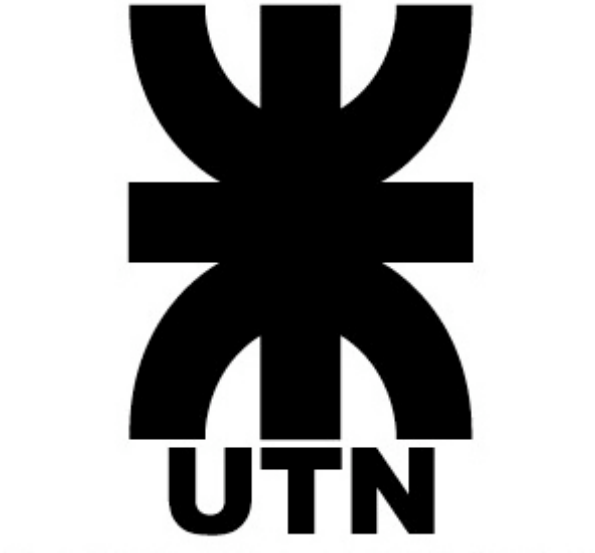
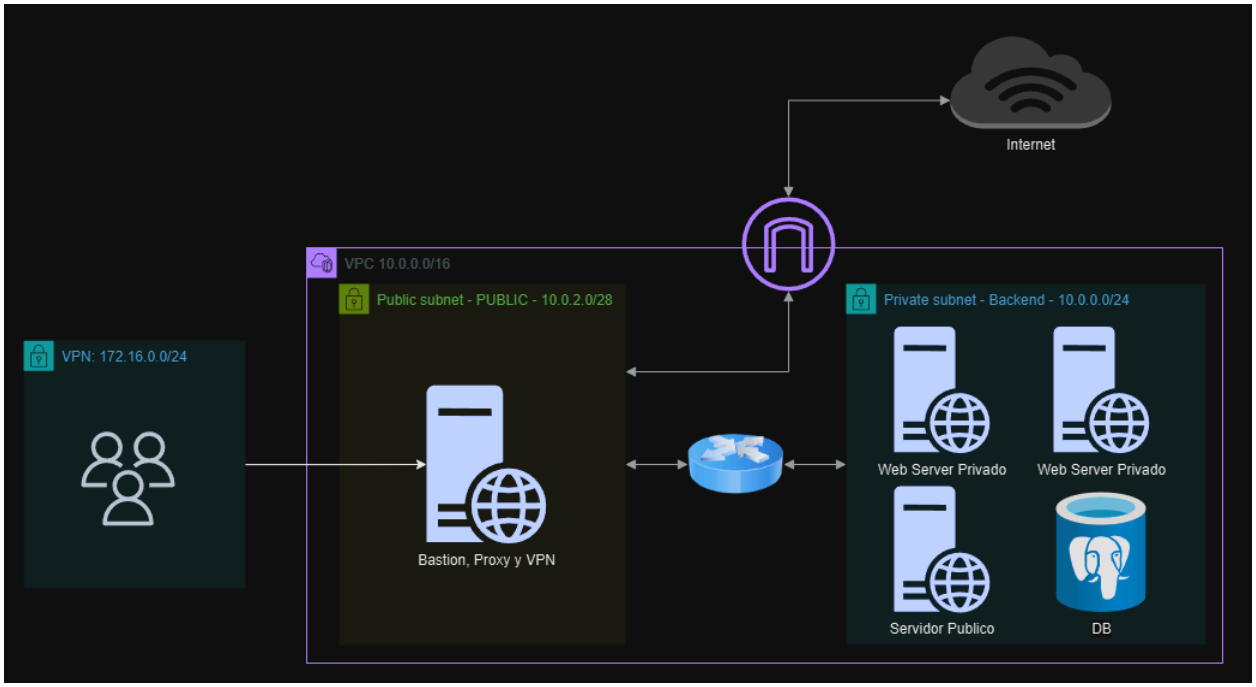
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL DE SANTA FE



Trabajo práctico final

Cátedra:	Taller de redes
Carrera:	Ingeniería en Sistemas de Información
Año:	2024
Integrantes:	Barella, Lautaro
	Haulet, Tomás

1. Infraestructura



La imagen presentada describe una arquitectura de red en AWS que está diseñada para segmentar y asegurar los diferentes componentes del sistema. A continuación, se detalla la estructura y los elementos principales de esta infraestructura:

