



Trabajo Practico 2: Scripting avanzado

1_ El script “**ejercicio1.sh**” recolecta el UUID, nombre, tipo de sistema de archivo y su montaje de la información que arroja el comando “**blkid**”. Obteniendo una salida como el de la siguiente imagen:

```

/bin/bash
/bin/bash 149x38
familia@equipo-salon ~/TP_2-Script $ sudo ./ejercicio_1.sh
=====
UUID: C248A5EC48A5DF81
NOMBRE: Reservado para el sistema
TIPO: ntfs
PUNTO DE MONTAJE: /dev/sda1 No esta montado
=====
UUID: C00C0D530C0D45BE
NOMBRE: SIN NOMBRE
TIPO: ntfs
PUNTO DE MONTAJE: /dev/sda2 No esta montado
=====
UUID: 7CC60B12C60ACC78
NOMBRE: DATOS
TIPO: ntfs
PUNTO DE MONTAJE: /dev/sda3 en /media/Datos
=====
UUID: 3819d6d2-962c-4435-8bf6-054e9d89b0ab
NOMBRE: SIN NOMBRE
TIPO: swap
PUNTO DE MONTAJE: /dev/sda5 No esta montado
=====
UUID: ef79213a-bb55-4650-887d-3142b73ddf08
NOMBRE: SIN NOMBRE
TIPO: ext4
PUNTO DE MONTAJE: /dev/sda6 en /
=====
UUID: 92B7-CAB3
NOMBRE: KINGSTON
TIPO: vfat
PUNTO DE MONTAJE: /dev/sdc1 en /media/familia/KINGSTON
=====
familia@equipo-salon ~/TP_2-Script $
```

2_ El script llamado “**ejercicio2.sh**” recopila información del log de autenticación del sistema operativo y obtiene aquellas conexiones SSH que se realizaron al equipo y calcula cuanto tiempo duró esa conexión. Obteniendo un resultado como el siguiente:

```
/bin/bash
/bin/bash 149x38
familia@equipo-salon ~/Escritorio/TP_2/TP_2-Script $ ./ejercicio_2.sh
El usuario [familia] mantuvo una sesion SSH de 0 horas, 3 minutos y 50 segundos
El usuario [familia] mantuvo una sesion SSH de 0 horas, 5 minutos y 44 segundos
El usuario [familia] mantuvo una sesion SSH de 2 horas, 27 minutos y 23 segundos
El usuario [familia] mantuvo una sesion SSH de 0 horas, 0 minutos y 3 segundos
El usuario [familia] mantuvo una sesion SSH de 0 horas, 0 minutos y 14 segundos
El usuario [familia] mantuvo una sesion SSH de 0 horas, 42 minutos y 22 segundos
El usuario [familia] mantuvo una sesion SSH de: Informacion incompleta y/o la sesion esta activa
El usuario [familia] mantuvo una sesion SSH de 0 horas, 3 minutos y 57 segundos
familia@equipo-salon ~/Escritorio/TP_2/TP_2-Script $
```

3_ Este script detecta cuando se presentan picos máximos establecidos en variables que determinan si sobrepaso ese máximo o no, de la información obtenida del comando “**vmstat**”. En la prueba, puse valores mínimos para que se compruebe su funcionamiento, dado que este script tiene más sentido en un servidor que en una pc hogareña, donde se está probando el script.

Obteniendo este resultado:

```
/bin/bash
/bin/bash 149x38
familia@equipo-salon ~/Escritorio/TP_2 $ ./ejercicio_3.sh
Existencia de picos de I/O. INPUT:76 OUTPUT:59
Uso excesivo de memoria de intercambio: 296 MB
familia@equipo-salon ~/Escritorio/TP_2 $ ./ejercicio_3.sh
Existencia de picos de I/O. INPUT:80 OUTPUT:62
Uso excesivo de memoria de intercambio: 377 MB
Existencia de picos maximos en procesos ejecutados: 3
Existencia de picos maximos en procesos esperando: 3
familia@equipo-salon ~/Escritorio/TP_2 $
```

MAXSWAP=40

MAXPROCUN=1

MAXPROCSLEEP=1

MAXINPUT=40

MAXOUTPUT=40

Estos son los valores que utilicé para comprobar que el script funciona, dado que al intentar utilizar mucha memoria **swap** el equipo deja de responder, por tal motivo utilice valores muy bajos, para comprobar su funcionamiento.



