

INFORME PRÁCTICA 4

Julio César Velásquez Cárdenas 1397896

David Sánchez González 1401641

Asignatura: Gestió i Administració de Xarxes

Trabajo previo

- ¿Qué comando nos permite ver los límites del usuario de procesos, file descriptors, memoria, etc?
 - Ese comando se llama *ulimit*. Su uso es muy simple, basta con hacer *ulimit -a nombre_usuario*, dónde *nombre_usuario* será el usuario de la máquina del que queramos comprobar sus límites dentro de la máquina.
 - Para estas pruebas comprobaremos los límites de tres de los usuarios en la máquina, **root**, **adminp** y **user1** (usuario creado mediante el comando *adduser* en una práctica anterior).

```
root@master-1-8:~# ulimit -a root
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
scheduling priority     (-e) 0
file size               (blocks, -f) unlimited
pending signals         (-i) 7929
max locked memory       (kbytes, -l) 64
max memory size         (kbytes, -m) unlimited
open files              (-n) 1024
pipe size               (512 bytes, -p) 8
POSIX message queues    (bytes, -q) 819200
real-time priority      (-r) 0
stack size              (kbytes, -s) 8192
cpu time                (seconds, -t) unlimited
max user processes      (-u) 7929
virtual memory          (kbytes, -v) unlimited
file locks              (-x) unlimited
```

```
root@master-1-8:~# ulimit -a adminp
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
scheduling priority     (-e) 0
file size               (blocks, -f) unlimited
pending signals         (-i) 7929
max locked memory       (kbytes, -l) 64
max memory size         (kbytes, -m) unlimited
open files              (-n) 1024
pipe size               (512 bytes, -p) 8
POSIX message queues    (bytes, -q) 819200
real-time priority      (-r) 0
stack size              (kbytes, -s) 8192
cpu time                (seconds, -t) unlimited
max user processes      (-u) 7929
virtual memory          (kbytes, -v) unlimited
file locks              (-x) unlimited
```

```
root@master-1-8:~# ulimit -a user1
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
scheduling priority     (-e) 0
file size               (blocks, -f) unlimited
pending signals         (-i) 7929
max locked memory       (kbytes, -l) 64
max memory size         (kbytes, -m) unlimited
open files              (-n) 1024
pipe size               (512 bytes, -p) 8
POSIX message queues    (bytes, -q) 819200
real-time priority      (-r) 0
stack size              (kbytes, -s) 8192
cpu time                (seconds, -t) unlimited
max user processes      (-u) 7929
virtual memory          (kbytes, -v) unlimited
file locks              (-x) unlimited
```

- Como se puede ver en la captura de pantalla, todos los usuarios tienen los mismos límites, como un máximo de 7929 procesos.
- Si los archivos del log del sistema comienzan a crecer indefinidamente, ¿cómo evitaremos que nos consuma todo el espacio del disco?
 - En caso de que los archivos de log empiecen a ocupar demasiado espacio, primero habría que comprobar los logs para asegurar que todo funciona correctamente, y en segunda instancia cambiar las políticas de *logrotate* para que sea más agresivo a la hora de comprimir los archivos de log. Ya que como se puede ver en la siguiente captura de pantalla, es un proceso realizado automáticamente según unos parámetros definidos.

```

root@master-1-8:~# ls -l -h /var/log/apache2/
total 1.1M
-rw-r----- 1 root adm      0 Nov 18 17:45 access.log
-rw-r----- 1 root adm  7.0K Nov 12 18:57 access.log.1
-rw-r----- 1 root adm   307 Nov 10 22:28 access.log.2.gz
-rw-r----- 1 root adm   288 Nov  9 17:38 access.log.3.gz
-rw-r----- 1 root adm   280 Nov  8 11:33 access.log.4.gz
-rw-r----- 1 root adm   404 Nov  7 11:47 access.log.5.gz
-rw-r----- 1 root adm  2.9K Nov 27 18:25 error.log
-rw-r----- 1 root adm   3.1K Nov 26 20:50 error.log.1
-rw-r----- 1 root adm   3.4K Nov 18 17:45 error.log.2.gz
-rw-r----- 1 root adm   940 Nov 12 10:07 error.log.3.gz
-rw-r----- 1 root adm   948 Nov 10 21:37 error.log.4.gz
-rw-r----- 1 root adm   891 Nov  9 13:33 error.log.5.gz
-rw-r----- 1 root adm   565 Nov  8 10:52 error.log.6.gz
-rw-r----- 1 root adm      0 Nov 18 17:45 other_vhosts_access.log
-rw-r----- 1 root adm  1.1M Nov 12 22:06 other_vhosts_access.log.1
-rw-r--r-- 1 root root      0 Nov 12 19:13 proxy-access.log
-rw-r--r-- 1 root root      0 Nov 12 19:13 proxy-error.log

```

- Como se puede ver en la captura, ya se han realizado diferentes iteraciones para comprimir los archivos de log en la carpeta de Apache.

```

# see "man logrotate" for details
# rotate log files weekly
#weekly
daily
# keep 4 weeks worth of backlogs
rotate 2

# create new (empty) log files after rotating old ones
create

# uncomment this if you want your log files compressed
#compress

```

- Comprobando el archivo de configuración de *logrotate* (*/etc/logrotate.conf*) podemos ver que se hace una rotación y compresión diaria de los logs y crear un archivo una vez terminado el proceso.
- Con *rotate 2* se está indicando que se mantenga una copia adicional del log.
- **¿Cómo podemos evitar que un usuario concreto pueda dejar ejecutar un binario que se encuentra en */usr/bin*?**
 - Podemos hacer uso de ACLs y eliminar los permisos de ejecución de un directorio o directorios en concreto.
 - Para ello primero debemos instalar el paquete *acl* disponible en las librerías de Debian.

```

root@master-1-8:~# setfacl -m u:user1:r /usr/bin
root@master-1-8:~# █

```

- Una vez ejecutado ese comando, el *user1* sólo tendrá acceso a la lectura de los ficheros en */usr/bin*, pero no tendrá permiso de ejecución sobre los mismos. De esta forma, *user1* tampoco podrá iniciar sesión en la máquina.

