Ingeniería Informática

Práctica 4 GAX

Maksym Lakhmanets - 1495282
Carles Costas Mateu - 1491578
02/12/2021

Treball previ: Contestar a les següents preguntes (es pot deixar com treball de grup en casa i que s'han d'incloure en un apartat al final de l'informe).

• Quina ordre ens permet veure els límits del usuari de processos, file descriptors, memòria, etc?

more /etc/security/limits.conf

• Si els arxius de log del sistema comencen a créixer indefinidament, com evitaríem que ens consumeixi tot l'espai del disc?

Podemos utilizar el comando logrotate para eliminar y/o comprimir los logs.

• Com podem evitar que un usuari concret pugui deixar d'executar un binari que es troba a /usr/bin?

Podemos denegar la capacidad de ejecución de archivos en la carpeta /usr/bin utilizando el comando chmod o-x

• Què és un atac de DoS? És Apache susceptible d'un atac de DoS? Com el podríem evitar?

Un ataque de DoS (ataque de denegación del servicio) ocurre cuando una máquina A hace muchas peticiones de un servicio a un servidor B, provocando el colapso de éste, haciendo que se sature por no poder satisfacer todas las peticiones.

Apache es susceptible de este tipo de ataques pero podemos evitarlos modificando el archivo host.deny para denegar servicios a las IPs que especifiquemos.

• Quines són les diferents taules de Iptables? Com s'accepta o denega un servei concret (per exemple SSH)? Quins són els paràmetres més comuns a Iptables?

Las diferentes tablas que tiene iptables son: Filter, nat, mangle y raw.
Para aceptar o denegar un servicio en concreto podemos utilizar los siguientes comandos:

iptables -A INPUT -p tcp --dport 3306 -j ACCEPT
iptables -A INPUT -p tcp --dport 3306 -j DROP

Parámetros importantes en iptables:
-p: Indicamos el protocolo.
-s: Especificamos la dirección de origen.
-d: Especificamos la dirección de destino.
-i: interfaz por la que se reciben los paquetes.
-o: interfaz por la que salen los paquetes.
-dport: Especificamos el puerto de destino.
-j: Especificamos si se aceptará el servicio o se denegará.

Excercici 1

Crearemos un nuevo usuario troll:

```
root@debian:/# useradd -m troll
root@debian:/# passwd troll
New password:
Retype new password:
passwd: password updated successfully
```

a) Què hauríem de fer per a que l'usuari "troll" no pugui executar binaris ni scripts.

Posicionándonos en la carpeta /home/troll y ejecutando el comando chmod o-x * para quitarle el permiso de ejecución en todos los directorios.

b) Què faríem si també vol omplir el directori /tmp?

Podemos limitar el espacio que el usuario podrá usar, en el archivo etc/security/limits.conf:

```
#*               soft    core            0
#root            hard    core            100000
#*               hard    rss             10000
#@student        hard    nproc           20
#@faculty        soft    nproc           20
#@faculty        hard    nproc           50
#ftp             hard    nproc           0
#ftp             -       chroot          /ftp
#@student        -       maxlogins       4
@troll           hard    as              1000
```

Solo podrá usar 1000 KB.

<u>El usuario troll hará lo siguiente:</u>

- Utilitzar tota la memòria del sistema.

Solucionado en el apartado b.

- Crear un fork bomb i paralitzar la CPU.

El usuario troll procederá a con el comando :() { :|:& };: pero antes deberemos volver a editar el archivo limits.conf para limitar el número de procesos que puedo ejecutar el usuario troll:

```
#*              soft    core            0
#root           hard    core            100000
#*              hard    rss             10000
#@student       hard    nproc           20
#@faculty       soft    nproc           20
#@faculty       hard    nproc           50
#ftp            hard    nproc           0
#ftp            -       chroot          /ftp
#@student       -       maxlogins       4
@troll          hard    as              1000
@troll          hard    nproc           10
```

Vamos a establecer que no pueda ejecutar más de 10 procesos con nproc.

- Obrirà tants file descriptors que no tindrem espai per més.

```
#*              soft    core            0
#root           hard    core            100000
#*              hard    rss             10000
#@student       hard    nproc           20
#@faculty       soft    nproc           20
#@faculty       hard    nproc           50
#ftp            hard    nproc           0
#ftp            -       chroot          /ftp
#@student       -       maxlogins       4
@troll          hard    as              1000
@troll          hard    nproc           10
@troll          hard    nofiles         2
```

Con nofiles especificamos que no pueda abrir más de 2 archivos (puede generar posibles errores del sistema).

- En el seu directori /home/troll crearà molts arxius grans per a bloquejar l'espai del sistema de fitxers.

```
#*              soft    core            0
#root           hard    core            100000
#*              hard    rss             10000
#@student       hard    nproc           20
#@faculty       soft    nproc           20
#@faculty       hard    nproc           50
#ftp            hard    nproc           0
#ftp            -       chroot          /ftp
#@student       -       maxlogins       4
@troll          hard    as              1000
@troll          hard    nproc           10
@troll          hard    nofiles         2
@troll          hard    fsize           1
```

Con fsize especificamos que el tamaño máximo de archivo que puede crear es de 1 KB

- Començarà a omplir els logs del sistema per a fer-los créixer indefinidament i bloquejar l'espai del sistema de fitxers.

```
# see "man logrotate" for details

# global options do not affect preceding include directives

# rotate log files weekly
weekly

# keep 4 weeks worth of backlogs
rotate 4

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

# use date as a suffix of the rotated file
#dateext

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

# packages drop log rotation information into this directory
include /etc/logrotate.d
maxage 7
maxsize 100
```

Para evitar que el usuario troll llene los logs del sistema modificaremos el archivo /etc/logrotate.conf para restringir con los parámetros maxage y maxsize.