1- Virtual Private Cloud (VPC)
CIDR: `10.0.0.0/16`
Este es el contenedor principal de la infraestructura de red. Dentro de la VPC se definen dos subredes principales:

- Subred Pública: `10.0.2.0/28`
- Subred Privada (Backend): `10.0.0.0/24`

2- Subred Pública
Esta subred está diseñada para alojar recursos accesibles desde el exterior, pero controlados. Un servidor (Bastión, Proxy y VPN) que cumple múltiples funciones:

- Como bastión, permite el acceso seguro a la infraestructura privada.
- Funciona como proxy para canalizar tráfico autorizado.
- Actúa como servidor VPN, proporcionando acceso remoto seguro para los usuarios externos desde la red `172.16.0.0/24`.

3- Subred Privada (Backend)
Diseñada para alojar recursos sensibles que no tienen acceso directo a Internet.
Componentes alojados:

- Web Servers Privados: Dos instancias de servidores web que procesan solicitudes en el backend.

- Servidor Público: Un servidor accesible a través del bastión/proxy.
- Base de Datos (DB): Servicio de base de datos.

4- Comunicación y Seguridad

- VPN: La red `172.16.0.0/24` representa los usuarios que acceden de forma remota a través de una conexión VPN segura. El tráfico autorizado desde esta red pasa al bastión/proxy para luego acceder a los recursos internos.
- Internet Gateway: Proporciona conectividad a Internet para la subred pública, permitiendo que el bastión, proxy y VPN se comuniquen con servicios externos.

2. Servicio VPN

Para el servicio VPN primero vamos a la instancia del bastión y generamos el par de claves pública y privada de la siguiente manera:

```
wg genkey > tp_taller_private
wg pubkey < tp_taller_private > tp_taller_public
```

Luego configuramos la interfaz de la siguiente manera

```
vi tp_interface.conf

[Interface] PrivateKey = <PRIVATE KEY>
Address = 172.16.0.100/32
ListenPort = 51820

[Peer]
PublicKey = <PUBLIC KEY PEER 1>
AllowedIPs = <IP_peer_1>/32

[Peer]
PublicKey = <PUBLIC KEY PEER 2>
AllowedIPs = <IP_peer_2>/32

[Peer]
PublicKey = <PUBLIC KEY PEER 3>
AllowedIPs = <IP_peer_3>/32
```

Luego iniciamos los servicios mediante

```
systemctl enable --now wg-quick@tp_interface
```

3. Balanceador de carga NGINX

Comenzamos creando el archivo /etc/nginx/conf.d/balancer.conf con el siguiente contenido

```
upstream backend{
      server <IP_WP1>:80;
      server <IP_WP2>:80;
}

server {
 listen <IP_SERVER_VPN>:80;
 listen [::]:80;
 access_log /var/log/nginx/reverse-access.log;
 error_log /var/log/nginx/reverse-error.log;
 location / {
 proxy_pass http://backend ;
 }
}
```

Luego activamos el servicio mediante el siguiente comando

```
systemctl start nginx
```

4. Servicio proxy Squid

Comenzamos modificando el archivo /etc/squid/squid.conf en el cual agregamos los siguientes datos

```
acl localnet src 0.0.0.1-0.255.255.255 # RFC 1122 "this" network (LAN)
acl vpn src 172.16.0.0/24
acl localnet src 10.0.0.0/8 # RFC 1918 local private network (LAN)
acl localnet src 100.64.0.0/10 # RFC 6598 shared address space (CGN)
acl localnet src 169.254.0.0/16 # RFC 3927 link-local (directly plugged)
machines
acl localnet src 172.16.0.0/12 # RFC 1918 local private network (LAN)
acl localnet src 192.168.0.0/16 # RFC 1918 local private network (LAN)
acl localnet src fc00::/7 # RFC 4193 local private network range
```

```
acl localnet src fe80::/10 # RFC 4291 link-local (directly plugged)
machines

acl dominios dstdomain "/etc/squid/lista.txt"
acl dominios_hora dstdomain "/etc/squid/lista2.txt"
acl horario_permitido time 16:00-23:59
acl horario_permitido2 time 00:00-08:00

acl SSL_ports port 443
acl SSL_ports port 80
acl Safe_ports port 80 # http
acl Safe_ports port 21 # ftp
acl Safe_ports port 443 # https
acl Safe_ports port 70 # gopher
acl Safe_ports port 210 # wais
acl Safe_ports port 1025-65535 # unregistered ports
acl Safe_ports port 280 # http-mgmt
acl Safe_ports port 488 # gss-http
acl Safe_ports port 591 # filemaker
acl Safe_ports port 777 # multiling http
acl CONNECT method CONNECT

http_port 3128

# Deny requests to certain unsafe ports
http_access deny !Safe_ports
# Deny CONNECT to other than secure SSL ports
http_access deny CONNECT !SSL_ports
# Only allow cachemgr access from localhost
http_access allow vpn dominios
http_access allow vpn dominios_hora horario_permitido
http_access allow vpn dominios_hora horario_permitido2
http_access deny manager
http_access deny all
# We strongly recommend the following be uncommented to protect innocent
# web applications running on the proxy server who think the only
# one who can access services on "localhost" is a local user
#http_access deny to_localhost
```