- **¿Qué es un ataque de DoS? ¿Es Apache susceptible de un ataque DoS? ¿Cómo lo podríamos evitar?**
 - Es un ataque de denegación de servicio (Denial Of Service), que consiste en la creación de múltiples conexiones simultáneas al mismo servidor o dominio, provocando que el mismo se sature y deje de estar disponible.
- **¿Cuáles son las diferentes tablas de Iptables? ¿Cómo se acepta o deniega un servicio concreto (por ejemplo FTP)? ¿Cuáles son los parámetros más comunes en Iptables?**
 -

Ejercicio 1

- a) **¿Qué tendríamos que hacer para que el usuario “troll” no pueda ejecutar binarios ni scripts?**

Establecer al usuario troll en un grupo diferente en el que estén todos aquellos usuarios a los que no queramos permitir ejecutar binarios.

Después establecer los límites de este grupo en */etc/limits.conf* como *nproc* igual a 0 y *hard*, lo que hará que no pueda ejecutar ninguno proceso, por lo que no podría ejecutar ningún proceso ni script.

- b) **¿Qué haríamos si también quiere llenar el directorio */tmp*?**

Estableceremos un límite en */etc/limits.conf* para el espacio de usuario disponible para ese usuario.

- c) **En relación a la red:**

- i) **Queremos saber qué conexiones hay establecidas a nuestros sistemas y qué puertos se están utilizando. ¿Qué comando se debería utilizar?**

Podemos hacer un *nmap* que escanee las conexiones establecidas en la interfaz en la cual esté conectada la máquina, en el caso de A podemos hacer *nmap -v 10.10.10.53*.

PORT	STATE	SERVICE
22/tcp	open	ssh
53/tcp	open	domain
80/tcp	open	http
111/tcp	open	rpcbind
443/tcp	open	https
2049/tcp	open	nfs

Como se puede ver, tenemos abiertos diferentes puertos en nuestra máquina, que concuerdan con lo que hemos tenido que ir haciendo a lo largo de las prácticas, como NFS o SSH.

- ii) **Se cree que hay un ordenador dentro de la red local que tiene algunas conexiones sospechosas. ¿Cómo se debe explorar con *nmap* esta IP?**

Debemos escanear individualmente con *nmap -v -A ip*, que nos indicará más información sobre la máquina asociada a esa IP, como el SO que utiliza, si está ejecutando algún script, etc, tal como se encuentra en la documentación de *nmap*.

```

Initiating NSE at 17:51
Completed NSE at 17:51, 0.00s elapsed
Nmap scan report for slave1m.gax.org (20.20.20.57)
Host is up (0.00060s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.4p1 Debian 10+deb9u7 (protocol 2.0)
| ssh-hostkey:
|   2048 e8:0a:b6:09:0e:09:3f:71:da:36:24:0c:bf:83:6a:5b (RSA)
|   256  b1:c6:25:07:b3:02:65:a3:3d:a2:14:bc:15:5b:3f:06 (ECDSA)
80/tcp    open  http      Apache httpd 2.4.25 ((Debian))
| http-methods:
|   Supported Methods: GET HEAD POST OPTIONS
|_ http-server-header: Apache/2.4.25 (Debian)
|_ http-title: Apache2 en la maquina B: It works
111/tcp   open  rpcbind  2-4 (RPC #100000)
| rpcinfo:
|   program version  port/proto  service
|   100000   2,3,4      111/tcp     rpcbind
|   100000   2,3,4      111/udp     rpcbind
MAC Address: 02:00:14:14:14:39 (Unknown)
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.6
Uptime guess: 0.010 days (since Sat Nov 28 17:36:49 2020)
Network Distance: 1 hop
TCP Sequence Prediction: Difficulty=253 (Good luck!)
IP ID Sequence Generation: All zeros
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE
HOP RTT      ADDRESS
1   0.60 ms  slave1m.gax.org (20.20.20.57)

```

- iii) Un usuario mirará de crear errores en Apache para llenar sus logs. ¿Cómo se puede evitar que esta acción también sature el sistema?