Para que los logs se eliminen automáticamente estableceremos un tiempo de 7 dias, y para que los logs no tengan un tamaño demasiado grande, estableceremos su límite a 100 kb con maxsize.

c) En relació a la xarxa:

c1. Volem saber quines connexions hi han establertes al nostre sistema i quins ports s'estan utilitzant. Quina comanda s'hauria d'utilitzar?

Con el comando netstat podremos conocer que conexions y que puertos se estan usando;

```
adminp@debian:~$ netstat -a
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 sysetet.gax.org:42066   36.75.98.34.bc.go:https ESTABLISHED
tcp        0      0 sysetet.gax.org:47622   mad41s11-in-f2.1e:https ESTABLISHED
tcp        0      0 sysetet.gax.org:52786   mil04s44-in-f3.1e1:http ESTABLISHED
tcp        0      0 sysetet.gax.org:35926   server-52-85-3-76:https TIME_WAIT
tcp        0      0 sysetet.gax.org:53792   mad07s24-in-f2.1e:https ESTABLISHED
tcp        0      0 sysetet.gax.org:47628   mad41s11-in-f2.1e:https TIME_WAIT
```

c2. Es creu que hi ha un ordinador dintre de la xarxa local que té algunes connexions sospitoses. Com s'ha d'explorar amb nmap aquesta IP?

Ejecutamos *nmap -A 172.16.1.1* , que es la IP del ordenador sospechoso (mvB).

```
root@slave1:~# nmap -A 172.16.1.1
Starting Nmap 7.80 ( https://nmap.org ) at 2021-12-02 19:34 CET
Nmap scan report for master.gax.org (172.16.1.1)
Host is up (0.0043s latency).
Not shown: 994 closed ports
PORT     STATE    SERVICE  VERSION
22/tcp   open     ssh      OpenSSH 8.4p1 Debian 5 (protocol 2.0)
53/tcp   open     domain   dnsmasq 2.85
| dns-nsid:
|_  bind.version: dnsmasq-2.85
80/tcp   filtered http
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
|   100000  3,4          111/tcp6  rpcbind
|   100000  3,4          111/udp6  rpcbind
|   100003  3           2049/udp   nfs
|   100003  3           2049/udp6  nfs
|   100003  3,4         2049/tcp   nfs
|   100003  3,4         2049/tcp6  nfs
|   100005  1,2,3      32877/udp   mountd
|   100005  1,2,3      38175/tcp   mountd
|   100005  1,2,3      46593/udp6  mountd
|   100005  1,2,3      54209/tcp6  mountd
|   100021  1,3,4      37797/tcp6  nlockmgr
|   100021  1,3,4      39577/udp   nlockmgr
|   100021  1,3,4      44901/tcp   nlockmgr
|   100021  1,3,4      55219/udp6  nlockmgr
|   100227  3           2049/tcp   nfs_acl
|   100227  3           2049/tcp6  nfs_acl
|   100227  3           2049/udp   nfs_acl
|_  100227  3           2049/udp6  nfs_acl
443/tcp  open     ssl/http Apache httpd 2.4.51 ((Debian))
|_http-server-header: Apache/2.4.51 (Debian)
|_http-title: 400 Bad Request
| ssl-cert: Subject: commonName=UAB/organizationName=UAB/stateOrProvinceName=UAB/countryName=ES
| Not valid before: 2021-10-01T19:30:00
|_Not valid after:  2022-11-10T19:30:00
| tls-alpn:
|_  http/1.1
2049/tcp open     nfs_acl  3 (RPC #100227)
MAC Address: 08:00:27:45:D6:83 (Oracle VirtualBox virtual NIC)
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=7.80%E=4%D=12/2%OT=22%CT=1%CU=33638%PV=Y%DS=1%DC=D%G=Y%M=080027%T
```

c3. Un usuari mirarà de crear errors a Apache per a omplir els seus logs. Com es pot evitar que aquesta acció també saturi el sistema?

Si accedemos a /etc/logrotate.d podem configurar (si o esta configurado ya) las opciones para los logs de apache:

```
adminp@debian:/etc/logrotate.d$ cat apache2
/var/log/apache2/*.log {
    daily
    missingok
    rotate 14
    compress
    delaycompress
    notifempty
    create 640 root adm
    sharedscripts
    prerotate
        if [ -d /etc/logrotate.d/httpd-prerotate ]; then
            run-parts /etc/logrotate.d/httpd-prerotate
        fi
    endscript
    postrotate
        if pgrep -f ^/usr/sbin/apache2 > /dev/null; then
            invoke-rc.d apache2 reload 2>&1 | logger -t apache2.logrotate
        fi
    endscript
}
```

Se podría seguir la misma configuración que en el apartado anterior, limitando el tamaño del log y el tiempo hasta su borrado.

c4. Des de fora ha preparat un atac de denegació de servei (DoS) a Apache. Apache té eines per evitar ser susceptible a aquests atacs. Descriu quina lògica es segueix per evitar que deixin fora de servei el servidor.