Luego reiniciamos el servicio para que se guarden los cambios mediante:

```
systemctl restart squid
```

5. Servicio de comunicaciones, Asterisk

Para configurar este servicio primero vamos a /etc/asterisk, aquí dentro del sip.conf le decimos al asterisk cuales son nuestras ip pública e ip privada del servicio VPN.

```
localnet=172.16.0.0/24
externaddr = 172.16.0.100

[tp_taller](!)
        type=friend
        context=tp
        host=dynamic
        nat=force_rport, comedia
        dtmfmode=auto
        disallow=all
[telefono1](tp_taller)
        secret=1234
        allow=ulaw
[telefono2](tp_taller)
        secret=1234
        allow=ulaw
```

Luego definimos el contexto dentro del extensions.conf

```
[tp]
        exten => 101,1,Dial(SIP/telefono1)
        exten => 102,1,Dial(SIP/telefono2)
        exten => 200,1,Answer()
                same => n,Playback(hello-world)
                same => n, Hangup()
```

Luego recargamos el asterisk para activar los cambios mediante:

```
asterisk -rvvvv
sip reload
dialplan reload
```

6. IPTables

Bastión

Para el bastion aplicamos las siguientes reglas:

```
#!/bin/bash

red_bastion="10.0.2.0/28"
red_back="10.0.0.0/24"
red_usuario="10.0.1.0/24"
red_vpn="172.16.0.0/24"

bastion="10.0.2.14"
vpn_bastion="172.16.0.100"
server_public="10.0.0.217"
server_private1="10.0.0.62"
server_private2="10.0.0.206"
server_bdd="10.0.0.118"
usuario="10.0.1.155"

#Para que sirva de ruter
sysctl -w net.ipv4.ip_forward=1
sysctl -p
#Para DNS
cp /etc/resolv.conf.backup /etc/resolv.conf

# Limpiar tablas
iptables -F INPUT
iptables -F OUTPUT
iptables -F FORWARD
iptables -t nat -F PREROUTING
iptables -t nat -F POSTROUTING

# Politica PD drop
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

# Aceptar conexiones ya establecidas y relacionadas
```

```
iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -m state --state RELATED,ESTABLISHED -j ACCEPT

########## NAT
# SSH
iptables -t nat -A PREROUTING -i tp -d ${vpn_bastion} -p tcp --dport 2222
-j DNAT --to-destination ${server_public}:22
iptables -t nat -A PREROUTING -i tp -d ${vpn_bastion} -p tcp --dport 2223
-j DNAT --to-destination ${server_private1}:22
iptables -t nat -A PREROUTING -i tp -d ${vpn_bastion} -p tcp --dport 2224
-j DNAT --to-destination ${server_private2}:22
iptables -t nat -A PREROUTING -i tp -d ${vpn_bastion} -p tcp --dport 2225
-j DNAT --to-destination ${server_bdd}:22


#Web
iptables -t nat -A PREROUTING -d ${bastion} -i eth0 -p tcp --dport 80 -j
DNAT --to-destination ${server_public}:80

#MASQUERADE
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE




########## NO NAT
# Permitir SSH al bastion
iptables -A INPUT -d ${bastion} -p tcp --dport 22 -j ACCEPT

#Permitir SSH a los servidores desde la vpn
iptables -A FORWARD -s ${red_vpn} -d ${red_back} -p tcp --dport 22 -j
ACCEPT

#Permitir acceder a los servidores web
#Privados (usamos nginx asi que no podemos forwardear)
iptables -A INPUT -d ${vpn_bastion} -p tcp --dport 80 -j ACCEPT
iptables -A OUTPUT -d ${red_back} -p tcp --dport 80 -j ACCEPT
#Publicos
iptables -A FORWARD -d ${server_public} -p tcp --dport 80 -j ACCEPT
```

```
#Permitir ping a los servidores desde cualquier host de la vpn y al
bastion, notar que el BASTION no se puede hacer ping a sÃ mismo
iptables -A INPUT -i tp -p icmp --icmp-type echo-request -j ACCEPT
iptables -A OUTPUT -o tp -p icmp --icmp-type echo-reply -j ACCEPT
iptables -A OUTPUT -o tp -p icmp --icmp-type echo-request -j ACCEPT
iptables -A INPUT -i tp -p icmp --icmp-type echo-reply -j ACCEPT
iptables -A FORWARD -s ${red_vpn} -d ${red_vpn} -p icmp --icmp-type
echo-request -j ACCEPT

#Servicios
#VPN
iptables -A FORWARD -i tp -o tp -s ${red_vpn} -d ${red_vpn} -j ACCEPT

#Wireguard
iptables -A INPUT -i eth0 -p tcp --dport 51820 -j ACCEPT
iptables -A INPUT -i eth0 -p udp --dport 51820 -j ACCEPT

#Permitir que accedan a internet
#Bastion
iptables -A OUTPUT -o eth0 -s ${bastion} -p tcp --dport 80 -j ACCEPT
iptables -A OUTPUT -o eth0 -s ${bastion} -p tcp --dport 443 -j ACCEPT
iptables -A OUTPUT -o eth0 -s ${bastion} -p udp --dport 53 -j ACCEPT
iptables -A OUTPUT -s ${bastion} -p tcp --dport 53 -j ACCEPT

#Asterisk
iptables -A INPUT -i tp -p udp --dport 5060 -j ACCEPT
iptables -A INPUT -i tp -p udp --dport 10000:50000 -j ACCEPT

#Proxy
iptables -A INPUT -i tp -s ${red_vpn} -p tcp --dport 3128 -j ACCEPT
```

