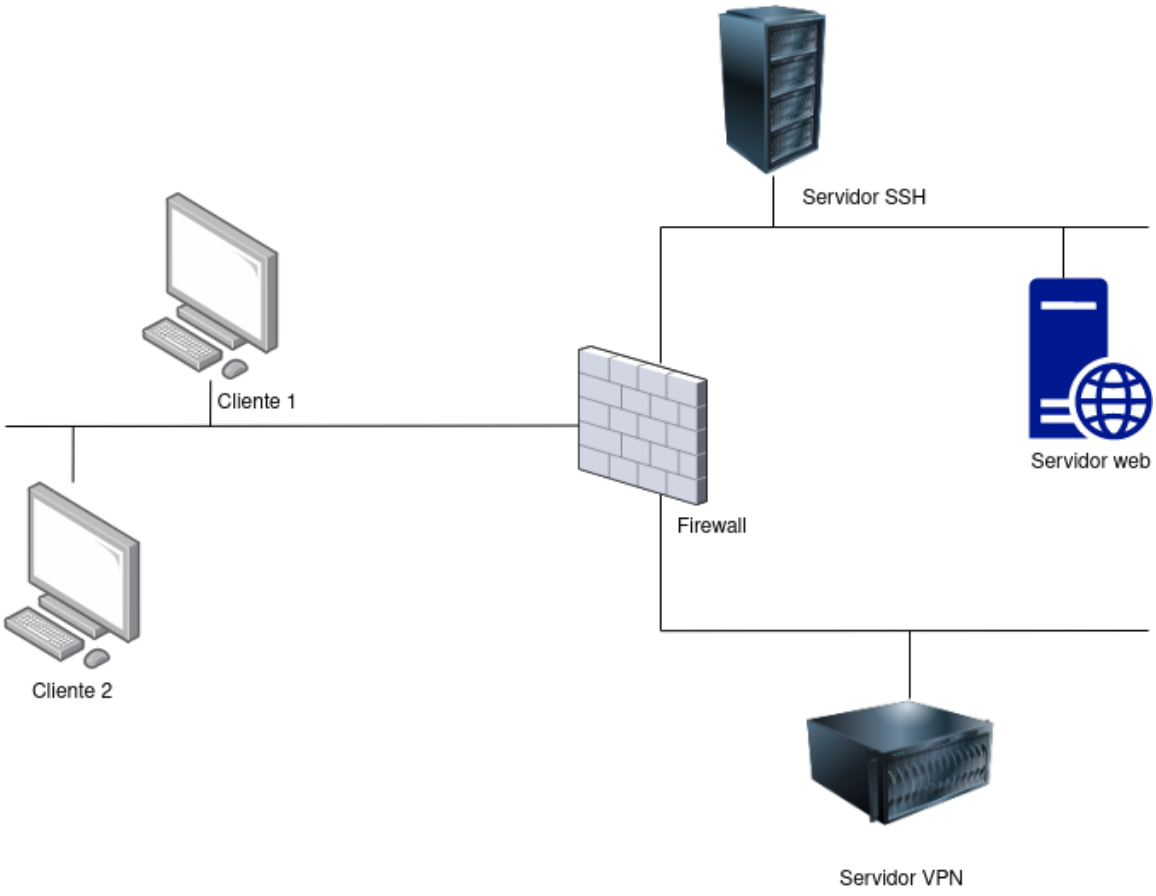
Servidor de VPN amb wireguard en un escenari muntat amb Docker

Introducció

En aquesta pràctica configurarem un escenari simil·lar a una infraestructura empresarial real. Tenim un parell d'empleats que treballen a distància i necesiten 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 puguen 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 conexions al servidor web i al servidor SSH.

