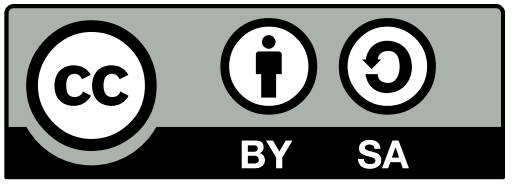
WireGuard®

FAST, MODERN, SECURE VPN TUNNEL

Atelier CLUSIR Tahiti
Jeudi 13 décembre 2018

Jean-Denis Girard
SysNux
https://www.SysNux.pf/

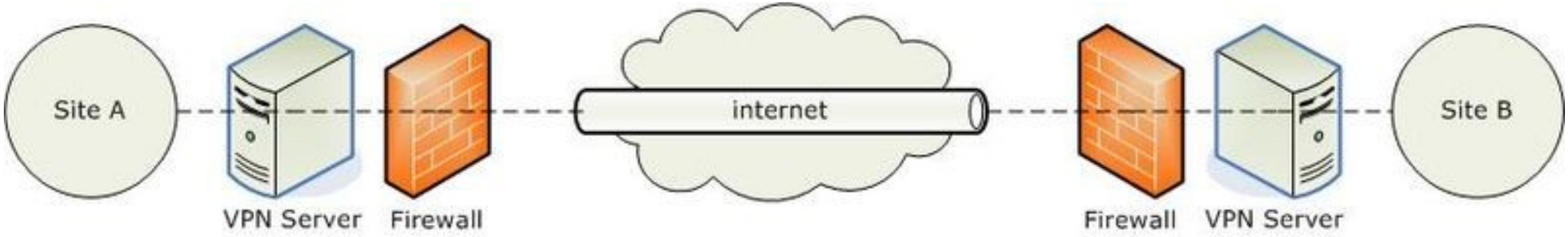


Plan

- **Rappels VPN**
- **Wireguard, les principes**
- **Wireguard, la pratique**

Quoi ?

- Réseau Privé Virtuel utilise l'infrastructure publique (internet) pour établir un lien réseau entre des entités privées.



Source : Wikipédia

- Le VPN assure l'authentification, l'intégrité, et, presque toujours, le chiffrement des paquets.
- Avantages : faible coût, disponibilité.
- Inconvénients : pas de garantie de débit, pas de QOS

Existant

De nombreuses technologies existent :

- GRE : Cisco rfc1701 (1990) et rfc2784 (2000)
- PPTP : Microsoft rfc2637 (1999)
- L2TP : rfc3931 (2005, 2009)
- MPLS : rfc3031 (2001, 2011, 2012), opérateurs (ProLan)
- SSH : rfc4251 (2006, 2010)
- **SSL / TLS** : OpenVPN, Cisco AnyConnect (openconnect), 1999 à aujourd'hui
- **IPsec** : rfc1825 (1995)

Problèmes

- Protocoles anciens, qui permettent à l'utilisateur de choisir des méthodes d'authentification / chiffrement aujourd'hui considérées comme non sûres.
- Difficiles à utiliser, incompatibilités.
- Certains rencontrent des problèmes avec NAT / PAT.
- Taille du code :
  - OpenVPN ~120 000 lignes de code + OpenSSL
  - IPsec (StrongSwan) : ~120 000 lignes de code kernel + ~400 000 lignes de code utilisateur
  - SoftEther : ~330 000 lignes de code

=> Difficile à auditer / prouver

=> CVES en 2018 : **10 sur IPsec, 15 sur OpenVPN**

WireGuard

- Conçu et développé par **Jason Donenfeld**, aussi connu comme zx2c4.
- Chercheur en sécurité et développeur, expérience en découverte et exploitation de failles dans le chiffrement, développement pour le noyau Linux.
- Souhaitait créer un VPN qui évite les problèmes de cryptographie ou d'implémentation qu'il avait découverts dans certains projets.

