Universidad de Castilla - La Mancha

Seguridad en Redes

Escuela Superior de Informática

Diseño e Implementación de una Topología Segura

Autor:
Álvaro Cerdá Pulla

Profesor:
David Villa Alises

19 de Enero de 2021

Índice

Enlace a Github
`https://github.com/alvaroc20/SR-Topologias`

1. Servicio Web

Para proporcionar el servicio web, hacemos uso del servidor **nginx** que viene instalado por defecto en la máquina proporcionada. Con él creamos un servidor que aloje una pequeña web. Para realizarlo, tendremos que buscary modificar la configuración de nginx. Lo que tendremos que tener dentro de esa configuración, son atributos básicos como:

- Puerto en el que se escucha
- URL de la página
- Dirección de reenvío
- Número de conexiones que permitiremos

nginx.conf

```
events {
  worker_connections  4096;  ## Default: 1024
}
http {
    server {
      listen  80;
      server_name www.alvaroYJuanmaSR.com alvaroYJuanmaSR.com;
      location /app {
          proxy_pass http://127.0.0.1:8080;
      }
  }
}
```

De esta manera, ya tendríamos creado nuestro servicio web. Lógicamente, hasta este punto solo se podría acceder mediante la IP donde se aloje el servicio web, ya que aún, no tiene configurado el DNS.

2. DNS y DHCP

Ya tenemos configurado el servicio web, por lo que ahora queremos buscar por un nombre, y no por una dirección. De manera, que haremos uso de un sistema de nombres de dominio (DNS), que cuando ingresemos la URL nos redireccione a la dirección donde se aloja.

Además de esto, debemos realizar servicio DHCP, es decir, que el servidor asigne direcciones IP válidas a los dispositivos que estén solicitándolo.

Esto, como dice en el enunciado, lo resolvemos con **dnsmasq**, que es un servidor caché de DNS, que adicionalmente, dispone de servidor DHCP y permite resolver los nombres de los PCs a los que se le ha asignado dirección IP dinámica.

Como hemos hecho en el paso anterior, debemos buscar la configuración del dnsmasq, y realizarlo según nuestros intereses:

- Configurar puertos de escucha DNS
- Interfaces del DNS
- Configurar el rango de direcciones que puede asignar DHCP

dnsmasq.conf

```
#Configuring dnsmasq Server
# DNS
listen-address =::1,127.0.0.1,192.167.0.1,10.0.0.1
interface=eth3
interface=eth2
expand-hosts
# DHCP
dhcp-range=192.167.0.2,192.167.0.254,12h
```

En el DNS estamos escuchando 2 redes, además del local:

- **Red DMZ**: 10.0.0.1
- **Red MZ**: 192.167.0.1

Para **DHCP**, damos un rangos de direcciones entre 192.167.0.2 - 192.167.0.254 que cambiarán cada 12 horas.

También necesitaremos configurar el archivo de **resolv**, donde estará configurado el servidor DNS externo, en nuestro caso de la DMZ.

resolv.conf

```
nameserver 10.0.0.1
```

Ahora que tenemos el servidor DNS y DHCP funcionando, vamos a ir más alla. Lo que tenemos que hacer a continuación, es que nuestro DNS también resuelva los nombres y las IPs de nuestra red. Para poder realizar esto debemos acceder al fichero hosts.

hosts

```
127.0.1.1       router    router
10.0.0.2        www.alvaroYJuanmaSR.com alvaroYJuanmaSR.com
# The following lines are desirable for IPv6 capable hosts
::1         localhost ip6-localhost ip6-loopback
ff02::1     ip6-allnodes
ff02::2     ip6-allrouters
```