ModSecurity es un firewall de aplicaciones web de código abierto (WAF) diseñado como un módulo para servidores web Apache. ModSecurity proporciona un motor de reglas flexible, que permite a los usuarios escribir (o utilizar reglas de terceros) para proteger sitios web de ataques como XSS, SQLi, CSRF, DDoS e inicio de sesión por fuerza bruta (así como una serie de otras vulnerabilidades).

Para protegernos de un ataque DoS, podemos instalar el módulo apache-mod-security:

sudo apt-get install libxml2 libxml2-dev libxml2-utils
sudo apt-get install libaprutil1 libaprutil1-dev
sudo apt-get install libapache2-mod-security2

Una vez instalado, el módulo se activa solo por lo que no hará falta activarlo manualmente.

ModSecurity corre inicialmente en modo DetectionOnly, en el que WAF examina el tráfico HTTP(S), pero en realidad no bloquea las solicitudes maliciosas. Esto debe ajustarse para que ModSecurity denegue el tráfico de ataque. En el archivo /etc/modsecurity/modsecurity.conf, busca la directiva SecRuleEngine:

SecRuleEngine DetectionOnly
Y configura sus valores como On:

SecRuleEngine On
Y claramente, recarga Apache para efectuar los cambios:

[admin@debian]# service apache2 restart

c5. Com es pot deshabilitar tot tipus de connexions per a una IP concreta, i per al servei de SSH des de altre IP.

Modificaremos el archivo /etc/host.deny donde agregaremos la IP concreta que queramos deshabilitar.

ALL:_IPconcreta
SSH: sshd:IP_concreta

c6. Després de veure que s'estan realitzant atacs des de fora hem de restringir l'accés a la nostra màquina amb regles de iptables que realitzin les següents restriccions:

- No contestar missatges de ICMP.

Utilizaremos la regla de iptables:

iptables -A OUTPUT -p icmp --icmp-type echo-request -j DROP

- Bloquejar als usuaris interns la sortida a twitter.com facebook.com i youtube.com.

iptables -A OUTPUT -d twitter.com -j DROP
iptables -A OUTPUT -d facebook.com -j DROP
iptables -A OUTPUT -d youtube.com -j DROP

- No es vol que hi hagin més de tres connexions simultànies per SSH

iptables -A INPUT -p tcp --syn --dport 22 -m connlimit --connlimit-above 3 -j REJECT

- Evitar que algunes IPs concretes es connectin. Mirar rangs de ips que es donen a un país o conjunt de països i no permetre l'accés.

Ya que no nos gusta francia, bloquearemos las IP's de francia:

iptables -A INPUT -p tcp -s 5.49.1.0/24 -j DROP

d) Reconfigurar les MV (o esborrar i crear de noves) de forma A tingui tres adaptadors de xarxa i B,C solament un i en totes les MVs s'haurà de tenir Apache2 i l'Openssh-server (modificar els index.html per indicar desde quina màquina s'està carregant). C estarà connectada a una xarxa per exemple 172.16.1.0/24 i B estarà connectada A fent servir una nova xarxa (per exemple 172.16.66/0). S'haurà de crear sobre B una DMZ on A serà el Firewall que controlarà l'accés. Les accions que haurà de controlar A són:

i. Des d'una 4a MV (D) es podrà accedir a l'Apache instal·lat a B però res més (es a dir cap servei d'A ni de C).
ii. Des de C i A es podrà conectar a B tant a l'Apache com a ssh.
iii. Des de B NO és podrà connectar ni a A ni a C a cap servei.

Este ejercicio lo dejamos para al final y no he conseguido pasar las pruebas, aun asi dejamos la configuración de como deberían estar configuradas las máquinas:

Modificaremos el archivo etc/network/interfaces.d de la máquina A:

```
auto lo
iface lo inet loopback

auto enp0s3
iface enp0s3 inet static
        adress 20.20.20.24
        network 20.20.20.0
        netmask 255.255.254.0

auto enp0s3:0
iface enp0s3 inet static
        adress 20.20.20.27
        network 20.20.20.0
        netmask 255.255.254.0

auto enp0s5
iface enp0s5 inet static
        adress 10.10.10.39
        network 10.10.10.0
        netmask 255.255.254.0
        gateway 10.10.10.1


auto enp0s8
iface enp0s8 inet static
        adress 172.16.1.1
        network 172.16.1.0
        netmask 255.255.255.0
```

Indicamos la red 20.20.20.0 del host B y añadimos la interfaz de la DMZ (enp0s8) con una IP para que la DMZ sea fácilmente reconocible.