Servicios webs

Para los servicios webs aplicamos las siguientes reglas:

```
#!/bin/bash

red_bastion="10.0.2.0/28"
red_back="10.0.0.0/24"
```

```
red_usuario="10.0.1.0/24"
red_vpn="172.16.0.0/24"

bastion="10.0.2.14"
vpn_bastion="172.16.0.100"
server_public="10.0.0.217"
server_private1="10.0.0.62"
server_private2="10.0.0.206"
server_bdd="10.0.0.118"
usuario="10.0.1.155"

# Limpiar tablas
iptables -F INPUT
iptables -F OUTPUT
iptables -F FORWARD
iptables -t nat -F PREROUTING
iptables -t nat -F POSTROUTING

# Politica PD drop
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

# Aceptar conexiones ya establecidas y relacionadas
iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -m state --state RELATED,ESTABLISHED -j ACCEPT

#SSH
iptables -A INPUT -d ${red_back} -p tcp --dport 22 -j ACCEPT

#Permitir ping
iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT
iptables -A OUTPUT -p icmp --icmp-type echo-reply -j ACCEPT

#Permitir servicio web
iptables -A INPUT -s ${bastion} -p tcp  --dport 80 -j ACCEPT
```

Servicios MySQL

Para el servicio de base de datos MySQL aplicamos las siguientes reglas:

```
#!/bin/bash

red_bastion="10.0.2.0/28"
red_back="10.0.0.0/24"
red_usuario="10.0.1.0/24"
red_vpn="172.16.0.0/24"

bastion="10.0.2.14"
vpn_bastion="172.16.0.100"
server_public="10.0.0.217"
server_private1="10.0.0.62"
server_private2="10.0.0.206"
server_bdd="10.0.0.118"
usuario="10.0.1.155"

# Limpiar tablas
iptables -F INPUT
iptables -F OUTPUT
iptables -F FORWARD
iptables -t nat -F PREROUTING
iptables -t nat -F POSTROUTING

# Politica PD drop
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

# Aceptar conexiones ya establecidas y relacionadas
iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -m state --state RELATED,ESTABLISHED -j ACCEPT

#SSH
iptables -A INPUT -d ${red_back} -p tcp --dport 22 -j ACCEPT
```

```
#Permitir ping
iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT
iptables -A OUTPUT -p icmp --icmp-type echo-reply -j ACCEPT

#Permitir acceso mysql
iptables -A INPUT -s ${bastion} -p tcp  --dport 3306 -j ACCEPT
```