Antes de empezar, hacemos una captura de pantalla del estado actual del directorio de logs de Apache, para poder comparar después de ejecutar el script que generará los errores.

```

root@master-1-8:~# ls -l -h /var/log/apache2/
total 1.2M
-rw-r----- 1 root adm      0 Nov 18 17:45 access.log
-rw-r----- 1 root adm  7.0K Nov 12 18:57 access.log.1
-rw-r----- 1 root adm  307 Nov 10 22:28 access.log.2.gz
-rw-r----- 1 root adm  288 Nov  9 17:38 access.log.3.gz
-rw-r----- 1 root adm  280 Nov  8 11:33 access.log.4.gz
-rw-r----- 1 root adm  404 Nov  7 11:47 access.log.5.gz
-rw-r----- 1 root adm  6.1K Nov 28 17:30 error.log
-rw-r----- 1 root adm  4.8K Nov 28 13:05 error.log.1
-rw-r----- 1 root adm  909 Nov 27 18:30 error.log.2.gz
-rw-r----- 1 root adm  868 Nov 26 20:50 error.log.3.gz
-rw-r----- 1 root adm  3.4K Nov 18 17:45 error.log.4.gz
-rw-r----- 1 root adm  940 Nov 12 10:07 error.log.5.gz
-rw-r----- 1 root adm  948 Nov 10 21:37 error.log.6.gz
-rw-r----- 1 root adm  891 Nov  9 13:33 error.log.7.gz
-rw-r----- 1 root adm  565 Nov  8 10:52 error.log.8.gz

```

Como se puede ver, ya se han hecho varias rotaciones en los logs, y antes de realizar nada el directorio tiene un tamaño total de 1.2 MB. Debido a un error en la captura de pantalla y no poder repetirla, falta mencionar un log que será importante una vez ejecutado el script, y este es el *other_vhosts_access.log* que ocupa 1.1 MB.

A continuación ejecutamos el script que creará las peticiones falsas y por lo tanto incrementará el tamaño de los logs,

cambiando 'ip' por la de la máquina donde ejecutaremos el script, en nuestro caso será 10.10.10.53 (máquina A).

```
for n in `seq 1 20000`  
do  
    wget http://IP/$n  
done
```

El script generará peticiones similares a la siguiente, pidiendo descargar el html de la misma, no pudiendo hacerlo y por consecuencia generando un error.

```
--2020-11-28 18:14:31-- http://10.10.10.53/20000  
Connecting to 10.10.10.53:80... connected.  
HTTP request sent, awaiting response... 404 Not Found  
2020-11-28 18:14:31 ERROR 404: Not Found.
```

Esto ha generado que el tamaño del directorio haya aumentado debido a la gran cantidad de peticiones falsas.

```
root@master-1-8:~# ls -l -h /var/log/apache2/  
total 3.4M  
-rw-r----- 1 root adm      0 Nov 18 17:45 access.log  
-rw-r----- 1 root adm   7.0K Nov 12 18:57 access.log.1  
-rw-r----- 1 root adm  307 Nov 10 22:28 access.log.2.gz  
-rw-r----- 1 root adm  288 Nov  9 17:38 access.log.3.gz  
-rw-r----- 1 root adm  280 Nov  8 11:33 access.log.4.gz  
-rw-r----- 1 root adm  404 Nov  7 11:47 access.log.5.gz  
-rw-r----- 1 root adm   6.1K Nov 28 17:30 error.log  
-rw-r----- 1 root adm   4.8K Nov 28 13:05 error.log.1  
-rw-r----- 1 root adm   909 Nov 27 18:30 error.log.2.gz  
-rw-r----- 1 root adm   868 Nov 26 20:50 error.log.3.gz  
-rw-r----- 1 root adm   3.4K Nov 18 17:45 error.log.4.gz  
-rw-r----- 1 root adm   940 Nov 12 10:07 error.log.5.gz  
-rw-r----- 1 root adm   948 Nov 10 21:37 error.log.6.gz  
-rw-r----- 1 root adm   891 Nov  9 13:33 error.log.7.gz  
-rw-r----- 1 root adm   565 Nov  8 10:52 error.log.8.gz  
-rw-r----- 1 root adm  2.3M Nov 28 18:14 other_vhosts_access.log  
-rw-r----- 1 root adm  1.1M Nov 12 22:06 other_vhosts_access.log.1  
-rw-r--r-- 1 root root      0 Nov 12 19:13 proxy-access.log  
-rw-r--r-- 1 root root      0 Nov 12 19:13 proxy-error.log
```

También se ha generado un nuevo fichero de log de *other_vhosts_access.log* que sustituye al anterior y ocupando algo más del doble que el original.

Echando un vistazo a ese fichero, podemos ver que se trata de las peticiones *wget* realizadas con el script.

```
127.0.0.1:80 10.10.10.53 - - [28/Nov/2020:18:14:31 +0000] "GET /19996 HTTP/1.1" 404 490 "-" "Wget/1.18 (linux-gnu)"  
127.0.0.1:80 10.10.10.53 - - [28/Nov/2020:18:14:31 +0000] "GET /19997 HTTP/1.1" 404 490 "-" "Wget/1.18 (linux-gnu)"  
127.0.0.1:80 10.10.10.53 - - [28/Nov/2020:18:14:31 +0000] "GET /19998 HTTP/1.1" 404 490 "-" "Wget/1.18 (linux-gnu)"
```

En él podemos ver la IP desde donde se ha realizado la petición y la IP de nuestra máquina, incluyendo la fecha y hora en la cual se ha realizado junto con el tipo de petición.

- iv) Desde fuera se ha preparado un ataque de denegación de servicio (DoS) en Apache. Apache tiene herramientas para evitar ser susceptible a estos ataques. Describe qué lógica se sigue para evitar que dejen fuera de servicio el servidor.

1)

- v) ¿Cómo se puede deshabilitar todo tipo de conexiones para una IP concreta, y para el servicio de SSH desde otra IP?