También configuraremos la máquina B (DMZ) de la siguiente manera:
r

```
auto lo
iface lo inet loopback

auto enp0s3
iface enp0s3 inet static
        adress 172.16.1.1
        network 172.16.1.0
        netmask 255.255.255.0
        gateway 172.168.1.10
```

Para que el tráfico se establezca siguiendo el esquema, añadiremos en la máquina A las siguientes reglas iptables:

Denegamos tráfico de la DMZ:

post-up iptables -t nat -A POSTROUTING -o enp0s5 -j MASQUERADE

post-up iptables -t nat -A FORWARD -o enp0s8 -m state –state NEW -j DROP
post-up iptables -t nat -A INPUT -o enp0s8 -m state –state NEW -j DROP

Redirigimos los paquetes que nos lleguen desde la B desde A y desde fuera

post-up iptables -t nat -A OUTPUT -p -tcp -d 20.20.20.27 -j DNAT –to-destination 172.16.1.1
post-up iptables -t nat -A PREROUTING -p -tcp -d 20.20.20.27 -j DNAT –to-destination 172.16.1.1

Exercici 2:

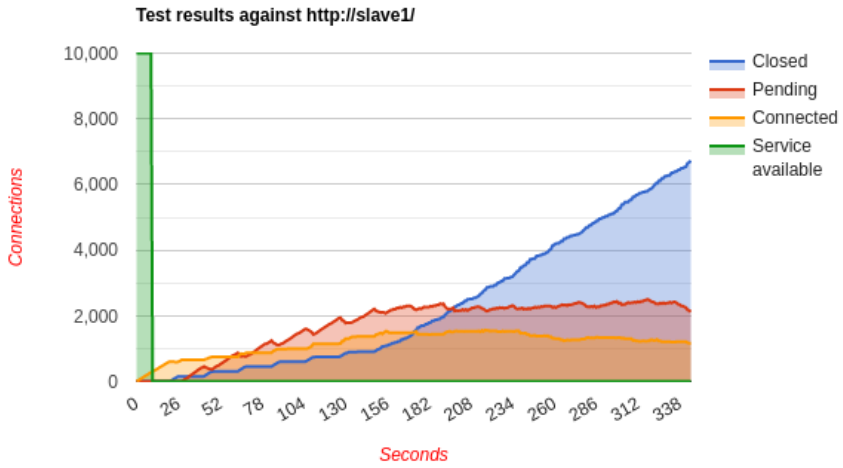
En este ejercicio se pretende proteger al servicio apache de posibles ataques DoS. Para realizar ataques DoS y poder probar que nuestra seguridad cumple unos mínimos, vamos a utilizar el paquete **slowhttptest** con el cual podremos realizar denegación de servició de la máquina A a la máquina B.

Primero vamos a analizar los diferentes parámetros con los que ejecutaremos slowhttptest, tal como mencionamos anteriormente, con la opción -g forzamos a que genere un csv y un fichero html al final del test con las estadísticas del ataque. Con la opción -c definiremos el número de conexiones que se usarán durante el test. Con la opción -l definiremos el tiempo que durará el test. Finalmente con -u especificaremos la url completa sobre la que vamos a realizar el test. En nuestro caso vamos a realizar un test en el servicio apache, sobre la máquina B con un número de 10000 conexiones durante 1000 segundos.

```
slowhttptest -g -c 10000 -u http://slave1/ -l 1000
```

El resultado del test es el tal como vemos en la captura, gracias a la opción -g nos genera un reporte, donde podemos ver que a partir de 700 conexiones a los 10 segundos aproximadamente se ha caído el servició. También podemos ver otra información como el tiempo time out por conexión que en este caso serían 5 segundos (valor por defecto), la largada del contenido extra de cada request, el método por el que se realizan las request en este caso GET y las conexiones por segundo que serian 50 conexiones por segundo.

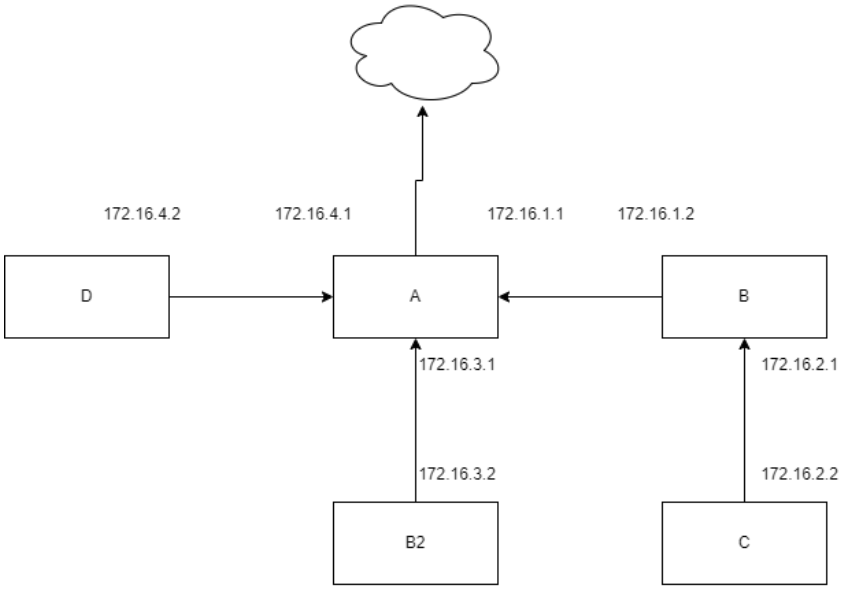
Test parameters	
Test type	SLOW HEADERS
Number of connections	10000
Verb	GET
Content-Length header value	4096
Cookie	
Extra data max length	68
Interval between follow up data	10 seconds
Connections per seconds	50
Timeout for probe connection	5
Target test duration	1000 seconds
Using proxy	no proxy



Slowhttptest, nos proporciona otros parámetros que podemos modificar entre ellos -r para definir el número de conexiones por segundo, -x para modificar el tamaño máximo de los datos extras de cada request, -p para definir un timeout por conexión diferente, y muchos más parámetros para poder personalizar el test y poder realizar diferentes pruebas.

