Wireguard 実践入門

Kazuhiro NISHIYAMA

東京エリア・関西合同Debian勉強会

2020/04/18

自己紹介

- 西山 和広
- Ruby のコミッター
- twitter, github など: @znz

WireGuard とは?

- WireGuard: fast, modern, secure VPN tunnel
- 暗号化などは最近のアルゴリズムを採用
  - ChaCha20, Curve25519, BLAKE2, SipHash24, HKDF
- アルゴリズムの選択で悩む必要がない
- 別実装も存在

インストール

- https://www.wireguard.com/install/
- Debian (buster) なら backports から
- Ubuntu (19.04以前) なら ppa:wireguard/wireguard から
- その他ディストリビューションにも対応
- Windows, macOS, Android, iOS にも対応

カーネルモジュール

- Linux 5.6 以降ならカーネルが直接サポート (wireguard-dkms 不要)
- その関連で wireguard-dkms の postinst が自動リロードから再起動要求に https://salsa.debian.org/debian/wireguard-linux-compat/-/commit/af9f90b13118cd259227773c2a81ccfa25cf3e5d

ネットワークのイメージ

- WireGuard のトンネル = スイッチングハブ
- Peer の AllowedIPs がルーティング
- 例: 10.192.122.0/24 のネットワークをスター型やメッシュ型で接続
- 例: 10.192.124.1/32 と 10.192.124.2/32 を P2P 接続

基本的な使い方

- wg コマンド : 鍵の設定など
- wg-quick コマンド : IP アドレスの設定などを含むラッパーコマンド

ip コマンドで直接設定する場合は https://www.wireguard.com/quickstart/ 参照

systemd with wg-quick

- この資料では `wg-quick@.service` を使うのを想定
  - 他の設定方法は https://wiki.debian.org/WireGuard 参照
- 例: `/etc/wireguard/wg0.conf` が `wg-quick@wg0.service` に対応

設定ファイル

- `/etc/wireguard/wg0.conf` の `wg0` が interface 名になる
  - `systemctl enable --now wg-quick@wg0` で開始
- 名前は何でも良さそう
- `wg0`, `wg1`, ... のように `wg数字` が一般的

man にある設定ファイル例

```
[Interface]
PrivateKey = yAnz5TF+lXXJte14tji3zlMNq+hd2rYUIgJBgB3fBmk=
ListenPort = 51820

[Peer]
PublicKey = xTIBA5rboUvnH4htodjb6e697QjLERt1NAB4mZqp8Dg=
Endpoint = 192.95.5.67:1234
AllowedIPs = 10.192.122.3/32, 10.192.124.1/24

[Peer]
PublicKey = TrMvSoP4jYQlY6RIzBgbssQqY3vxI2Pi+y71lOWWXX0=
Endpoint = [2607:5300:60:6b0::c05f:543]:2468
AllowedIPs = 10.192.122.4/32, 192.168.0.0/16

[Peer]
PublicKey = gN65BkIKy1eCE9pP1wdc8ROUtkHLF2PfAqYdyYBz6EA=
Endpoint = test.wireguard.com:18981
AllowedIPs = 10.10.10.230/32
```

設定説明

- 次のページからのタイトルの wg(8)
  Interface = wg(8) コマンドで使われる
  Interface セクションの設定
- wg(8) と wg-quick(8) の区別は設定する時にはあまり気にしない
- man を調べる時に影響する程度

wg(8) Interface

- PrivateKey: 自分側の秘密鍵
  - 設定ファイルに直書きするなら権限に注意
- ListenPort: 待ち受けポート
  - ノート PC 側などの固定しない場合は不要
  - interface ごとに別ポートにする必要あり
- FwMark (iptables などと組み合わせるためのもの)

wg(8) Peer

- PublicKey : ID を兼ねていて必須
  - 接続相手の公開鍵
- PresharedKey : なくても良い
  - OpenVPN の ta.key のように追加の共有鍵認証
- AllowedIPs
  - ’,’ 区切りの IP/CIDR
  - ’,’ で並べる代わりに複数並べても OK

wg(8) Peer

- Endpoint : 接続先
  - IP アドレスまたはホスト名 + ‘:’ + ポート番号
  - IPv6 なら `Endpoint = [XXXX:YYYY:ZZZZ::1]:51820` のように [] でくくる
- PersistentKeepAlive : NAT 環境で設定
  - keep alive パケット送信間隔 (秒)
  - 0 や “off” で無効 (デフォルト)

wg-quick(8)

