faut choisir qui ou quoi va être à l'autre extrémité du réseau virtuel ainsi créé. Dans un cadre professionnel, on va se connecter au VPN de son employeur. Il y a aussi des fournisseurs de VPN commerciaux (comme celui qui est cité dans la moitié des vidéos YouTube parce que les youtubeur·ses ont des factures à payer). Et on peut aussi être son propre fournisseur de VPN, via une machine qu'on loue quelque part. Du point de vue de la sécurité, attention, le VPN dissimule votre activité à votre FAI mais le fournisseur de VPN devient votre vrai FAI et lui voit tout le trafic. Le VPN ne dispense donc pas de faire du chiffrement de bout en bout, et il faut bien choisir son fournisseur de VPN, sans se fier à l'influenceur payé pour cela qui va vanter son sponsor dans sa vidéo, avec des arguments malhonnêtes (« personne ne pourra voir ce que vous visitez »).

Moi, j'ai choisi d'être mon propre fournisseur de VPN et j'ai loué pour cela (c'était un projet temporaire) une machine chez V.PS <https://v.ps/>. J'ai choisi cet hébergeur car la même entreprise <https://xtom.com/> a un résolveur DNS <https://www.bortzmeyer.org/resolveur-dns.html> sympa <https://www.bortzmeyer.org/dns-sb.html>. Et surtout ils louent des VPS dans de nombreux pays, ce qui m'a permis de placer le serveur VPN pas trop loin du client VPN (pour éviter que les paquets ne fassent le tour du monde), mais quand même dans un autre pays. J'ai choisi Debian comme système d'exploitation car ça juste marche.

Une fois ce choix fait, quel logiciel choisir ? Il existe de nombreuses techniques pour faire un VPN, IPsec (RFC 4301 [1]), OpenVPN, WireGuard…Les mises en œuvre d'IPsec sont complexes, et j'ai pris WireGuard car j'en avais entendu dire du bien (et il parait qu'il est plus rapide qu'OpenVPN mais je n'ai pas mesuré).

Effectivement, la configuration est simplissime. Côté serveur, on installe WireGuard (il y a un paquetage Debian), on vérifie qu'il est bien là :

```
server% wg version
wireguard-tools v1.0.20210914 - https://git.zx2c4.com/wireguard-tools/
```

WireGuard protège toute la communication avec de la cryptographie asymétrique. On va donc générer des clés, d'abord la clé privée (que je ne vous montre pas en entier) :

```
server% wg genkey
E....
```

(En vrai, on va sans doute rediriger vers un fichier, car il faut garder cette clé. Pensez à donner à ce fichier des permissions bien strictes.)

Il reste à calculer la clé publique à partir de la privée (ici, la privée a été mise dans `private.key`). Elle, on peut la montrer :

```
server% wg pubkey < private.key
t0Bjw+XGBE1PI81vmhx+bBEIeoWO1f6JWthwi1vYHBk=
```

Ensuite, le gérant du serveur de VPN (moi, en l'occurrence) va devoir choisir un plan d'adressage. Ici, avec seulement un client, je ne me suis pas embêté. Comme V.PS ne me donne qu'une seule adresse IPv4 publique et, hélas, une seule adresse IPv6 publique, on va donner des adresses privées au client VPN, et faire du NAT. Le VPN aura donc des adresses dans `10.66.0.0/24` (RFC 1918) et `fd00:666:ab::1/64` (ce sont les ULA - "*Universal Local Address*" du RFC 4193, je détaille plus loin les choix pour IPv6). Le serveur aura une adresse se terminant par 1 et le client par 2.