Debemos modificar el archivo *hosts.deny*, incluyendo la IP o IPs en las cuales queremos desactivar todas las conexiones.

```
# /etc/hosts.deny: list of hosts that are _not_ allowed to access the system.
# See the manual pages hosts_access(5) and hosts_options(5).
#
# Example:  ALL: some.host.name, .some.domain
#          ALL EXCEPT in.fingerd: other.host.name, .other.domain
#
# If you're going to protect the portmapper use the name "rpcbind" for the
# daemon name. See rpcbind(8) and rpc.mountd(8) for further information.
#
# The PARANOID wildcard matches any host whose name does not match its
# address.
#
# You may wish to enable this to ensure any programs that don't
# validate looked up hostnames still leave understandable logs. In past
# versions of Debian this has been the default.
# ALL: PARANOID

ALL: 10.10.11.7
sshd: 30.30.30.52
```

Tal y como tenemos el fichero, estaríamos denegando todo tipo de conexiones a la máquina A desde D, lo que implica también perder acceso al servidor de Apache, por lo que no podremos visualizar la página desde D.

También estamos negando el acceso por SSH a la máquina A desde C, sin importar a que interfaz de A intente conectarse.

```
root@slave2-1-8:~# ssh root@masterI
ssh_exchange_identification: read: Connection reset by peer
root@slave2-1-8:~# ssh root@masterM
ssh_exchange_identification: read: Connection reset by peer
```

- vi) Después de ver que se están realizando ataques desde fuera debemos restringir el acceso a nuestra máquina con reglas de iptables que realicen las siguientes restricciones:

- 1) No contestar mensajes de ICMP.

Para ello debemos configurar una regla con *iptables* que evite que se pueda hacer un ping a otra IP, por ejemplo. Haciendo *iptables -A OUTPUT -p icmp --icmp-type echo-request -j DROP*. Esto hará que todos los mensajes ICMP que quieran salir de la máquina serán dropeados, pero quedará registrado conforme se ha hecho la petición.

```
root@slave3-1-8:~# iptables -A OUTPUT -p icmp --icmp-type echo-request -j DROP
root@slave3-1-8:~# ping 10.10.10.53
PING 10.10.10.53 (10.10.10.53) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
^C
--- 10.10.10.53 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2065ms
```


2) Bloquear usuarios internos la salida a twitter.com facebook.com y youtube.com.

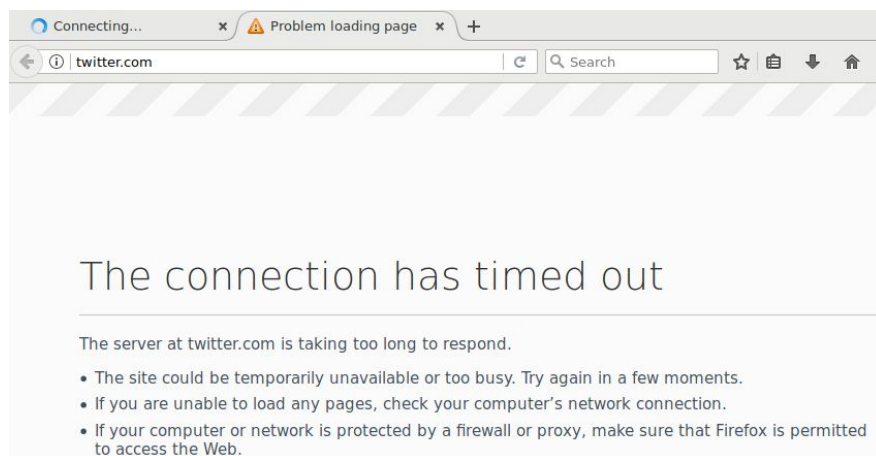
Deberemos añadir una regla similar a la anterior, pero en este caso indicando los dominios a los que queremos bloquear el acceso.

Para ello será tan simple como hacer lo siguiente:

```
iptables -A OUTPUT -d dominio.com -j DROP
```

Que evitará que los paquetes con destino a alguno de los dominios indicados salgan de la máquina.

```
root@slave3-1-8:~# iptables -A OUTPUT -d facebook.com -j DROP
root@slave3-1-8:~# iptables -A OUTPUT -d twitter.com -j DROP
root@slave3-1-8:~# iptables -A OUTPUT -d youtube.com -j DROP
```



3) Que no hayan más de tres conexiones simultáneas por SSH.

Volveremos a hacer una regla de iptables similar en este caso limitando las conexiones que entran en el puerto 22 del protocolo TCP de la máquina, que es el puerto utilizado para establecer las conexiones por SSH, por lo que quedaría una regla así.

```
root@master-1-8:~# iptables -A INPUT -p tcp --syn --dport 22 -m connlimit --connlimit-above 3 -j REJECT
```

Podemos variar el número de conexiones que queremos permitir cambiando el número que va después de *connlimit-above* al número deseado.

```
root@slave1-1-8:~# ssh root@master1
ssh: connect to host master1 port 22: Connection refused
```

Esto hará que después se hayan establecido tres conexiones por SSH se rechacen los siguientes intentos de realizar la conexión.

4) Evitar que algunas IPs concretas se conecten. Mirar rangos de IPs que se dan a un país o conjunto de países y no permitir el acceso.

