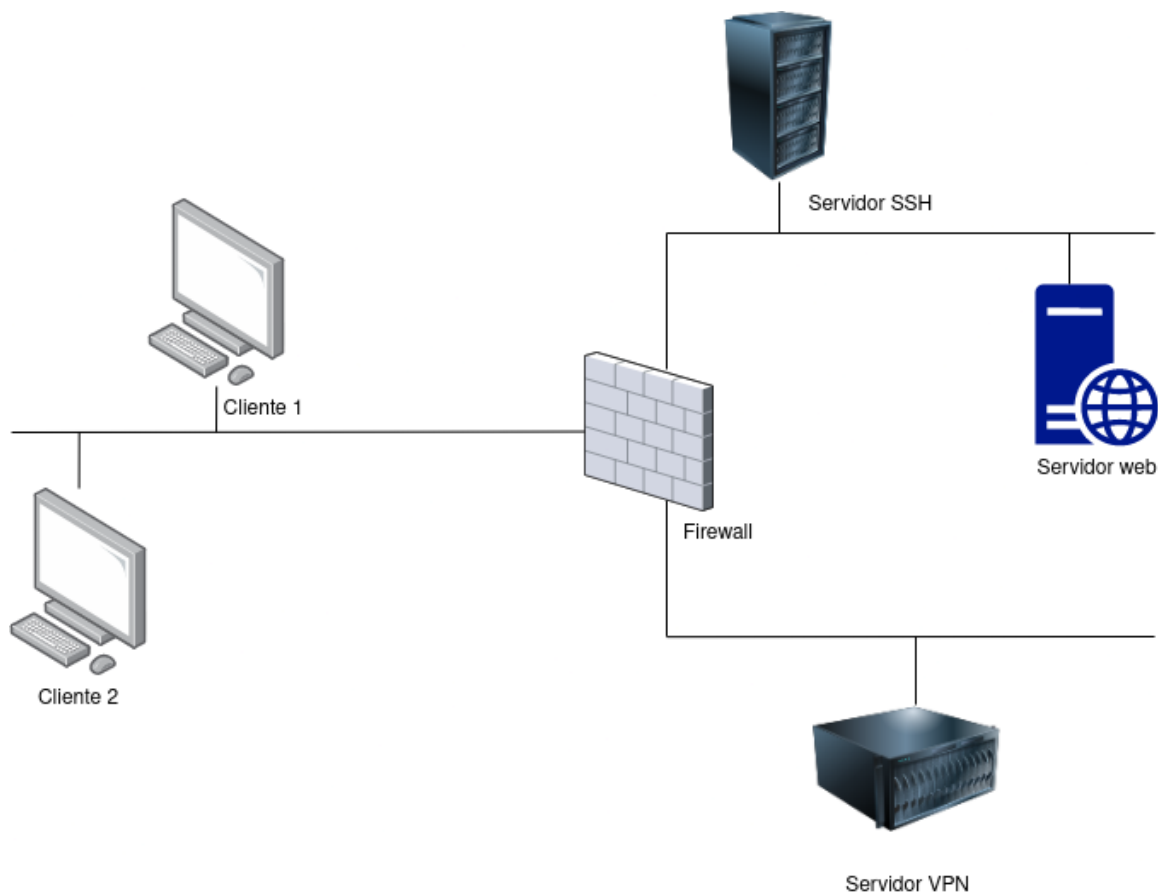


Servidor de VPN amb wireguard en un escenari muntat amb Docker

Introducció

En aquesta pràctica configurarem un escenari similar a una infraestructura empresarial real. Tenim un parell d'empleats que treballen a distància i necessiten fer ús d'una VPN. Hem evaluat les opcions i, encara que en principi havíem pensat utilitzar **OpenVPN**, finalment ens hem decidit per **Wireguard** amb la possibilitat de, si en un futur tenim més teletreballadors, canviar a OpenVPN.

Simularem la nostra infraestructura íntegrament amb contenidors Docker i serà tal que així:



Els clients establiran un túnel xifrat amb el servidor de VPN a la DMZ a través del firewall i aquest servidor, redirigirà el tràfic ja descifrat a la LAN. El tràfic de tornada seguirà el camí invers.

El túnel s'establirà mitjançant UDP i utilitzant el port **51820** entre cada client i el servidor de VPN.

Tasques a realitzar

1. Configurar la VPN per a que els dos clients puguin fer ús d'ella
2. Configurar el servidor VPN per a que assigne IPs d'una subxarxa concreta
3. Comprovar que una vegada establerta la VPN, el tràfic viatja xifrat a través de la WAN i sense xifrar cap a la LAN.
4. Configurar el firewall (iptables) per a permetre les connexions al servidor web i al servidor SSH.