WireGuard fonctionne au-dessus d'UDP, pour maximimiser ses chances de passer dans un réseau mal fichu qui bloquerait d'autres protocoles de couche 4 (le problème de l'ossification). J'ai choisi un port UDP inhabituel, 41834 car les VPN sont souvent bloqués par les FAI, surtout dans certains pays, un port inhabituel a un petit peu plus de chances d'avoir été oublié par le censeur. Voici la configuration du serveur (clé privée masquée) :

```
server% cat /etc/wireguard/wg0.conf
[Interface]
Address = 10.66.0.1/24,fd00:666:ab::1/64
ListenPort = 41834
PrivateKey = E...Y=

[Peer]
PublicKey = xwVsq/MqwhqUxCRk5KeU0h8nLVCmBRPV3PFgRoay3VA=
AllowedIPs = 10.66.0.2/32,fd00:666:ab::2/128

# Ajouter d'autres pairs si vous avez d'autres clients.
```

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc4301.txt`

(Pourquoi `wg0` ? C'est le nom de l'interface réseau couramment utilisé mais vous pouvez en mettre un autre.) Le premier groupe indique les préfixes IP du VPN, les adresses du serveur et son port. Le deuxième indique les caractéristiques du client, dont sa clé publique, qu'il nous a communiqué.

Car il faut faire pareil chez le client (Ubuntu, dans mon cas). Il a aussi une clé privée (qu'il garde pour lui), une clé publique (qu'il communique au gérant du serveur) et la clé publique du serveur, que le fournisseur de VPN lui a communiquée.

```
[Interface]
Address = 10.66.0.2/32,fd00:666:ab::2/128
PrivateKey = y...Q=

[Peer]
PublicKey = t0Bjw+XGBE1PI81vmhx+bBEIeoWO1f6JWthwi1vYHBk=
AllowedIPs = 0.0.0.0/0, ::/0
Endpoint = wg-server.bortzmeyer.org:41834
```

Vous noterez qu'on a indiqué le server par son nom mais on aurait aussi pu utiliser son adresse IP, ce qui peut être nécessaire si le résolveur DNS <https://www.bortzmeyer.org/resolveur-dns.html> par défaut est menteur. Sinon, il y a aussi dans le NetworkManager d'Ubuntu un cliquodrome pour configurer le client WireGuard mais je ne l'ai pas utilisé.

La directive `AllowedIPs` veut dire qu'on va utiliser le VPN pour toutes les destinations, IPv4 et IPv6. (Attention à ne pas mettre une directive pareille sur le serveur, ou vous déclencherez une boucle sans fin, empêchant tout accès par le réseau. Croyez-moi. J'ai essayé.)

Il reste à configurer le routage, et la traduction d'adresses puisque, aussi bien en IPv4 qu'en IPv6, on a des adresses IP privées, qui ne seront pas routées sur l'Internet. Déjà, activer le routage sur le serveur :

```
server% sysctl net.ipv4.ip_forward=1
server% sysctl   net.ipv6.conf.all.forwarding=1
```

Si vous voulez que le routage soit activé systématiquement, vous devez éditer `/etc/sysctl.conf` pour y mettre :

```
net.ipv4.ip_forward=1
net.ipv6.conf.all.forwarding=1
```

Notez que ces paramètres ne sont pas forcément appliqués au démarrage, pour des raisons complexes (façon polie de dire que c'est une bogue <https://bugs.launchpad.net/ubuntu/+source/procps/+bug/50093>). Si ce n'est pas le cas (vérifiez avec `sysctl -a | grep forward`), ajoutez `@reboot /usr/bin/sleep 5 && /usr/sbin/sysctl --system` dans la crontab de root.