3. Proxy

Para realizar el servicio proxy haremos uso de **Squid**, una de las aplicaciones más populares para esta función, también instalado por defecto en la máquina **router**. En el fichero de configuración **squid.conf** se establece las direcciones y puertos a los que esta permitido el tráfico dentro de la red además de varios opciones de configuración de la herramienta.

```
acl localnet src 0.0.0.1-0.255.255.255  # RFC 1122 "this" network (LAN)
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 localnet src 10.0.0.1
acl localnet src 10.0.0.2
acl localnet src 192.167.0.1
acl localnet src 192.167.0.0/8
acl localnet src 127.0.0.1
acl localnet src 192.168.1.150

acl SSL_ports port 443
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

#
# Recommended minimum Access Permission configuration:
#
# 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 localhost manager
http_access deny manager
http_access allow localnet
http_access allow localhost

# And finally deny all other access to this proxy
http_access deny all

# Squid normally listens to port 3128
http_port 3128
```

```
50
51 # Leave coredumps in the first cache dir
52 coredump_dir /var/spool/squid
53 #
54 # Add any of your own refresh_pattern entries above these.
55 #
56 refresh_pattern ^ftp:           1440    20%     10080
57 refresh_pattern ^gopher:        1440    0%      1440
58 refresh_pattern -i (/cgi-bin/|\?) 0     0%      0
59 refresh_pattern \/(Packages|Sources)(|\.bz2|\.gz|\.xz)$ 0 0% 0 refresh-ims
60 refresh_pattern \/Release(|\.gpg)$ 0 0% 0 refresh-ims
61 refresh_pattern \/InRelease$ 0 0% 0 refresh-ims
62 refresh_pattern \/(Translation-.*)(|\.bz2|\.gz|\.xz)$ 0 0% 0 refresh-ims
63 #example pattern for deb packages
64 #refresh_pattern (\.deb|\.udeb)$   129600 100% 129600
65 refresh_pattern .               0       20%     4320
```

Se puede comprobar el funcionamiento del proxy junto con el servicio web. Se puede comprobar con el siguiente comando:

```
$ systemctl status squid
```

En caso de que el sistema esté correctamente activo nos mostrará por pantalla lo siguiente:

```
vagrant@router:/etc/squid$ systemctl status squid
● squid.service - LSB: Squid HTTP Proxy version 3.x
   Loaded: loaded (/etc/init.d/squid; generated; vendor preset: enabled)
   Active: active (running) since Sun 2021-01-17 16:28:50 GMT; 9min ago
     Docs: man:systemd-sysv-generator(8)
  Process: 3253 ExecStop=/etc/init.d/squid stop (code=exited, status=0/SUCCESS)
  Process: 3278 ExecStart=/etc/init.d/squid start (code=exited, status=0/SUCCESS)
 Main PID: 3319 (squid)
    Tasks: 4 (limit: 4915)
   CGroup: /system.slice/squid.service
           ├─3317 /usr/sbin/squid -YC -f /etc/squid/squid.conf
           ├─3319 (squid-1) -YC -f /etc/squid/squid.conf
           ├─3320 (logfile-daemon) /var/log/squid/access.log
           └─3321 (pinger)
```

Figura 1: Estado de squid

Ahora podemos realizar una petición web a nuestra página y deberemos obtener una respuesta **HTTP** satisfactoria:

```
$ curl -I alvaroYJuanmaSR.com
```

```
vagrant@pc1:~$ curl -I alvaroYJuanmaSR.com
HTTP/1.1 200 OK
Server: nginx/1.10.3
Date: Tue, 19 Jan 2021 17:39:50 GMT
Content-Type: text/html
Content-Length: 612
Last-Modified: Tue, 31 Jan 2017 15:01:11 GMT
Connection: keep-alive
ETag: "5890a6b7-264"
Accept-Ranges: bytes
```

Figura 2: Petición HTTP

Para comprobar que realmente el proxy realiza su función podemos añadir una configuración de restricción de nuestro dominio en el archivo **squid.conf** y denegar el acceso de este servicio:

```
acl blocked_sites dstdomain www.alvaroYJuanmaSR.com
acl blocked_sites dstdomain alvaroYJuanmaSR.com

http_access deny blocked_sites
```