Arxiu docker-compose.yml

```
1  version: '3'
2
3  # Definim les xarxes que tindrem
4  networks:
5    wan:
6      driver: bridge
7      ipam:
8        config:
9          - subnet: ${WAN}
10   lan:
11     driver: bridge
12     ipam:
13       config:
14         - subnet: ${LAN}
15   dmz:
16     driver: bridge
17     ipam:
18       config:
19         - subnet: ${DMZ}
20
21   # Definim tots els endpoints que formaran part de l'escenari
22   services:
23     client_1:
24       build:
25         context: ./dockerfiles
26         dockerfile: Dockerfile
27       container_name: client1
28       hostname: client1
29       extra_hosts:
30         - "webserver: ${IP_WEBSERVER}"
31         - "wireguard: ${IP_WIREGUARD}"
32         - "sshserver: ${IP_SSHSERVER}"
33       networks:
34         wan:
35           ipv4_address: ${IP_CLIENT_1}
```

```
36     cap_add:
37         - NET_ADMIN
38     volumes:
39         - ./wireguard:/etc/wireguard
40     privileged: true
41     command: /bin/bash -c "echo '1' > /proc/sys/net/ipv4/ip_forward &&
42 ip route del default && ip route add default via ${IP_FIREWALL_WAN} &&
43 exec sleep infinity"
44
45     client_2:
46     build:
47         context: ./dockerfiles
48         dockerfile: Dockerfile
49     container_name: client2
50     hostname: client2
51     extra_hosts:
52         - "webserver: ${IP_WEBSERVER}"
53         - "wireguard: ${IP_WIREGUARD}"
54         - "sshserver: ${IP_SSHSERVER}"
55     networks:
56         wan:
57             ipv4_address: ${IP_CLIENT_2}
58     cap_add:
59         - NET_ADMIN
60     privileged: true
61     command: /bin/bash -c "echo '1' > /proc/sys/net/ipv4/ip_forward &&
62 ip route del default && ip route add default via ${IP_FIREWALL_WAN} &&
63 exec sleep infinity"
64
65     firewall:
66     build:
67         context: ./dockerfiles
68         dockerfile: Dockerfile.firewall
69     container_name: firewall
70     hostname: firewall
71     working_dir: /usr/local/bin
72     extra_hosts:
73         - "client_1: ${IP_CLIENT_1}"
74         - "client_2: ${IP_CLIENT_2}"
75         - "webserver: ${IP_WEBSERVER}"
76         - "sshserver: ${IP_SSHSERVER}"
77         - "wireguard: ${IP_WIREGUARD}"
78     networks:
79         wan:
80             ipv4_address: ${IP_FIREWALL_WAN}
81         lan:
82             ipv4_address: ${IP_FIREWALL_LAN}
83         dmz:
84             ipv4_address: ${IP_FIREWALL_DMZ}
85     privileged: true
86     cap_add:
87         - NET_ADMIN
88     volumes:
89         - ./iptables:/usr/local/bin
90     command: /bin/bash -c "echo '1' > /proc/sys/net/ipv4/ip_forward &&
91 ip route del default && iptables-rules.sh && exec sleep infinity"
92
```

```
93     webserver:
94         build:
95             context: ./dockerfiles
96             dockerfile: Dockerfile.server
97         container_name: webserver
98         hostname: servidorweb
99         extra_hosts:
100             - "client_1: ${IP_CLIENT_1}"
101             - "client_2: ${IP_CLIENT_2}"
102             - "sshserver: ${IP_SSHSERVER}"
103             - "wireguard: ${IP_WIREGUARD}"
104         networks:
105             lan:
106                 ipv4_address: ${IP_WEBSERVER}
107         cap_add:
108             - NET_ADMIN
109         command: /bin/bash -c "ip route del default && ip route add default
110 via ${IP_FIREWALL_LAN} && service nginx start && exec sleep infinity"
111
112     ssh_server:
113         build:
114             context: ./dockerfiles
115             dockerfile: Dockerfile.ssh
116         container_name: ssh_server
117         hostname: ssh_server
118         extra_hosts:
119             - "client_1: ${IP_CLIENT_1}"
120             - "wireguard: ${IP_WIREGUARD}"
121         networks:
122             lan:
123                 ipv4_address: ${IP_SSHSERVER}
124         cap_add:
125             - NET_ADMIN
126         command: /bin/bash -c "ip route del default && ip route add default
127 via ${IP_FIREWALL_LAN} && service ssh start && exec sleep infinity"
128         restart: unless-stopped
129
130
131     wireguard:
132         image: lscr.io/linuxserver/wireguard:latest
133         container_name: wireguard
134         hostname: wireguard
135         extra_hosts:
136             - "client_1: ${IP_CLIENT_1}"
137             - "server: ${IP_WEBSERVER}"
138         networks:
139             dmz:
140                 ipv4_address: ${IP_WIREGUARD}
141         cap_add:
142             - NET_ADMIN
143             - SYS_MODULE
144         command: /bin/bash -c "apk add tcpdump && ip route del default &&
145 ip route add default via ${IP_FIREWALL_DMZ} && exec sleep infinity"
146         environment:
147             - PUID=1000
148             - PGID=1000
149             - TZ=Europe/Madrid
```

```

150         - SERVERURL=
151         - SERVERPORT=
152         - PEERS=
153         - INTERNAL_SUBNET=
154         - ALLOWEDIPS=
155         - LOG_CONFS=false # No guardar conf en logs
156     volumes:
157         - ./volumes/wireguard:/config
158         - /lib/modules:/lib/modules
159     ports:
160         - 51820:51820/udp
161     sysctls:
162         - net.ipv4.conf.all.src_valid_mark=1
163         - net.ipv4.conf.all.forwarding=1
164     restart: unless-stopped