Arxiu docker-compose.yml

```yaml
version: '3'

# Definim les xarxes que tindrem
networks:
  wan:
    driver: bridge
    ipam:
      config:
        - subnet: ${WAN}
  lan:
    driver: bridge
    ipam:
      config:
        - subnet: ${LAN}
  dmz:
    driver: bridge
    ipam:
      config:
        - subnet: ${DMZ}

# Definim tots els endpoints que formaran part de l'escenari
services:
  client_1:
    build:
      context: ./dockerfiles
      dockerfile: Dockerfile
    container_name: client1
    hostname: client1
    extra_hosts:
      - "webserver: ${IP_WEBSERVER}"
      - "wireguard: ${IP_WIREGUARD}"
      - "sshserver: ${IP_SSHSERVER}"
    networks:
      wan:
        ipv4_address: ${IP_CLIENT_1}
```

```
    cap_add:
      - NET_ADMIN
    volumes:
      - ./wireguard:/etc/wireguard
    privileged: true
    command: /bin/bash -c "echo '1' > /proc/sys/net/ipv4/ip_forward &&
ip route del default && ip route add default via ${IP_FIREWALL_WAN} &&
exec sleep infinity"

  client_2:
    build:
      context: ./dockerfiles
      dockerfile: Dockerfile
    container_name: client2
    hostname: client2
    extra_hosts:
      - "webserver: ${IP_WEBSERVER}"
      - "wireguard: ${IP_WIREGUARD}"
      - "sshserver: ${IP_SSHSERVER}"
    networks:
      wan:
        ipv4_address: ${IP_CLIENT_2}
    cap_add:
      - NET_ADMIN
    privileged: true
    command: /bin/bash -c "echo '1' > /proc/sys/net/ipv4/ip_forward &&
ip route del default && ip route add default via ${IP_FIREWALL_WAN} &&
exec sleep infinity"

  firewall:
    build:
      context: ./dockerfiles
      dockerfile: Dockerfile.firewall
    container_name: firewall
    hostname: firewall
    working_dir: /usr/local/bin
    extra_hosts:
      - "client_1: ${IP_CLIENT_1}"
      - "client_2: ${IP_CLIENT_2}"
      - "webserver: ${IP_WEBSERVER}"
      - "sshserver: ${IP_SSHSERVER}"
      - "wireguard: ${IP_WIREGUARD}"
    networks:
      wan:
        ipv4_address: ${IP_FIREWALL_WAN}
      lan:
        ipv4_address: ${IP_FIREWALL_LAN}
      dmz:
        ipv4_address: ${IP_FIREWALL_DMZ}
    privileged: true
    cap_add:
      - NET_ADMIN
    volumes:
      - ./iptables:/usr/local/bin
    command: /bin/bash -c "echo '1' > /proc/sys/net/ipv4/ip_forward &&
ip route del default && iptables-rules.sh && exec sleep infinity"

```

```
  webserver:
    build:
      context: ./dockerfiles
      dockerfile: Dockerfile.server
    container_name: webserver
    hostname: servidorweb
    extra_hosts:
      - "client_1: ${IP_CLIENT_1}"
      - "client_2: ${IP_CLIENT_2}"
      - "sshserver: ${IP_SSHSERVER}"
      - "wireguard: ${IP_WIREGUARD}"
    networks:
      lan:
        ipv4_address: ${IP_WEBSERVER}
    cap_add:
      - NET_ADMIN
    command: /bin/bash -c "ip route del default && ip route add default
via ${IP_FIREWALL_LAN} && service nginx start && exec sleep infinity"

  ssh_server:
    build:
      context: ./dockerfiles
      dockerfile: Dockerfile.ssh
    container_name: ssh_server
    hostname: ssh_server
    extra_hosts:
      - "client_1: ${IP_CLIENT_1}"
      - "wireguard: ${IP_WIREGUARD}"
    networks:
      lan:
        ipv4_address: ${IP_SSHSERVER}
    cap_add:
      - NET_ADMIN
    command: /bin/bash -c "ip route del default && ip route add default
via ${IP_FIREWALL_LAN} && service ssh start && exec sleep infinity"
    restart: unless-stopped


  wireguard:
      image: lscr.io/linuxserver/wireguard:latest
      container_name: wireguard
      hostname: wireguard
      extra_hosts:
      - "client_1: ${IP_CLIENT_1}"
      - "server: ${IP_WEBSERVER}"
      networks:
        dmz:
          ipv4_address: ${IP_WIREGUARD}
      cap_add:
        - NET_ADMIN
        - SYS_MODULE
      command: /bin/bash -c "apk add tcpdump && ip route del default &&
ip route add default via ${IP_FIREWALL_DMZ} && exec sleep infinity"
      environment:
        - PUID=1000
        - PGID=1000
        - 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:
              - ./volumes/wireguard:/config
              - /lib/modules:/lib/modules
            ports:
              - 51820:51820/udp
            sysctls:
              - net.ipv4.conf.all.src_valid_mark=1
              - net.ipv4.conf.all.forwarding=1
            restart: unless-stopped
```