El resultado devuelto al realizar de nuevo la petición nos devuelve un **error** 403 provocado por nuestro proxy:

```
vagrant@pc1:~$ curl -x http://192.168.1.150:3128  -I http://google.com
HTTP/1.1 403 Forbidden
Server: squid/3.5.23
Mime-Version: 1.0
Date: Sun, 17 Jan 2021 12:45:13 GMT
Content-Type: text/html;charset=utf-8
Content-Length: 3515
X-Squid-Error: ERR_ACCESS_DENIED 0
Vary: Accept-Language
Content-Language: en
X-Cache: MISS from router
X-Cache-Lookup: NONE from router:3128
Via: 1.1 router (squid/3.5.23)
Connection: keep-alive
```

Figura 3: Petición satisfactoria

4. Iptables según Política de Seguridad

Las reglas acordes para la configuración de **iptables** con respecto a nuestra política de seguridad vienen definidas en el archivo de configuración **iptables-rules** el cual se configura de manera automática en la máquina y son las siguientes:

- Permitir loopback

```
-A INPUT -i lo -j ACCEPT
-A OUTPUT -o lo -j ACCEPT
```

- Permitir ICMP contra nodos de internet

```
-A OUTPUT -p ICMP -j ACCEPT
```

- Permitir UDP, DNS,HTTP ,HTTPS y FTP pasivo

```
-A INPUT -p tcp -m state --state NEW,ESTABLISHED -m tcp --dport 22 -j
    ACCEPT
-A INPUT -j ACCEPT -i eth2 -p tcp --sport 1024:65535 -m multiport --dports 80,443
-A OUTPUT -j ACCEPT -o eth1 -p tcp --sport 1024:65535 -m multiport --dports
    80,443
-A OUTPUT -o eth1 -p udp --sport 1021:65535 --dport 53 -m state --state NEW -j
    ACCEPT
-A INPUT -i eth1 -m state --state ESTABLISHED,RELATED -j ACCEPT
```

- Acceso ilimitado en LAN

```
1 -A FORWARD -i eth1 -o eth2 -m state --state NEW,ESTABLISHED,RELATED -j
    ACCEPT
2 -A FORWARD -i eth1 -o eth3 -m state --state NEW,ESTABLISHED,RELATED -j
    ACCEPT
3 -A FORWARD -i eth2 -o eth1 -m state --state NEW,ESTABLISHED,RELATED -j
    ACCEPT
4 -A FORWARD -i eth2 -o eth3 -m state --state NEW,ESTABLISHED,RELATED -j
    ACCEPT
5 -A FORWARD -i eth3 -o eth1 -m state --state NEW,ESTABLISHED,RELATED -j
    ACCEPT
6 -A FORWARD -i eth3 -o eth2 -m state --state NEW,ESTABLISHED,RELATED -j
    ACCEPT
```

- Introducir tráfico en el LOG

```
1 -A INPUT -j LOG
```

- Redirigir trafico puerto 90 a proxy y enmascarar interfaz red pública (tabla NAT)

```
1 -A PREROUTING -i eth2 -p tcp --dport 80 -j DNAT --to 192.168.1.150:3128
2 -A PREROUTING -i eth1 -p tcp --dport 80 -j REDIRECT --to-port 3128
3 -A POSTROUTING -o eth1 -j MASQUERADE
```

Nota: La IP 192.168.1.150 va cambiando según la que asigne DCHP a la interfaz eth1

Podemos comprobar el LOG del firewall ejecutando el siguiente comando:

```
$ sudo tail -f /var/log/kern.log
```

5. Bibliografía

- http://recursostic.educacion.es/observatorio/web/gl/software/software-general/638-servidor-dns-sencillo-en-linux-con-dnsmasq
- https://likegeeks.com/es/servidor-proxy-de-linux-squid/
- https://ajpdsoft.com/modules.php?name=News&file=article&sid=441
- https://www.ionos.es/digitalguide/servidores/configuracion/nginx-tutorial-primer