Deberíamos crear otra regla con *iptables* de forma que rechace todas las conexiones provenientes de un cierto país o países. Ya que sabemos que cada país tiene asignado un rango de IPs. Por ejemplo, para bloquear todas las IPs asociadas con la ISP Orange en Francia, sabiendo que estas IPs van de 2.0.0.0 a 2.15.255.255, la regla sería la siguiente:

```
iptables -A INPUT -p tcp -s 2.0.0.0/12 -j DROP
```

d) Reconfigurar las MV de forma que C esté conectada haciendo uso de la red 20.20.20/21.0 y crear sobre B una DMZ donde A será el Firewall que controlará el acceso. Todas las MV deberán de tener Apache2 y OpenSSH-server.

Para conseguir que la máquina C se conecte con la red 20.20.20.0/24 hemos tenido que añadir la interfaz de red Middle en OpenNebula y después eliminado la interfaz Deep. De esta forma nos aseguramos que se está conectando desde la red 20.20.20.0/24.

Una vez dentro de la máquina C, hemos modificado el archivo */etc/network/interfaces* para actualizarlo a los nuevos parámetros de red, para que en esta ocasión utilice la máquina B como gateway.

```
auto lo
iface lo inet loopback

auto ens3
iface ens3 inet static
address 20.20.20.200
network 20.20.20.0
netmask 255.255.255.0
gateway 20.20.20.44

source /etc/network/interfaces.d/*.cfg
```

Antes de continuar debemos configurar la nueva interfaz Middle añadida a la máquina A, que servirá de para conectar con la DMZ en la red 20.20.25.0/24.

```
auto ens5
iface ens5 inet static
address 20.20.25.214
network 20.20.25.0
netmask 255.255.255.0
```

Cambiamos también la configuración de red de la máquina B para que se conecte con el firewall a través de la red 20.20.25.0/24.

```
auto ens3
iface ens3 inet static
address 20.20.25.57
network 20.20.25.0
netmask 255.255.254.0
gateway 20.20.25.214
```

Hemos intentado realizar los apartados que se especificaban, pero por alguna razón no hemos conseguido realizarlos. A nuestro entender para que la máquina D pudiese conectar a B, primero pasaría por la interfaz de Internet de la máquina A, con la que comparte red.

Para ello hemos añadido reglas de iptables para que aceptar los paquetes provenientes de 10.10.11.7 (IP de la máquina D) y encaminarlos desde 10.10.10.53 hasta 20.20.20.44 (IPs de las interfaces de Internet y Middle respectivamente de la máquina A). Para ello hemos creado un script para no perder los cambios.

```
#!/bin/sh
# FLUSH
iptables -F
iptables -X
iptables -Z

# ACEPTAMOS CONEXIONES AL FIREWALL DE PARTE LA LAN
iptables -A INPUT -s 10.10.11.7 -i ens3 -j ACCEPT
#iptables -A INPUT -s 20.20.20.0/23 -i ens4 -j ACCEPT

# ENMASCARAMIENTO DMZ Y LAN
#iptables -t nat -A POSTROUTING -s 20.20.20.0/23 -i ens3 -j MASQUERADE
#iptables -t nat -A POSTROUTING -s 20.20.25.0/24 -o ens5 -j MASQUERADE

# DAMOS ACCESO A LA MAQUINA D PARA APACHE
iptables -A FORWARD -s 10.10.11.7 -d 10.10.10.53 -j ACCEPT
iptables -A FORWARD -s 10.10.10.53 -d 10.10.11.7 -j ACCEPT
iptables -A FORWARD -s 10.10.10.53 -d 20.20.20.44 -j ACCEPT
iptables -A FORWARD -s 20.20.20.44 -d 10.10.10.53 -j ACCEPT

# CERRAMOS ACCESOS EXTERIORES A LA DMZ
#iptables -A INPUT -s 0.0.0.0/0 -p tcp --dport 1:1024 -j DROP
#iptables -A INPUT -s 0.0.0.0/0 -p udp --dport 1:1024 -j DROP

# CERRAMOS PUERTO DE GESTION
#iptables -A INPUT -s 0.0.0.0/0 -p tcp --dport 10000 -j DROP
```