Maintenant, la traduction d'adresses, en IPv4 (RFC 3424) et en IPv6 (notez que ce n'est pas du NAT66, RFC 6296, qui se ferait avec SNPT et pas SNAT, car NAT66 nécessite un préfixe routé) :

```
server% iptables -t nat -A POSTROUTING -s 10.66.0.0/24 -o eth0 -j SNAT --to-source 149.62.44.9
server% iptables -t nat -A POSTROUTING -s 10.66.0.0/24 -j MASQUERADE

server% ip6tables -t nat -A POSTROUTING -s fd00:666:ab::1/64 -o eth0 -j SNAT --to-source 2a12:a301:2010::10e
server% ip6tables -t nat -A POSTROUTING -s fd00:666:ab::1/64 -j MASQUERADE
```

Pour rendre ces règles systématiques à chaque démarrage, vous pouvez utiliser le paquetage iptables-persistent :

```
server% iptables-save -f /etc/iptables/rules.v4
server% ip6tables-save -f /etc/iptables/rules.v6
```

Une autre solution, pour ne pas retaper les commandes à chaque démarrage, est de les mettre dans des directives `PostUp` et `PostDown` de `wg0.conf`, qui sont exécutées lorsque le VPN est activé ou désactivé (je les ai simplifiées ici) :

```
PostUp = ip6tables -t nat -A POSTROUTING -o eth0 -j MASQUERADE; sysctl -q -w net.ipv6.conf.all.forwarding=1;
PostDown = ip6tables -t nat -D POSTROUTING -o eth0 -j MASQUERADE; sysctl -q -w net.ipv6.conf.all.forwarding=
```

Enfin, on peut démarrer le service, ici en utilisant systemd puisque c'est ce que prévoit le paquetage Debian :

```
server% systemctl start wg-quick@wg0

client% systemctl start wg-quick@wg0
```

(Si on veut que ce soit fait automatiquement au démarrage, on `systemctl enable wg-quick@wg0`.) À partir de là, un examen du trafic avec tcpdump va montrer qu'on utilise bien le VPN. Ici, je pingue `1.1.1.1` :

```
22:27:42.017959 IP 10.253.203.139.59473 > 149.62.44.9.41834: UDP, length 128
22:27:42.086272 IP 149.62.44.9.41834 > 10.253.203.139.59473: UDP, length 128
22:27:43.020249 IP 10.253.203.139.59473 > 149.62.44.9.41834: UDP, length 128
22:27:43.390246 IP 149.62.44.9.41834 > 10.253.203.139.59473: UDP, length 128
```

Et ici mon blog :

```
22:32:13.428890 IP 149.62.44.9.41834 > 10.253.203.139.59473: UDP, length 1452
22:32:13.428891 IP 149.62.44.9.41834 > 10.253.203.139.59473: UDP, length 1452
22:32:13.428891 IP 149.62.44.9.41834 > 10.253.203.139.59473: UDP, length 1452
22:32:13.429001 IP 10.253.203.139.59473 > 149.62.44.9.41834: UDP, length 96
22:32:13.429028 IP 10.253.203.139.59473 > 149.62.44.9.41834: UDP, length 96
22:32:13.564711 IP 149.62.44.9.41834 > 10.253.203.139.59473: UDP, length 80
22:32:13.564712 IP 149.62.44.9.41834 > 10.253.203.139.59473: UDP, length 80
```

Dans les deux cas, le FAI qui regarderait le trafic ne verrait pas `1.1.1.1` ou mon blog mais uniquement qu'il y a de la communication UDP chiffrée avec `149.62.44.9` (le serveur VPN). Notez aussi que l'utilisation du VPN permet au client de faire de l'IPv6 même si son réseau d'accès est bloqué au XXe siècle et ne connait qu'IPv4 :

```
client%  traceroute 2a09::
traceroute to 2a09:: (2a09::), 30 hops max, 80 byte packets
 1  2a12:a301:2010::1 (2a12:a301:2010::1)  0.923 ms  0.851 ms  0.842 ms
 2  * * *
 3  * * *
 4  * * *
 5  public-dns-a.dns.sb (2a09::)  0.148 ms  0.125 ms  0.133 ms
```

Voilà, configurer WireGuard prend moins de temps que lire entièrement cet article. La documentation que j'ai suivie était celle de Hetzner <https://community.hetzner.com/tutorials/install-and-configur