```

La vostra tasca serà completar les variables d'entorn de les línies ressaltades del docker-compose.

Quan s'iniciï l'escenari correctament, podreu trobar les configuracions de la VPN autogenerades a partir de les variables d'ambient d'aquest **docker-compose.yml** en el contenidor de *wireguard*, en el directori `/config`.

La configuració del servidor està dins del directori `/config/wg_confs` i la dels clients dins de `/config/peerX`, on la *X* és el número de client. Esta configuració haureu de copiar-la als respectius clients, dins del directori `/etc/wireguard` amb el nom de `wg0.conf`.



Enllaç al repositori



Nota

En l'arxiu `.env` es troben guardades les variables que s'han utilitzat al `docker-compose.yml`. Podeu ficar les dades que falten al `docker-compose` de dues formes: + Posant directament el seu valor on pertoque + Assignar-li el valor a una variable en l'arxiu `.env` i posant després la variable al `docker-compose`

Detalls de la VPN

La informació relativa a la configuració de la VPN és:

1. Els clients establiran el túnel xifrat amb la VPN del servidor
2. El port que faran servir és el 51820 UDP
3. Hi hauran dos clients
4. La subxarxa de la VPN serà la `192.168.10.0/24`

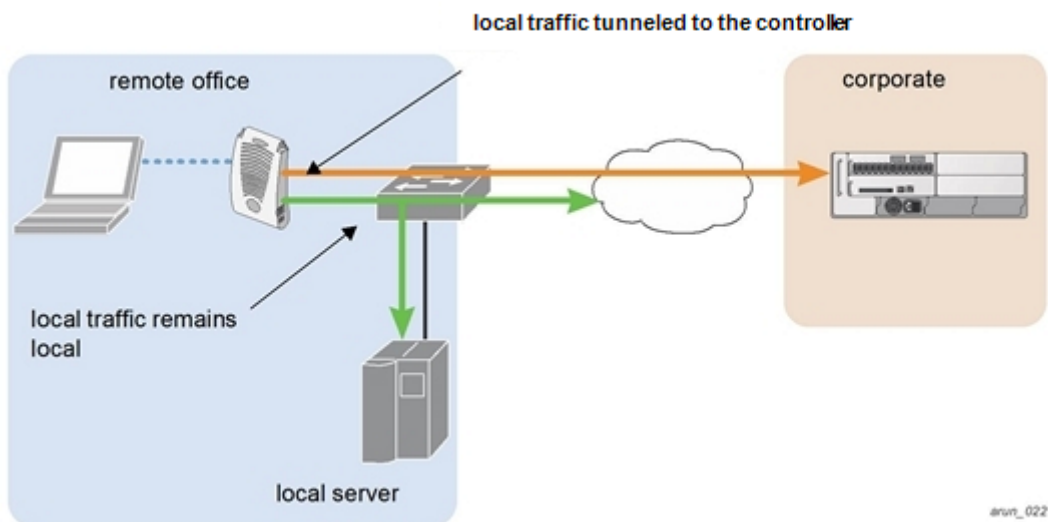
5. Les xarxes que s'han d'enrutar seran tant la LAN com la de la VPN

La VPN tindrà una subxarxa independent de les altres tres. No obstant, per a que siga funcional, farem que es pugui comunicar amb la LAN.