Exercici 3

En este ejercicio vamos a implementar un proxy balancer para intentar evitar la sobrecarga de un servidor apache. En el último ejercicio vimos como con slowhttptest podríamos saturar el servidor apache de la máquina B, ahora vamos a configurar 2 máquinas más una máquina B2 que tendrá el servicio apache instalado y otra máquina D estas máquinas están conectadas de la siguiente forma:



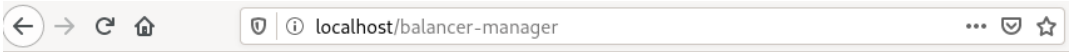
Primero de todo deberemos activar los siguientes módulos de apache en la máquina A:

proxy, proxy_balancer proxy_connect proxy_html proxy_http lbmethod_byrequests lbmethod_bytraffic lbmethod_bybusyness lbmethod_heartbeat status

Una vez tengamos todos los módulos activados, procederemos a crear un site en la máquina A, en mi caso se llamará blancer.conf, aquí configuraremos nuestro proxy-balancer que distribuirá la carga entre la máquina B y B2.

```
<VirtualHost *:80>
        # The ServerName directive sets the request scheme, hostname and port that
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request's Host: header to
        # match this virtual host. For the default virtual host (this file) this
        # value is not decisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicitly.
        ServerName balancer.gax.org

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html

        <Proxy balancer://mycluster>
                BalancerMember http://172.16.1.2:80
                BalancerMember http://172.16.3.2:80
                Options Indexes FollowSymlinks Multiviews
                #AllowOveride None
                Order allow,deny
                Allow from all
                ProxySet lbmethod=bytraffic
        </Proxy>

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

        # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
        # error, crit, alert, emerg.
        # It is also possible to configure the loglevel for particular
        # modules, e.g.
        #LogLevel info ssl:warn

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined
        ProxyRequests Off
        #ProxyPreserveHost On
        ProxyPass /balancer-manager !
        ProxyPass / balancer://mycluster/
        ProxyPassReverse / balancer://mycluster

        ProxyPass / http://172.16.1.2//
        ProxyPassReverse / http://172.16.1.2//
        ProxyPass / http://172.16.3.2//
        ProxyPassReverse / http://172.16.3.2//
        # For most configuration files from conf-available/, which are
```

El balanceo de carga lo podemos hacer por tráfico o por request, en nuestro caso decidimos realizar el balanceo por tráfico, también indicamos los nodos en los cuales se va a redirigir las peticiones, en nuestro caso la máquina B y B2. También configuramos un manager que nos proporcionará estadísticas sobre los nodos.



Load Balancer Manager for localhost

Server Version: Apache/2.4.51 (Debian) OpenSSL/1.1.1k mod_python/3.5.0- Python/3.9.2
Server Built: 2021-10-07T17:49:44
Balancer changes will NOT be persisted on restart.
Balancers are inherited from main server.
ProxyPass settings are inherited from main server.

LoadBalancer Status for balancer://mycluster [p420bf4f2_mycluster]

MaxMembers	StickySession	DisableFailover	Timeout	FailoverAttempts	Method	Path	Active
2 [2 Used]	(None)	Off	0	1	bytraffic	/	Yes

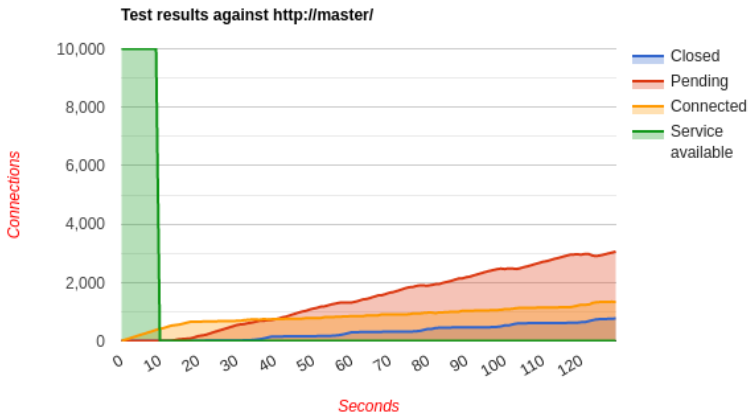
Worker URL	Route	RouteRedir	Factor	Set	Status	Elected	Busy	Load	To	From
http://172.16.1.2			1.00	0	Init Ok	51	3	0	3.4K	115K
http://172.16.3.2			1.00	0	Init Ok	49	0	0	3.2K	125K

Apache/2.4.51 (Debian) Server at localhost Port 80

Una vez configurado, habilitamos el site y probamos desde la máquina D si nos redirige algún node de los dos, si recargamos la página reiteradamente podemos ver como nos redirige a diferentes nodos.