Para ello comenzamos haciendo una limpieza de las reglas anteriores como se puede ver en la captura, para después hacer las reglas de FORWARD e INPUT necesarias para encaminar los paquetes provenientes de D. Con lo que obtenemos las siguientes reglas.

```
root@master-1-8:~# iptables -L -n
Chain INPUT (policy ACCEPT)
target    prot opt source                destination
ACCEPT    all  --  10.10.11.7             0.0.0.0/0

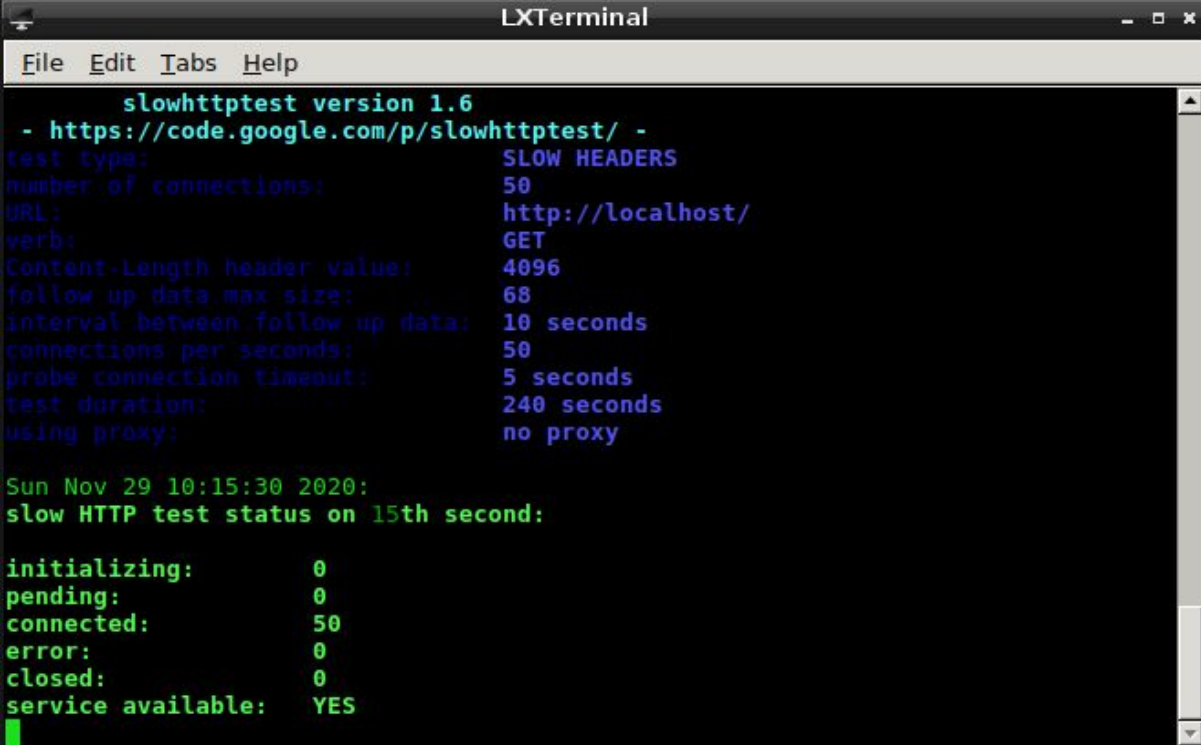
Chain FORWARD (policy ACCEPT)
target    prot opt source                destination
ACCEPT    all  --  10.10.11.7             10.10.10.53
ACCEPT    all  --  10.10.10.53            10.10.11.7
ACCEPT    all  --  10.10.10.53            20.20.20.44
ACCEPT    all  --  20.20.20.44            10.10.10.53

Chain OUTPUT (policy ACCEPT)
target    prot opt source                destination
```

Pero por alguna razón no ha sido suficiente y no hemos conseguido superar este apartado.

Ejercicio 2

Para hacer pruebas de un ataque DoS, primero hemos de instalar el paquete de *slowhttptest* con los comandos *apt-get update* y *apt-get install slowhttptest*. Después de esto podemos comprobar el estado de nuestra página usando el comando *slowhttptest -g* que nos enseñara la información siguiente.



```
LXTerminal
File Edit Tabs Help

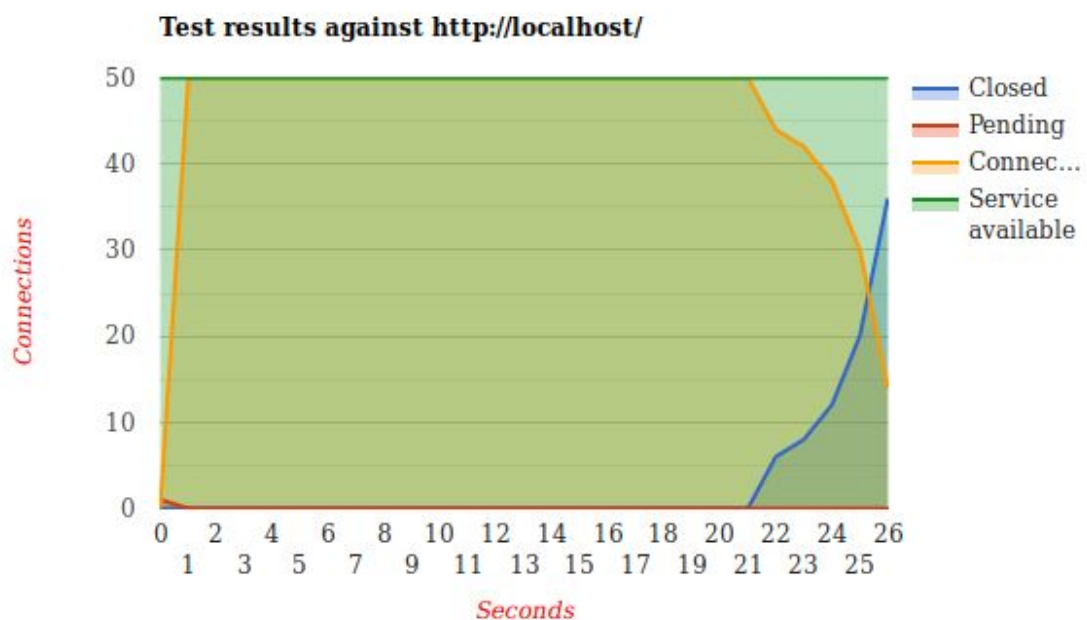
slowhttptest version 1.6
- https://code.google.com/p/slowhttptest/ -
test type:                SLOW HEADERS
number of connections:    50
URL:                      http://localhost/
verb:                     GET
Content-Length header value: 4096
follow up data max size:  68
interval between follow up data: 10 seconds
connections per seconds:  50
probe connection timeout: 5 seconds
test duration:            240 seconds
using proxy:              no proxy

Sun Nov 29 10:15:30 2020:
slow HTTP test status on 15th second:

initializing:             0
pending:                  0
connected:                50
error:                    0
closed:                   0
service available:        YES
```

También se crea un archivo html que podemos abrir y que nos enseñara una gráfica que indica que el servicio ha estado disponible durante todo el tiempo.