A més, existeix un altre concepte que hem de conèixer, l'**split tunnelling**

Mentre estigam connectats a la VPN potser que no ens interese que tot el tràfic siga encaminat per la VPN, per unes o altres raons. Un cas molt típic es quan a través de la VPN només es vol donar accés al treballador a recursos interns però no a Internet, per tal de reduir el consum d'ample de banda de la xarxa de la empresa. És a dir, que l'empleat pugui accedir a recursos compartits en la xarxa de la empresa amb la VPN però que navegue per Internet amb la seua pròpia línia d'internet.

Un altre cas típic:



Aleshores, en les línies del `docker-compose.yml` que heu d'emplenar, haureu d'indicar-le a la VPN que voleu encaminar pel túnel xifrat la pròpia subsarxa de la VPN i també la LAN, per a que puguin comunicar-se entre elles.

És molt important, **principalment per a configurar les regles del firewall**, que tingau en compte que el tràfic de la VPN segueix el següent itinerari:



client --> túnel xifrat des dels clients fins al servidor VPN --> servidor VPN fins al destí en la LAN. I la tornada del tràfic fa exactament el camí invers.

Realització de la pràctica

Així les coses, recordem les tasques a realitzar:

Tasca 1

Fer les configuracions pertinents amb les variables d'entorn del `docker-compose.yml` per tal d'establir la connexió VPN.

Una vegada tingau les configuracions, per a establir el túnel, dins del contenidor dels clients podem fer en el terminal:

```
wg-quick up wg0
```

I comproveu que es crea una interfaz anomenada `wg0` i que té assignada la IP que l'heu configurat.

Tip

Fixeu-vos que els contenidors del firewall, wireguard i clients tenen configurat uns volums. Això vol dir que les modificacions que fem en els arxius del directori local, es voran reflectides en els dels contenidors.

Tasca 2

Tots els contenidors tenen instal·lat `tcpdump` i ho podeu fer servir per a les tasques de *troubleshooting*.

- El que també heu de comprovar amb aquesta eina és que al accedir al servidor web amb `curl http://webserver` i al servidor SSH amb `ssh sad@sshserver` (contrasenya **seguretat**), amb la VPN sense connectar el tràfic viatja utilitzant el protocol TCP i/o sense xifrar.
- Que quan repetiu l'operació amb la VPN establerta, el tràfic viatja per UDP y xifrat.

Adjunta captura de pantalles ón es vega clarament la comprovació.

Tasca 3

Una vegada estiga funcionant perfectament la VPN és el torn de configurar el firewall.

- Fiqueu un arxiu anomenat `iptables-rules.sh` en el volum del contenidor del firewall. Aquest arxiu contendrà, per ordre d'aparició:
 - Neteja de totes les possibles regles que puguin estar configurades previament
 - Escriviu les regles adequades per a que la política per defecta siga denegar tot el tràfic.
 - Afegiu regles per a permetre el tràfic pel túnel VPN cap a la subxarxa LAN i als ports necessaris per als servidors HTTP i SSH.

Atenció

Per una qüestió de seguretat, configureu les regles d'iptables especificant els noms de les interfaces d'entrada, d'eixida i subxarxa destí la LAN.

Comproveu, adjuntant captures de pantalla, que les regles funcionen perfectament accedint a tots dos servidors i que els comptadors de tràfic de les regles augmenten quan les mostreu.

Tasca 4

Una vegada fet tot, prova de llevar de les xarxes que s'enruten per la VPN, la xarxa de la LAN i prova de connectar amb el servidor web i amb el servidor SSH. Què observes i per què?

Tasca 5

Com a tasca adicional, intenta configurar l'escenari per a poder accedir pel túnel VPN (interfaz wg0) al servidor Web però no al servidor SSH. A la mateixa vegada, s'ha de poder accedir per SSH des de l'interfaz WAN al servidor SSH però no al servidor web.

Contingut extra per a la recuperació de la pràctica

Per tal de completar la recuperació d'aquesta pràctica es demana, a més de tot lo anterior **configurar un contenidor docker amb la imatge `kylemanna/openvpn`** :

1. Utilitzeu una màquina Linux
2. Com a mètode d'autenticació s'utilitzaran certificats (podeu posar-le contrasenya o no)
3. El *Common Name* pel certificat serà SAD
4. La IP del servidor a la que ens connectarem serà la de la màquina



Atenció

- S'ha d'explicar i documentar detalladament cada pas que doneu.
- S'ha de fer la comprovació de que la VPN funciona:
 - Conectant des d'un client
 - Mostrant la interfaz creada per a la VPN
- **¿Tens accés a Internet des del client? Independenment de la teua resposta, justifica-la**

Referències

[Com saber les IP dels contenidors](#)

[Wireguard docs](#)

[Wireguard docker documentació](#)

[How to get started with WireGuard VPN](#)

[WireGuard Remote Access to Docker Containers](#)

[Restricciones de puertos e IP's usando Wireguard](#)