Ahora probaremos de realizar desde la maquina D un slowhttptest con los mismos parámetros sobre la máquina máster, de tal manera que el tráfico se redirigirá sobre las máquina B y B2.

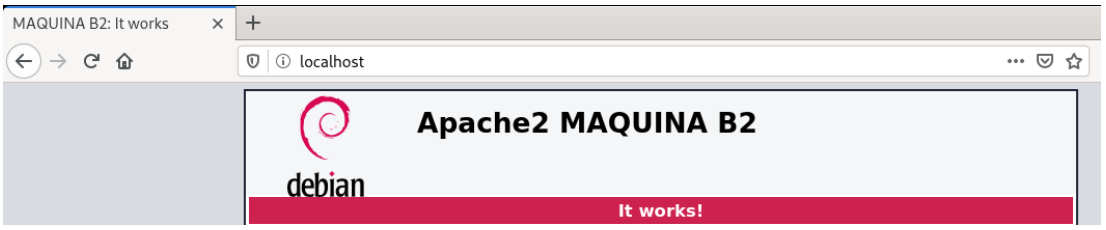
Test parameters	
Test type	SLOW HEADERS
Number of connections	10000
Verb	GET
Content-Length header value	4096
Cookie	
Extra data max length	68
Interval between follow up data	10 seconds
Connections per seconds	50
Timeout for probe connection	5
Target test duration	1000 seconds
Using proxy	no proxy



Tras realizar el test con los mismos parámetros que en el ejercicio 2 vemos que el tiempo que aguantan los servidores de apache soportando un DoS es el mismo, esto es debido a que el balanceador de carga no puede soportar tantas peticiones simultáneas. Dado que los nodos B y B2 no están caídos los usuarios que ya están conectados podrán seguir usando

el servicio, pero los usuarios que quieran conectarse no podrán, dado que el balanceador se cae. Para demostrar esto accedo directamente en uno de los nodos mientras el balanceador está caído.

```
Mon Nov 29 18:26:48 2021:
        slowhttptest version 1.8.2
 - https://github.com/shekyan/slowhttptest -
test type:                        SLOW HEADERS
number of connections:            10000
URL:                              http://master/
verb:                             GET
cookie:
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

Mon Nov 29 18:26:48 2021:
slow HTTP test status on 275th second:

initializing:        0
pending:             2446
connected:           1551
error:               0
closed:              6003
service available:   NO
```



Dado que el balanceador de carga no evita que nos puedan realizar una denegación de servició en la máquina A que hace de proxy_balancer, tendremos que aplicar algún módulo como mod_security, o mod_evade. Tras intentar configurar los dos módulos, nos protege de un ataque DoS realizado desde una librería de test que hay en mod_evade.

```
root@master:/etc/apache2#  perl /usr/share/doc/libapache2-mod-evasive/examples/test.pl
HTTP/1.1 400 Bad Request
HTTP/1.1 400 Bad Request
HTTP/1.1 400 Bad Request
HTTP/1.1 400 Bad Request
HTTP/1.1 400 Bad Request
HTTP/1.1 400 Bad Request
HTTP/1.1 400 Bad Request
HTTP/1.1 400 Bad Request
HTTP/1.1 400 Bad Request
HTTP/1.1 400 Bad Request
```

Pero slowhttptest nos sigue tirando el servicio en la máquina A, impidiendo que se conecten los usuarios.

Para evitar que se pueda realizar una denegación desde slowthttpest añadimos una regla a iptables para limitar las conexiones por cada ip al puerto 80 en 5. Esto evitará que se conecten más de 5 conexiones por ip.

```
root@master:/var/log/apache2# iptables -A INPUT -p tcp --syn --dport 80 -m connlimit --connlimit-above 5 -j REJECT
```

El problema que tenemos ahora es que todas las peticiones que no son atendidas se quedan esperando a ser aceptadas, estas irán incrementando con lo que también tumbara la máquina A. Para intentar mitigar este problema modificó el fichero de configuración en /etc/apache2/apache.conf y modificar el timeout de 300 a 3 segundos para así aumentar la cantidad de conexiones que se cierran. Tras volver a realizar un slowhttptest, vemos que ha mejorado significativamente todo y que todavía tenemos intervalos con caídas del servicio en A

Test parameters	
Test type	SLOW HEADERS
Number of connections	10000
Verb	GET
Content-Length header value	4096
Cookie	
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

Dado que slowhttptest es el “machote” de los programas para realizar denegación de servicio y es bastante complicado configurar correctamente mod_security y mod_evade. Hemos intentado crear una “jail” con fail2ban para banear con una regla en iptables. Lamentablemente no hemos podido acabar de configurar correctamente la jaula para que funcione.

Exercici 4
Considerar que els nostres servidors de SSH estan en un entorn hostil i necessitem controlar automàticament les connexions als servidors.

a) Instal·lar el programari medusa del repositori de Debian sobre A, crear un usuari en B, baixar una llista de passwd per exemple alguna de:

Grau d'Enginyeria Informàtica: Gestió i Administració de Xarxes
https://wiki.skullsecurity.org/Passwords
i fer l'atac esbrinar si podem superar per força bruta el passwd a través de connexions ssh a B.