- wg コマンドと ip コマンドのラッパー
- 次ページの設定項目は ip コマンドなどの wg コマンド以外での設定で使われる

wg-quick(8) Interface

- SaveConfig=true：設定の自動保存
  - `wg set` で peer 追加などするなら便利
- Address：自分側 IP アドレス (複数可能)
- DNS：up した時に設定する DNS サーバー
- PreUp, PostUp, PreDown, PostDown
  - iptables の NAT 設定などに使う
- MTU, Table

privatekey 自動読み込み

- PostUp で wg set %i private-key
- SaveConfig = true があると
  PrivateKey = が書き込まれてしまうので
  注意

```
[Interface]
Address = 10.10.0.2/32, fd86:ea04:1111::2/128
PostUp = wg set %i private-key <(cat /etc/wireguard/privatekey)
```

使用例

- umask 077 して wg genkey | tee privatekey | wg pubkey | tee publickey で鍵ペア作成
- /etc/wireguard/wg0.conf 作成
- systemctl enable wg-quick@wg0 --now

設定例のネットワーク構成

```
10.10.0.1 (VPS 1)
 | `- 10.10.0.2 (モバイル端末 2)
 `- 10.10.0.3 (VPS 3)
```

```
fd86:ea04:1111::1 (VPS 1)
 | `- fd86:ea04:1111::2 (モバイル端末 2)
 `- fd86:ea04:1111::3 (VPS 3)
```

2 と 3 は 1 をハブとして通信、 2 の IPv6 は 1 経由でグローバルに出る

VPS 1のInterface設定例

Address には /24 のように範囲を設定

(iptable の PostUp, PostDown は実際には1行)

```
[Interface]
Address = 10.10.0.1/24
Address = fd86:ea04:1111::1/64
ListenPort = 51820
PostDown = iptables -D FORWARD -i %i -j ACCEPT;
  iptables -t nat -D POSTROUTING -o e+ -j MASQUERADE;
  ip6tables -D FORWARD -i %i -j ACCEPT;
  ip6tables -t nat -D POSTROUTING -o e+ -j MASQUERADE
PostUp = iptables -A FORWARD -i %i -j ACCEPT;
  iptables -t nat -A POSTROUTING -o e+ -j MASQUERADE;
  ip6tables -A FORWARD -i %i -j ACCEPT;
  ip6tables -t nat -A POSTROUTING -o e+ -j MASQUERADE
PostUp = wg set %i private-key <(cat /etc/wireguard/privatekey)
```

VPS 1のPeer設定例

AllowIPs には接続先での Address と同じものを指定
(モバイル端末や NAT の中のサーバーなどの Endpoint が (設定でき) ない Peer は最初は相手からつないでもらう必要がある)

```
[Peer]
PublicKey = ugpA/M4UKHyPX9ymXI2ntHJ+uHbdUpK6duGnjj9QGnI=
AllowedIPs = 10.10.0.2/32, fd86:ea04:1111::2/128
```

VPS 1のPeer設定例 (2)

直接接続できる Peer なら Endpoint も設定すると、どちらからも接続開始可能

```
[Peer]
PublicKey = HXuu2SFfqOSN2iifw3Mh7J6rDRMjIXAR3wQvkyYrKyA=
AllowedIPs = 10.10.0.3/32, fd86:ea04:1111::3/128
Endpoint = XXX.YYY.ZZZ.123:51820
```

モバイル端末2のInterface設定例

/32 で特定のアドレスのみを設定

```
[Interface]
Address = 10.10.0.2/32, fd86:ea04:1111::2/128
PostUp = wg set %i private-key <(cat /etc/wireguard/privatekey)
```