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

Quan s'inicie l'escenari correctamente, podreu trobar les configuracions de la VPN autogenerades a partir de les variables d'ambient d'aquest **docker-compose.yaml** 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`

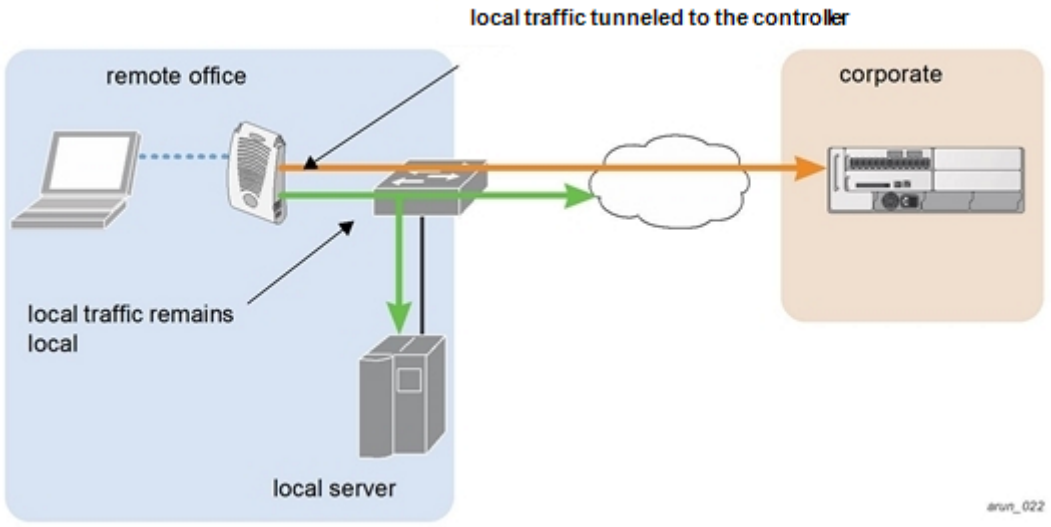
5. Les xarxes que s'han d'enrutar seran tant la LAN com la de la VPN

La VPN tindrá una subxarxa independent de les altres tres. No obstant, per a que siga funcional, farem que es puga 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 reduïr el consum d'ample de banda de la xarxa de la empresa. És a dir, que l'empleat puga accedir a recursos compartits en la xarxa de la empresa amb la VPN però que navegue per Internet amb la seua pròpia línea d'internet.

Un altre cas típic:



Aleshores, en les línees 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 puguen 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 seguent 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 conexió 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 conectar 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 contendrá, per ordre d'aparició:
  - Neteja de totes les posibles regles que puguen 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 tráfic pel túnel VPN cap a la subxarxa LAN i als ports necessaris per als servidors HTTP i SSH.

Atenció

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

Comproveu, adjuntant captures de pantalla, que les regles funcionen perfectament accedint a tots dos servidors i que els contadors de tràfic de les regles aumenten 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 conectar 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 método 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 conectarem 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