WireGuard

- VPN de niveau 3 (uniquement) pour IPv4 et IPv6
- Basé sur UDP, un seul flux facilite le franchissement des pare-feu
- Cryptographie moderne
- Authentification basée sur couple de clés (similaire SSH)
- Pas d'option, meilleurs choix effectués pour vous
- Intègre l'itinérance.
- Adapté aux conteneurs.

WireGuard

- Conçu pour le noyau Linux (très rapide), mais dispose d'une implémentation multi-plateformes (moins rapide) : Android, IOS, FreeBSD, OpenBSD, macOS, OpenWRT... Version officielle Windows bientôt disponible, mais version alternative existe (https://tunsafe.com/).
- Basé sur UDP, un seul flux facilite le franchissement des pare-feu
- Insiste sur la simplicité et l'audit
- Issu de l'expérience de Jason en tests de pénétrations
- Logiciel libre (GPLv2)

WireGuard

Moins de 4 000 lignes de code pour le VPN lui-même !

Moins, c'est mieux !

Pas intégré au noyau Linux pour l'instant, première soumission en août 2018

> « Can I just once again state my love for it and hope it gets merged soon? Maybe the code isn't perfect, but I've skimmed it, and compared to the horrors that are OpenVPN and IPSec, it's a work of art. » **Linus Torvalds**

> « It's been great to see Wireguard mature over the years into something that works really well. This past week it survived the LinuxCon/ELCE/Kernel Summit traffic thanks to Konstantin Ryabitsev and packet.net setting up a server for us to use. » **Greg Kroah-Hartman**

Zinc

Mais ~20 000 lignes de code dans une nouvelle bibliothèque de chiffrement : **Z**inc as **IN C**rypto. L'approche de Zinc se veut plus pragmatique, en définissant de simples fonctions, faciles à utiliser : moins de risque d'erreur.

Différentes objections remontées par les mainteneurs du sous-système de cryptographie du noyau Linux :

- patch trop monolithique,
- pourquoi ne pas utiliser l'existant,
- pas dans l'arborescence crypto,
- maintenance du code (Jason + Samuel Neves),
- code assembleur auto-généré difficile à lire.

Zinc

L'implémentation des fonctions est choisie par ordre de préférence :

- prouvée,
- largement utilisée (notamment dans OpenSSL avec le code d'Andy Polyakov, qui est aussi le plus rapide pour différentes architectures),
- issue de la référence.

Jason travaille avec une équipe de l'INRIA : il utilise des outils (HACL* et F*) qui permettent de vérifier les algorithmes et le code.

Jason a répondu aux critiques en produisant 7 révisions du code (v8) : il espère que la version 9 sera acceptée.

État de l’art en cryptographie

Jason a choisi pour WireGuard le meilleur :

- ChaCha20 pour le chiffrement symétrique, authentifié avec Poly1305, en utilisant la construction AEAD (RFC7539)
- Curve25519 for ECDH
- BLAKE2s pour les fonctions de hachage (RFC7693)
- SipHash24 pour les clés des tables de hachage
- HKDF comme fonction de dérivation de clé (RFC5869)

Ne permet pas à l'utilisateur d'effectuer des choix **catastrophiques** !

Implémenté avec les instructions les plus rapides.

Interface

WireGuard se présente comme une simple **interface** :