モバイル端末2のPeer設定例

AllowIPs には Peer にルーティングする IP アドレスの範囲を設定
(::/0でIPv6のデフォルト経路がwg0になる)

```
[Peer]
PublicKey = 4TBu1wBKSvH/Bl14hyxSTq1AEx3mOTxiR5e7Vpd13ng=
AllowedIPs = ::/0, 10.10.0.0/24
Endpoint = XXX.YYY.ZZZ.111:51820
PersistentKeepAlive = 25
```

ネットワークトポロジー

- 手動設定するならスター型が簡単
  - 端末の設定とハブとなるサーバーに Peer を追加していくだけ
- provision で自動化できるならメッシュ型にすると無駄な経路が減って高速かつ障害に強くなる
- (現在は VPS 間はメッシュ型でノート PC はハブになる特定のサーバーに接続中)

複数 interface 設定

- 複数物理 interface と同様に追加可能
- ListenPort は別々にする必要がある (同じポートを使うと後から start しようとした方が失敗する)
- Address などは物理 interface と同様に考えれば良い
  - 単一に複数サブネット設定も可能

トラブルシューティング

- `ip a` で IP アドレス確認
- `wg` コマンドで送受信の量を確認
- 送信できていなければ送信側の設定確認
- 送信しているのに受信していなければ途中の firewall などを確認

失敗例1

NATの中の自宅サーバーに外から接続できず

- → 最初は中から接続を開始する必要あり
  - → zabbix-agent で定期的に通信が発生するようにして解決

失敗例2

再起動したら繋がらない

- → `systemctl start` だけして `systemctl enable` していなかった
- → `wg-quick up wg0` で動作確認していて `systemctl enable` していなかった
- → ポートが衝突している → ListenPort 変更

失敗例3

wireguard で相互接続設定していた LAN 内のマシンをまとめて再起動したら繋がらない

- → mDNS の `raspi.local` などの名前で設定していた
  - → 起動時に名前解決ができず
  - → IP アドレス指定に変更

失敗例4

端末間が繋がらない

- → sysctl の `net.ipv4.ip_forward=1` や `net.ipv6.conf.all.forwarding=1` の設定漏れ
- → ufw route の許可漏れ

失敗例5

外に出られない

- → 先ほどの設定に加えて iptables, ip6tables の MASQUERADE 設定漏れ

失敗例6

複雑な設定をしようとして失敗

- 設定例の 2 と 3 との直接接続設定を追加
- 3 から 2 は 1 経由のままを目指して失敗
- → 同じサブネットに繋がっているので直接通信優先で行きと帰りを別経路にできず

wireguard-dkms

- 更新時に再起動を要求されるようになった
  https://salsa.debian.org/debian/wireguard-linux-compat/-/commit/af9f90b13118cd259227773c2a81ccfa25cf3e5d
- 変更前と同様の処理をして再起動を回避

再起動回避例

```
cat /sys/module/wireguard/version
modinfo -F version wireguard
[[ $(cat /sys/module/wireguard/version) !=
   $(modinfo -F version wireguard) ]] &&
  sudo sh -c 'systemctl stop wg-quick@wg0.service &&
              rmmod wireguard &&
              modprobe wireguard &&
              systemctl start wg-quick@wg0.service'
cat /sys/module/wireguard/version
head /run/reboot-required*
sudo rm /run/reboot-required*
```

実行例

```
$ cat /sys/module/wireguard/version
1.0.20200401
$ modinfo -F version wireguard
1.0.20200413
$ [[ $(cat /sys/module/wireguard/version) !=
     $(modinfo -F version wireguard) ]] &&
  sudo sh -c 'systemctl stop wg-quick@wg0.service &&
              rmmod wireguard &&
              modprobe wireguard &&
              systemctl start wg-quick@wg0.service'
$ cat /sys/module/wireguard/version
1.0.20200413
$ head /run/reboot-required*
==> /run/reboot-required <==

==> /run/reboot-required.pkgs <==
wireguard-dkms
$ sudo rm /run/reboot-required*
```

参考

- https://wiki.debian.org/WireGuard
- https://wiki.archlinux.jp/index.php/WireGuard
- 作って理解するWireGuard https://www.youtube.com/watch?v=grDEBt7oQho
- The Unofficial Wireguard Documentation https://github.com/pirate/wireguard-docs