Primero de todo instalamos medusa en la máquina A y creamos el usuario pixi con la contraseña “12345678” en la máquina B

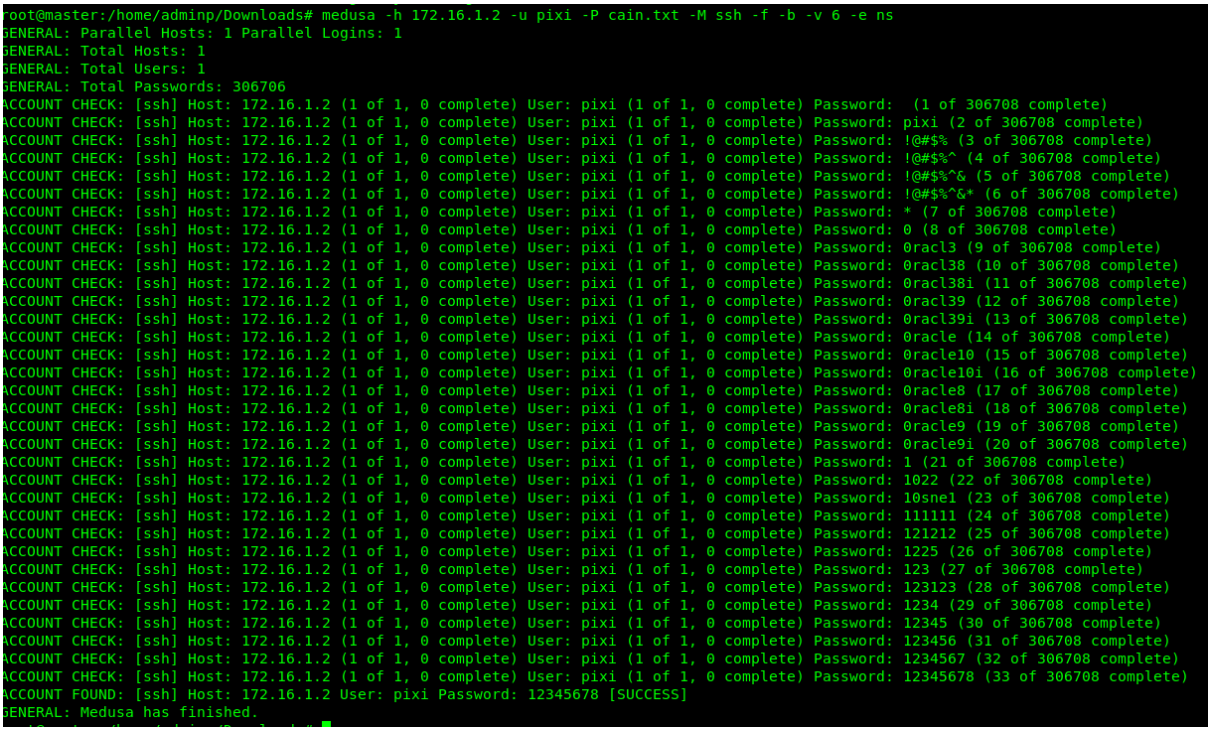
```
root@slave1:/var/www/html# adduser pixi
Adding user `pixi' ...
Adding new group `pixi' (1001) ...
Adding new user `pixi' (1001) with group `pixi' ...
Creating home directory `/home/pixi' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for pixi
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] y
```

Seguidamente, vamos a descargar una página de contraseñas con la que vamos a realizar el ataque de fuerza bruta.

Una vez descargado el fichero podemos proceder a realizar el ataque, para ello suponemos que sabemos la dirección ip del objetivo en nuestro caso sera la maquina B y que tiene activado el puerto 22 con el servicio de ssh.
Vamos a analizar la comanda que usaremos, con -h indicamos la dirección ip del objetivo si queremos indicar una lista podemos usar -H , -u indicamos el usuario el cual va a ser atacado, con -U podemos indicar una lista de usuarios, con -P indicamos que lista de contraseñas queremos probar, -M indicamos el módulo el cual vamos a utilizar en nuestro caso el módulo de ssh, finalmente indicamos -f -b -v -e ns para indicar que se detendrá al encontrar la contraseña , se van a suprimir los banners, -v para mostrar información extra siendo 6 el nivel más alto y finalmente -e con el que indicamos que se verifique un password vacío y el mismo nombre del usuario.

```
root@master:/home/adminp/Downloads# medusa -h 172.16.1.2 -u pixi -P cain.txt -M ssh -f -b -v 6 -e ns
```

Tras ejecutar medusa, vemos que empieza a probar contraseñas, tras realizar 33 intentos, dado que nuestra contraseña es poco segura, vemos que nos indica que ha encontrado la contraseña 12345678 para el usuario pixi. Ahora solo tendríamos que contactarnos a la máquina B con ssh utilizando el usuario pixi y la contraseña 12345678



Ahora vamos a instalar fail2ban para protegernos de ataques de fuerza bruta como el que acabamos de realizar con medusa, fail2ban tiene dos ficheros de configuración

/etc/fail2ban/fail2ban.conf: Este es el fichero de configuración para los ajustes operacionales del deamon de fial2ban, ajustes como el nivel de log, el socket y el id de proceso.

/etc/fail2ban/jail.conf: Este fichero contiene la configuración donde se definirá el tiempo por defecto de baneo, el número de intentos de loging, la lista de ips que pueden conectarse etc.. Básicamente desde este fichero podremos configurar todos los parámetros.