Test parameters	
Test type	SLOW HEADERS
Number of connections	50
Verb	GET
Content-Length header value	4096
Extra data max length	68
Interval between follow up data	10 seconds
Connections per seconds	50
Timeout for probe connection	5
Target test duration	240 seconds
Using proxy	no proxy



Ahora haremos una prueba contra nuestro Apache en la máquina B. Usaremos el comando siguiente `slowhttptest -g -c 10000 -u http://20.20.20.x -l 1000` que realizara conexiones simultáneas al Apache en nuestra máquina B y como se puede ver en las siguientes imágenes, esto hará que nuestro servicio deje de funcionar momentáneamente.

```

Sun Nov 29 14:44:33 2020:
slowhttpptest version 1.6
- https://code.google.com/p/slowhttpptest/ -
test type:                SLOW HEADERS
number of connections:    10000
URL:                      http://20.20.20.57/
verb:                     GET
content-length header value: 4096
follow up data max size:  68
interval between follow up data: 10 seconds
connections per seconds:  50
probe connection timeout:  5 seconds
test duration:            1000 seconds
using proxy:              no proxy

```

```

Sun Nov 29 14:44:33 2020:
slow HTTP test status on 5th second:

```

```

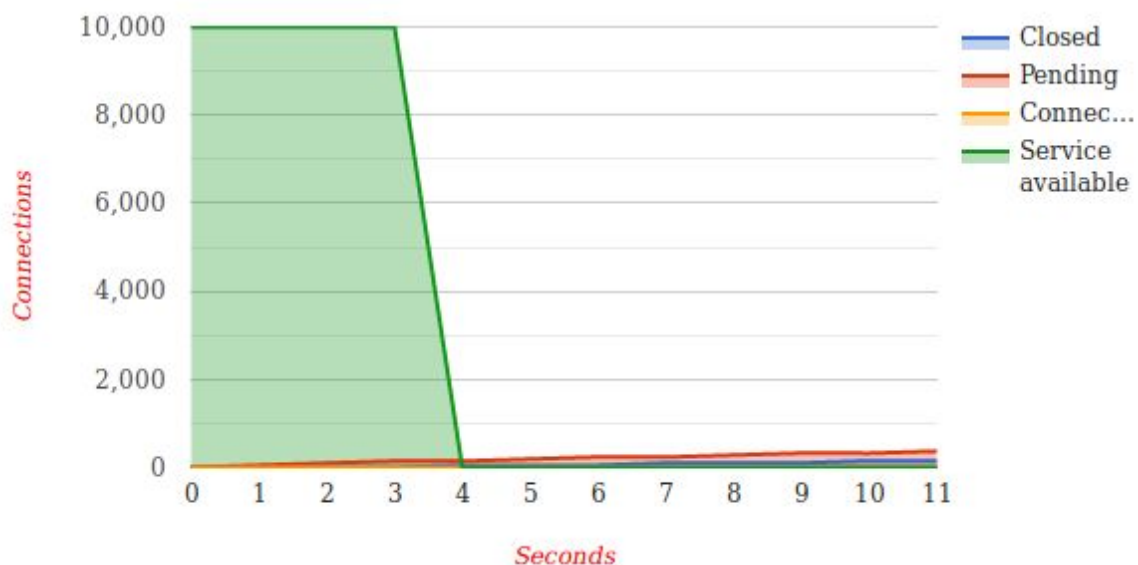
initializing:      0
pending:           190
connected:         0
error:             0
closed:            50
service available: NO

```

Test parameters

Test type	SLOW HEADERS
Number of connections	10000
Verb	GET
Content-Length header value	4096
Extra data max length	68
Interval between follow up data	10 seconds
Connections per seconds	50
Timeout for probe connection	5
Target test duration	1000 seconds
Using proxy	no proxy

Test results against http://20.20.20.57/



Ejercicio 3

Una de las formas de prevenir un ataque como el mostrado anteriormente es usando un Proxy Balancer que divide las conexiones entre dos nodos diferentes. Para realizar instalaremos Apache en el nodo C que es el que balancearemos junto con B las conexiones de D hacia A. Cargamos los módulos de balancer, `lbmethod_bytraffic` y `lbmethod_byrequest` que son con los que crearemos nuestro Proxy Balancer. Para esto usamos los comandos `a2enmodule proxy_balancer`, `a2enmodule lbmethod_byrequest` y `a2enmodule lbmethod_byttraffic`.

```
root@master-1-8:/etc/apache2# a2enmod proxy_balancer
Considering dependency proxy for proxy_balancer:
Module proxy already enabled
Considering dependency alias for proxy_balancer:
Module alias already enabled
Considering dependency slotmem_shm for proxy_balancer:
Enabling module slotmem_shm.
Enabling module proxy_balancer.
To activate the new configuration, you need to run:
    systemctl restart apache2
root@master-1-8:/etc/apache2# a2enmod lbmethod_byrequests
Considering dependency proxy_balancer for lbmethod_byrequests:
Considering dependency proxy for proxy_balancer:
Module proxy already enabled
Considering dependency alias for proxy_balancer:
Module alias already enabled
Considering dependency slotmem_shm for proxy_balancer:
Module slotmem_shm already enabled
Module proxy_balancer already enabled
Enabling module lbmethod_byrequests.
To activate the new configuration, you need to run:
    systemctl restart apache2
root@master-1-8:/etc/apache2# a2enmod lbmethod_bytraffic
Considering dependency proxy_balancer for lbmethod_bytraffic:
Considering dependency proxy for proxy_balancer:
Module proxy already enabled
Considering dependency alias for proxy_balancer:
Module alias already enabled
Considering dependency slotmem_shm for proxy_balancer:
Module slotmem_shm already enabled
Module proxy_balancer already enabled
Enabling module lbmethod_bytraffic.
To activate the new configuration, you need to run:
    systemctl restart apache2
root@master-1-8:/etc/apache2#
```