Los valores originales que había elegido son los siguiente:

MAXSWAP=2048

MAXPROCRUN=10

MAXPROCSLEEP=10

MAXINPUT=80

MAXOUTPUT=80

4_ Dada una serie de direcciones IP almacenadas en un archivo de texto será pasado como argumento al script llamado **"ejercicio_4.sh"** que comprueba para cada dirección si el host está activo o se perdió la conexión.

Para comprobar este script, cree un archivo llamado **"lista"** y en el coloque el siguiente contenido:

8.8.8.8

192.168.1.100

192.168.1.115

208.67.220.220

192.168.1.50

Obteniendo un resultado como este:

```

/bin/bash
/bin/bash 149x38
familia@equipo-salon ~/Escritorio/TP_2/TP_2-Script $ ./ejercicio_4.sh
Es necesario ingresar archivo con direcciones IP
Ejemplo: "./ejercicio_4.sh lista"
familia@equipo-salon ~/Escritorio/TP_2/TP_2-Script $ ./ejercicio_4.sh lista
Se perdió la conexión con 192.168.1.115
Se perdió la conexión con 192.168.1.50
familia@equipo-salon ~/Escritorio/TP_2/TP_2-Script $
```

5_ El script llamado **"ejercicio5.sh"** obtiene de la salida del comando **"netstat"** todos aquellos puertos nuevos abiertos y a que aplicación corresponde cada puerto. Este script obtiene aquellas conexiones ya establecidas, y dado un tiempo de 30 segundos obtiene otra captura del comando y luego compara para saber que puertos nuevos se han abierto.

Al momento de ejecutar el script, estos puertos estaban abiertos:

```
/bin/bash
familia@equipo-salon ~/Escritorio/TP_2/TP_2-Script $ sudo netstat -punta | grep "ESTABLECIDO"
tcp        0      0 192.168.1.200:34349    31.13.85.8:443        ESTABLECIDO 2309/firefox
tcp        0      0 192.168.1.200:50452    184.173.90.195:80     ESTABLECIDO 2309/firefox
tcp        0    126 192.168.1.200:58017    64.235.151.52:443     ESTABLECIDO 2064/CopyAgent
tcp        0      0 192.168.1.200:56556    74.125.2.185:443     ESTABLECIDO 2309/firefox
tcp        0      0 192.168.1.200:45487    64.235.151.30:443     ESTABLECIDO 2064/CopyAgent
familia@equipo-salon ~/Escritorio/TP_2/TP_2-Script $
```

Luego ejecuto el script e inicio la aplicación “**pidgin**”, por ejemplo, dando como resultado lo siguiente:

```
/bin/bash
familia@equipo-salon ~/Escritorio/TP_2/TP_2-Script $ ./ejercicio_5.sh
Este programa debe ejecutarse como root
Ejemplo: "sudo ./ejercicio_5.sh"
familia@equipo-salon ~/Escritorio/TP_2/TP_2-Script $ sudo ./ejercicio_5.sh
Puerto nuevo abierto [36186] por la aplicacion [CopyAgent]
Puerto nuevo abierto [39933] por la aplicacion [pidgin]
familia@equipo-salon ~/Escritorio/TP_2/TP_2-Script $
```

En la captura de pantalla anterior, se observa los nuevos puertos abiertos y su aplicación correspondiente.

6_ Con este script obtendremos información sobre el origen máximo y mínimo de trafico que recibe nuestro host, además de la cantidad de bytes enviados y recibido por la interfaz de red activa.

Se ejecuta el script llamado “**ejercicio6.sh**” y se le pasa como argumento el tiempo que se desea escanear la red: El ejemplo de la imagen el script escanea la red durante 30 segundos.

```
emanuel@notebook: ~
Archivo Editar Ver Buscar Terminal Ayuda
emanuel@notebook:~$ sudo ./ejercicio_6.sh 30
[sudo] password for emanuel:
=====
Interfaz de red analizada: wlan0
Origenes maximos: 186.122.153.36
Origenes minimos: 173.194.119.1
Bytes maximo recibidos: 42.6 MB
Bytes maximo enviados: 6.7 MB
=====
emanuel@notebook:~$
```

7_ Dada una lista de direcciones IP contenidas en un archivo de texto, se pasa como argumento a el script “**ejercicio_7.sh**” y este hace una resolución inversa de las direcciones contenidas en el archivo.