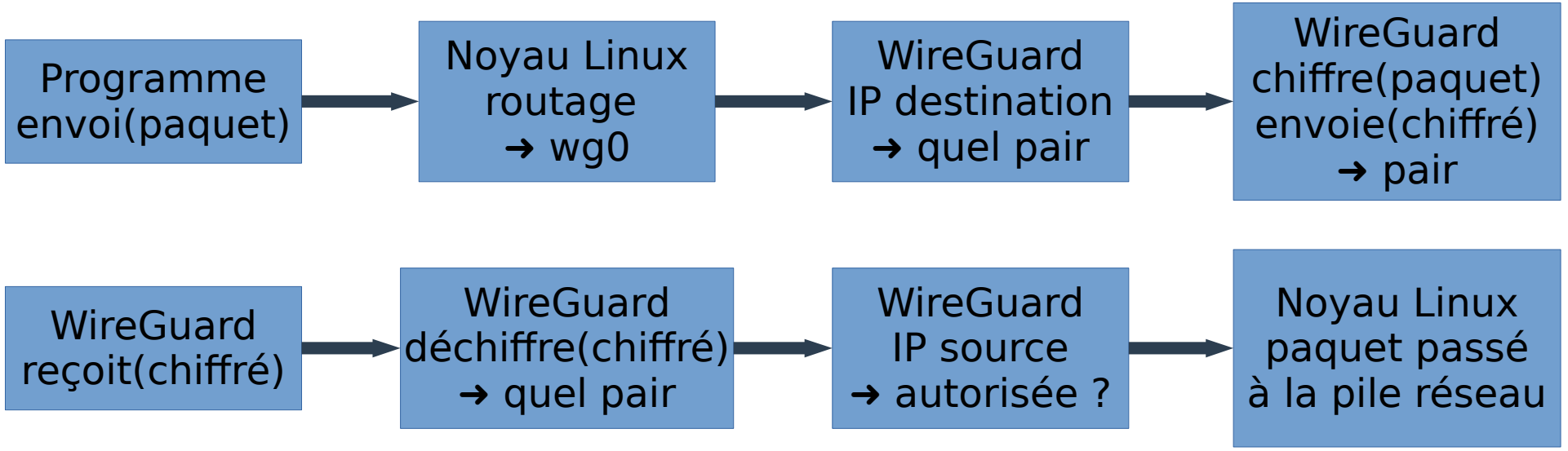
```
ip link add wg0 type wireguard
```

Tous les outils Linux normaux peuvent être utilisés : `ip ifconfig route iptables hosts.allow hosts.deny bind()`.

Après avoir ajouté l'interface, il faut configurer ses pairs (clés et adresses), et elle est immédiatement utilisable.

Routage par clé publique

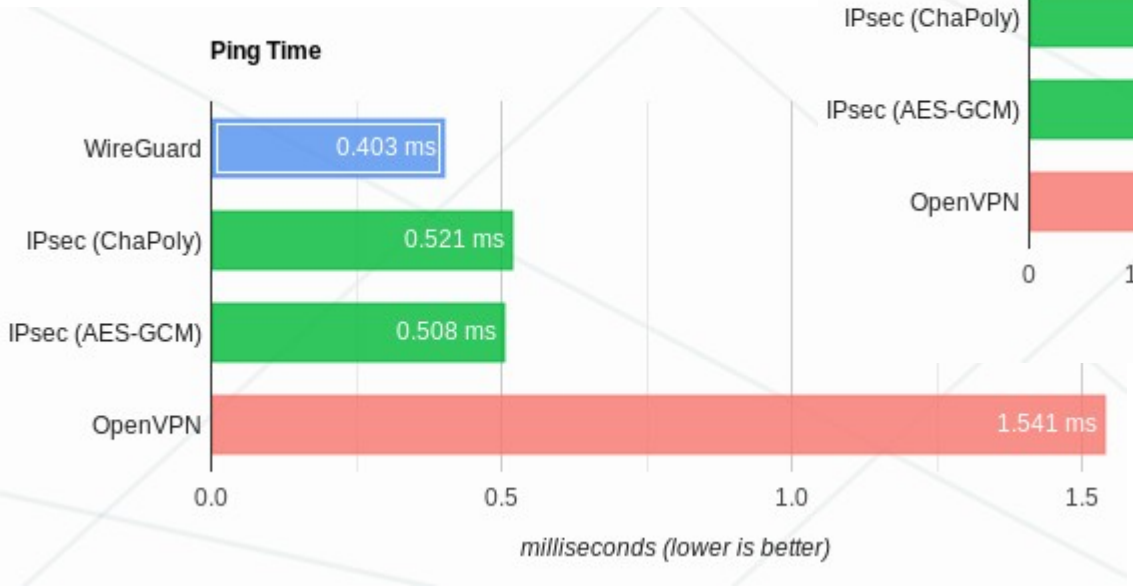
WireGuard fonctionne par association de clés avec des adresses IP.