Después de esto vamos a crear un nuevo archivo proxy.conf que lo definiremos de la siguiente manera.

```
<VirtualHost proxy.gax.org:80>
ProxyRequests off
ServerName proxy.gax.org
DocumentRoot /var/www/html

<Proxy balancer://mycluster>
BalancerMember http://20.20.20.57:80
BalancerMember http://20.20.20.200:80
Options Indexes FollowSymlinks Multiviews
AllowOverride None
Order Allow,Deny
Allow from all
ProxySet lbmethod=bytraffic
#ProxySet lbmethod=byrequests
</Proxy>

<Location /balancer-manager>
SetHandler balancer-manager
Order deny,allow
Allow from all
</Location>

ProxyPass /balancer-manager !
ProxyPass / balancer://mycluster/
ProxyPassReverse / balancer://mycluster
ProxyPass / http://20.20.20.57
ProxyPassReverse / http://20.20.20.57
ProxyPass / http://20.20.20.200
ProxyPassReverse / http://20.20.20.200

</VirtualHost>
```

Ahora al conectarnos a proxy.gax.org desde la máquina D, nos dirige al nodo B o al nodo C mitigando así el ataque del slowhttptest.

Ejercicio 4

Primero instalamos el repositorio de medusa usando `apt install medusa` en la máquina A. Crearemos un nuevo usuario llamado prueba con la contraseña password y también crearemos el archivo passwords.txt donde estarán unas cuantas contraseñas que usará el programa para intentar acceder al usuario prueba. Usaremos el siguiente comando para realizar el ataque.

```
root@master-1-8:~# medusa -u prueba -P passwords.txt -h 10.10.10.53 -M ssh
Medusa v2.2 [http://www.foofus.net] (C) JoMo-Kun / Foofus Networks <jmk@foofus.net>

ACCOUNT CHECK: [ssh] Host: 10.10.10.53 (1 of 1, 0 complete) User: prueba (1 of 1, 0 complete) Password: 12345678 (1 of 5 complete)
ACCOUNT CHECK: [ssh] Host: 10.10.10.53 (1 of 1, 0 complete) User: prueba (1 of 1, 0 complete) Password: password (2 of 5 complete)
ACCOUNT FOUND: [ssh] Host: 10.10.10.53 User: prueba Password: password [SUCCESS]
root@master-1-8:~#
```

Como se puede observar, el ataque ha sido realizado con éxito.

Para minimizar este tipo de ataques usaremos el fail2ban. Lo instalaremos con el comando `apt install fail2ban`. Dentro del directorio `/etc/fail2ban/` podemos modificar el archivo `jail.conf` como se puede ver en la siguiente imagen.

```
#
# MISCELLANEOUS OPTIONS
#
# "ignoreip" can be an IP address, a CIDR mask or a DNS host. Fail2ban will not
# ban a host which matches an address in this list. Several addresses can be
# defined using space (and/or comma) separator.
ignoreip = 127.0.0.1/8

# External command that will take an tagged arguments to ignore, e.g. <ip>,
# and return true if the IP is to be ignored. False otherwise.
# ignorecommand = /path/to/command <ip>
ignorecommand =

# "bantime" is the number of seconds that a host is banned.
bantime = 10m

# A host is banned if it has generated "maxretry" during the last "findtime"
# seconds.
findtime = 60m

# "maxretry" is the number of failures before a host get banned.
maxretry = 1

# "backend" specifies the backend used to get files modification.
# Available options are "pyinotify", "gamin", "polling", "systemd" and "auto".
# This option can be overridden in each jail as well.
#
# pyinotify: requires pyinotify (a file alteration monitor) to be installed.
#            If pyinotify is not installed, Fail2ban will use auto.
# gamin:     requires Gamin (a file alteration monitor) to be installed.
#            If Gamin is not installed, Fail2ban will use auto.
# polling:   uses a polling algorithm which does not require external libraries.
# systemd:   uses systemd python library to access the systemd journal.
#            Specifying "logpath" is not valid for this backend.
#            See "journalmatch" in the jails associated filter config
# auto:      will try to use the following backends, in order:
#            pyinotify, gamin, polling.
#
# Note: if systemd backend is chosen as the default but you enable a jail
```

Modificando este archivo podemos cambiar la configuración para poder elegir cuantas veces se podrá intentar entrar a un usuario, cuánto tiempo estará baneado si la contraseña es introducida incorrectamente, cuanto tiempo entre intentos, etc