Si vous observez que ping passe bien mais que les connexions TCP (par exemple pour le Web, ou pour SSH) s'établissent, mais sans pouvoir transférer de données, c'est sans doute un problème de MTU (la taille compte <https://www.bortzmeyer.org/ping-taille-compte.html>). Dans ce cas, un `ip link set wg0 mtu 1200` va sans doute régler le prosblème, le temps d'investiguer plus sérieusement (vous pouvez aussi utiliser la directive `MTU` dans le `wg0.conf`, ou bien mettre en `PostUp iptables -t mangle -A POSTROUTING -p tcp --tcp-flags SYN,RST SYN -o eth0 -j TCPMSS --clamp-mss-to-pmtu`).

Ah, je n'ai pas encore parlé de la résolution DNS. Par défaut, le résolveur <https://www.bortzmeyer.org/resolveur-dns.html> n'est pas changé. S'il est distant, le trafic DNS passera par le VPN. S'il est local (ce qui est le cas par défaut avec systemd-resolved), il n'est pas affecté mais le trafic du résolveur local, quand il parle aux serveurs faisant autorité <https://www.bortzmeyer.org/serveur-dns-faisant-autorite.html> (ou bien à son "*forwarder*"), passe par le VPN. Mais si vous êtes dans un pays où il y a un résolveur menteur, votre résolveur local a pu être empoisonné par de fausses réponses avant que vous ne démarriez le VPN. Bref, la configuration idéale dépend de beaucoup de choses. Quelques notes techniques : si vous utilisez systemd-resolved, la directive `DNS` dans le `wg0.conf` du client va changer le "*forwarder*" utilisé. Vous pouvez aussi mettre :

```
PostUp = resolvectl dns %i 2a09::0
```

Et ça donnera le même résultat. Le résolveur (celui indiqué dans `/etc/resolv.conf`) ne change pas mais il transmettra ("*forward*") désormais toutes les requêtes à `2a09::`. Trois points à retenir :

— Le résolveur utilisé ici, `2a09::` est celui de xTom <https://www.bortzmeyer.org/dns-sb.html> puisqu'il est sur le même réseau que le serveur VPN,
— La forme canonique de cette adresse IP est `2a09::`, pas `2a09::0` (RFC 5952), mais il y a une stupide bogue dans systemd qui affiche "*resolvectl[19427] : Failed to parse DNS server address : 2a09 :*" "*resolvectl[19427] : Failed to set DNS configuration : Invalid argument*",
— Rappelez-vous que le VPN configuré route IPv6 donc que ce résolveur DNS est joignable même si votre réseau d'accès local n'a pas IPv6.

Revenons un peu à l'adressage et au routage IPv6. Tout dépend de ce qu'offre l'hébergeur du serveur VPN. S'il vous délègue un préfixe (sans doute un /64), et vous route tous les paquets vers ce préfixe, vous avez juste à activer le routage (et peut-être à traduire le préfixe, si vous utilisez des adresses privées en interne, cf. RFC 6296), c'est la situation idéale (mais extrêmement rare sur un hébergement bon marché). Si vous avez un préfixe mais qu'il n'y a pas de route chez l'hébergeur vers votre machine, vous devrez le configurer en "*proxy*" NDP (je vous laisse faire l'exercice). Et si vous n'avez pas un préfixe, juste une adresse IP, ce qui était mon cas, il fallait utiliser les ULA. Les ULA sont des adresses privées, comme on peut le vérifier en demandant au RIR :

```
client% whois -h whois.ripe.net fd00:666:ab::1
...
inet6num:       fc00::/7
netname:        IANA-BLK
descr:          Unique Local Addresses (ULAs)
country:        EU # Country is really world wide
remarks:        This network should never be routed outside an enterprise
remarks:        See RFC4193 for further information
```

Amusez-vous bien avec le VPN et n'en profitez pas pour regarder des sites Web illégaux.