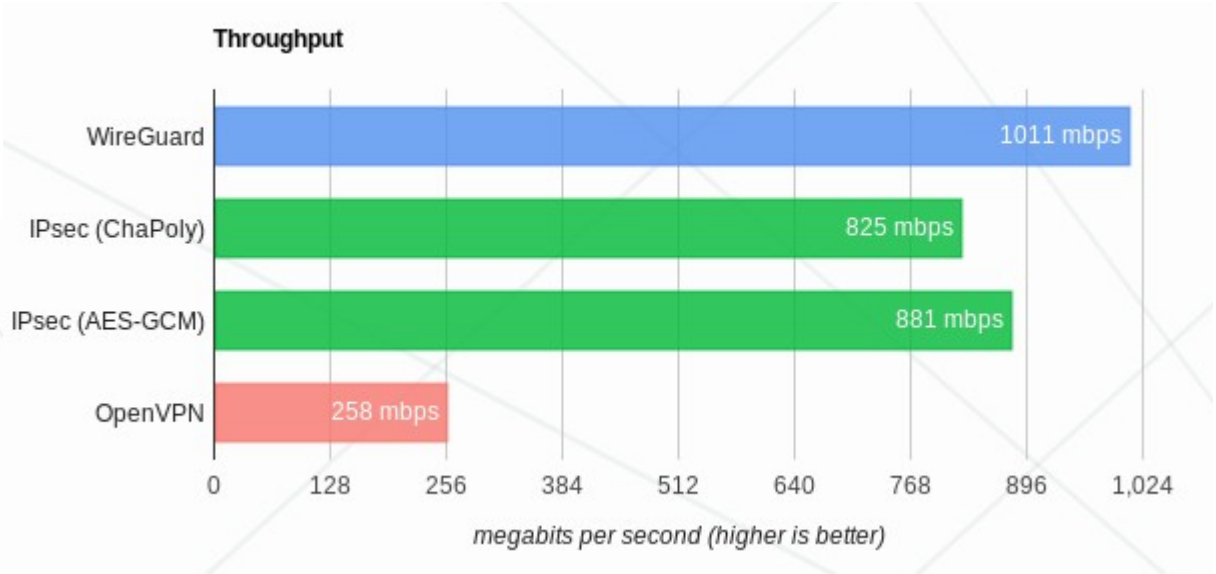


Si un paquet provient de l'interface wg0 avec une adresse correspondant à la clé, il est authentifié et accepté.

Performances

Grâce à son fonctionnement dans le noyau (sous Linux et éventuellement Android), WireGuard offre d'excellentes performances.





Performances

Tests effectués sur Rock64 :

Client direct

```
[ ID] Interval           Transfer     Bandwidth       Retr
[  5]   0.00-273.93 sec  3.00 GBytes  94.1 Mbits/sec    1             sender
[  5]   0.00-273.93 sec  3.00 GBytes  94.0 Mbits/sec                  receiver
```

Client SSH

```
[ ID] Interval           Transfer     Bandwidth       Retr
[  5]   0.00-586.45 sec  3.00 GBytes  43.9 Mbits/sec    0             sender
[  5]   0.00-586.45 sec  3.00 GBytes  43.9 Mbits/sec                  receiver
CPU Utilization: local/sender 0.8% (0.0%u/0.8%s), remote/receiver 2.6% (0.4%u/2.3%s)
```

Client IPsec

```
[ ID] Interval           Transfer     Bandwidth       Retr
[  5]   0.00-546.64 sec  3.00 GBytes  47.2 Mbits/sec  1123             sender
[  5]   0.00-546.64 sec  3.00 GBytes  47.2 Mbits/sec                  receiver
CPU Utilization: local/sender 86.8% (0.0%u/86.8%s), remote/receiver 29.8%
(0.3%u/29.5%s)
```