Una vez instalado por defecto, fail2ban solo dejará que cada ip pruebe un máximo de 5 contraseñas, después banea la ip durante el tiempo especificado, en mi caso he modificado el tiempo a 1 minuto dado que por defecto viene un tiempo de 10 minutos. Podríamos modificar múltiples parámetros, pero para realizar la prueba dejaremos esta configuración. Procedemos nuevamente a realizar el mismo ataque con medusa, pero esta vez al llegar al intento 5 se queda bloqueado y no prosigue, esto es debido a que fail2ban está bloqueando la ip. Medusa al cabo de unos segundos nos muestra que el host no está disponible, esto es debido a que no permite más conexiones.

```
root@master:/home/adminp/Downloads# medusa -h 172.16.1.2 -u pixi -P cain.txt -M ssh -f -b -v 6 -e ns
GENERAL: Parallel Hosts: 1 Parallel Logins: 1
GENERAL: Total Hosts: 1
GENERAL: Total Users: 1
GENERAL: Total Passwords: 306706
ACCOUNT CHECK: [ssh] Host: 172.16.1.2 (1 of 1, 0 complete) User: pixi (1 of 1, 0 complete) Password:  (1 of 306708 complete)
ACCOUNT CHECK: [ssh] Host: 172.16.1.2 (1 of 1, 0 complete) User: pixi (1 of 1, 0 complete) Password: pixi (2 of 306708 complete)
ACCOUNT CHECK: [ssh] Host: 172.16.1.2 (1 of 1, 0 complete) User: pixi (1 of 1, 0 complete) Password: !@#$% (3 of 306708 complete)
ACCOUNT CHECK: [ssh] Host: 172.16.1.2 (1 of 1, 0 complete) User: pixi (1 of 1, 0 complete) Password: !@#$%^ (4 of 306708 complete)
ACCOUNT CHECK: [ssh] Host: 172.16.1.2 (1 of 1, 0 complete) User: pixi (1 of 1, 0 complete) Password: !@#$%^& (5 of 306708 complete)



GENERAL: Unable to connect (invalid socket): unreachable destination - 172.16.1.2
NOTICE: ssh.mod: failed to connect, port 22 was not open on 172.16.1.2
GENERAL: Medusa has finished.
```

Si ahora intentamos conectarnos a la máquina B por ssh, vemos que el host nos refusa la conexión en el puerto 22.

```
root@master:/home/adminp/Downloads# ssh pixi@slave1
ssh: connect to host slave1 port 22: Connection refused
root@master:/home/adminp/Downloads#
```

Si vemos iptables en la máquina B podremos ver que se ha añadido una regla para realizar un REJECT de las conexiones en el puerto 22 para la ip 172.16.1.1 que es la ip de la máquina A.

```
root@slave1:/etc/fail2ban# iptables -n -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
f2b-sshd   tcp  --  0.0.0.0/0            0.0.0.0/0            multiport dports 22

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain f2b-sshd (1 references)
target     prot opt source               destination
REJECT     all  --  172.16.1.1           0.0.0.0/0            reject-with icmp-port-unreachable
RETURN     all  --  0.0.0.0/0            0.0.0.0/0
root@slave1:/etc/fail2ban#
```

Ahora vamos a modificar la configuración para que al banear la ip en el puerto 22 de realize un DROP de los paquetes en lugar de un REJECT, con lo que no se responderá a las peticiones de SSH con el mensaje de “Connection Refused”. Para ello primero deberemos modificar el fichero de configuración de fail2ban donde se establecen las reglas para banear las ip via iptables. El fichero en cuestión que nos interesa es /etc/fail2ban/actions.d/itables-common.conf. Aquí tendremos que buscar la variable blocktype donde podremos ver que tiene por defecto un REJECT y modificar la línea con un DROP. Igual que en la imagen a continuación y reiniciamos el servicio de fail2ban para aplicar los cambios

```
# Option:  blocktype
# Note:    This is what the action does with rules. This can be any jump target
#          as per the iptables man page (section 8). Common values are DROP
#          REJECT, REJECT --reject-with icmp-port-unreachable
# Values:  STRING
#blocktype = REJECT --reject-with icmp-port-unreachable
blocktype = DROP
```

Ahora vamos a realizar la prueba para confirmar que se realiza un DROP de los paquetes en el puerto 22. Volvemos a ejecutar medusa y esperamos a que la máquina B bloquee la ip. Tal como podemos ver en la imagen ahora se ha aplicado una regla de DROP en lugar de REJECT. De tal forma que ahora al intentar conectarse por ssh nos dará time out en lugar de conexión refusada.

```
root@slave1:/etc/fail2ban/action.d# iptables -n -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
f2b-sshd   tcp  --  0.0.0.0/0            0.0.0.0/0           multiport dports 22

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain f2b-sshd (1 references)
target     prot opt source               destination
DROP       all  --  172.16.1.1           0.0.0.0/0
RETURN     all  --  0.0.0.0/0            0.0.0.0/0
```

```
root@master:/home/adminp/Downloads# ssh pixi@172.16.1.2
ssh: connect to host 172.16.1.2 port 22: Connection timed out
root@master:/home/adminp/Downloads#
```