Client WireGuard

```
[ ID] Interval           Transfer     Bandwidth       Retr
[  5]   0.00-285.73 sec  3.00 GBytes  90.2 Mbits/sec  514             sender
[  5]   0.00-285.73 sec  3.00 GBytes  90.1 Mbits/sec                  receiver
CPU Utilization: local/sender 1.8% (0.0%u/1.8%s), remote/receiver 6.7% (0.9%u/5.8%s)
```

Configuration

Chaque partie associe son interface à sa clé privée et un port ; les pairs sont définis par leurs clés publiques. Les clients doivent en plus indiquer l'adresse / port du serveur.

Serveur :

```
[Interface]
PrivateKey =
ListenPort =

[Peer]
PublicKey =
AllowedIPs =

[Peer]
PublicKey =
AllowedIPs =
```

Client :

```
[Interface]
PrivateKey =
ListenPort =

[Peer]
PublicKey =
EndPoint = adr_serveur:port
AllowedIPs =
```

Utilisation

L'outil `wg` (`man wg`) permet de générer les clés et de configurer les interfaces WireGuard.

Clé privée :

```
wg genkey > cle_privee
```

Clé publique associée :

```
wg pubkey < cle_privee > cle_publique
```

Ou encore :

```
wg genkey | tee cle_privee | wg pubkey > cle_publique
```

WireGuard ne se charge pas de l'échange des clés publiques : copie, email, scp, ...

Utilisation

Ensuite, on réalise l'association entre la clé privée et l'interface, et on définit ses pairs :

```
wg set wg0 listen-port 51820 private-key cle_publique
peer ABCDEF... allowed-ips 192.168.88.0/24 endpoint
209.202.254.14:8172
```

Ou par un fichier de configuration :

```
wg setconf wg0 wg0.conf
```

L'état global peut être visualisé via :

```
wg show
```

Utilisation

Le script `wg-quick` (`man wg-quick`) permet de simplifier la configuration en regroupant tous les paramètres dans un seul fichier, recherché (par défaut) dans `/etc/wireguard/`.

Les scripts de démarrage (`systemd` ou autre) peuvent s'appuyer sur `wg-quick` pour démarrer le VPN comme un service.

WireGuard commence aussi à être intégré à des outils de configuration réseau de plus haut niveau (ex. `NetworkManager`).

Installation

Comme indiqué précédemment, WireGuard n'est pas encore intégré au noyau Linux.

Pour faciliter l'installation, Jason met à disposition des dépôts pour les principales distributions Linux, et macOS : module noyau, et outils.

L'application Android est disponible sur F-Droid et Play Store.

L'application iOS est dans App Store's TestFlight.

Démos !

Création tunnel simple entre deux machines Linux : comparaison performances WireGuard OpenVPN.

Accès à des ressources d’un réseau local.

Routage protocole difficile : SIP.

Accès VNC à serveur Windows

```
x220 -- routeur - (internet) - routeur - tiare -> win7 (VNC)
```

Roaming : accès Internet via ViniBox ou 3G.

Conteneurs

Grâce à son support des espaces nom réseau (netns) WireGuard peut être utilisé comme interface unique d'un conteneur (Docker par exemple).

Démo !

Fournisseurs

WireGuard commence à être supporté par quelques fournisseurs :

TunSafe https://tunsafe.com/vpn

Azire VPN https://www.azirevpn.com/wireguard

Mullvad https://www.mullvad.net/fr/servers/#wireguard

HiddenRouter https://hiddenrouter.com/

Démo !

Remerciements, références

Éric Mognat, liste WireGuard, CLUSIR Tahiti.

Site web https://www.wireguard.com/

Ars Technica
https://arstechnica.com/gadgets/2018/08/wireguard-vpn-review-fast-connections-amaze-but-windows-support-needs-to-happen/

The Linux-Kernel Archive

http://lkml.iu.edu/hypermail/linux/kernel/