Contenido del archivo llamado “**direcciones**”, por ejemplo:

8.8.8.8

200.58.118.30

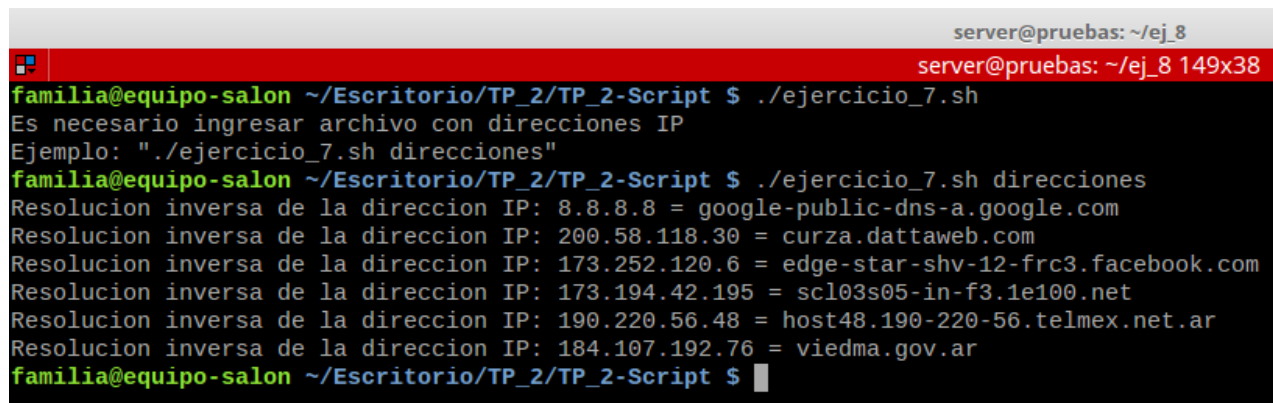
173.252.120.6

173.194.42.195

190.220.56.48

184.107.192.76

Se obtiene un resultado como el siguiente:



```
server@pruebas: ~/ej_8
server@pruebas: ~/ej_8 149x38
familia@equipo-salon ~/Escritorio/TP_2/TP_2-Script $ ./ejercicio_7.sh
Es necesario ingresar archivo con direcciones IP
Ejemplo: "./ejercicio_7.sh direcciones"
familia@equipo-salon ~/Escritorio/TP_2/TP_2-Script $ ./ejercicio_7.sh direcciones
Resolucion inversa de la direccion IP: 8.8.8.8 = google-public-dns-a.google.com
Resolucion inversa de la direccion IP: 200.58.118.30 = curza.dattaweb.com
Resolucion inversa de la direccion IP: 173.252.120.6 = edge-star-shv-12-frc3.facebook.com
Resolucion inversa de la direccion IP: 173.194.42.195 = scl03s05-in-f3.1e100.net
Resolucion inversa de la direccion IP: 190.220.56.48 = host48.190-220-56.telmex.net.ar
Resolucion inversa de la direccion IP: 184.107.192.76 = viedma.gov.ar
familia@equipo-salon ~/Escritorio/TP_2/TP_2-Script $
```

8_ Este script “**ejercicio8.sh**” genera un histograma de accesos **INPUT** y **OUTPUT**, es decir entrantes y salientes del equipo. Apliqué una regla de **IPTables** que registre el log de todo lo que entra y sale.

La regla de **IPTables** está contenida en el script “**iptables.sh**”. Este hace un flush de las reglas anteriores y luego aplica las nuevas que contiene un prefijo de log que es necesario para que funcione el script de este ejercicio:



```
server@pruebas: ~/ej_8
server@pruebas: ~/ej_8 149x38
server@pruebas:~/ej_8$ ./iptables.sh
Debe ser ejecutado como root
Ejemplo: "sudo ./iptables.sh"
server@pruebas:~/ej_8$ sudo ./iptables.sh
[sudo] password for server:
# Generated by iptables-save v1.4.21 on Mon Apr 27 12:01:57 2015
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -j LOG --log-prefix "INPUT_Connection: "
-A OUTPUT -j LOG --log-prefix "OUTPUT_Connection: "
COMMIT
# Completed on Mon Apr 27 12:01:57 2015
server@pruebas:~/ej_8$
```

Obteniendo un resultado como el de la siguiente imagen:

```
server@pruebas: ~/ej_8
server@pruebas: ~/ej_8 149x38
server@pruebas:~/ej_8$ ./ejercicio_8.sh
=====
# La contabilización corresponde a la fecha [lun abr 27 11:44:48 ART 2015] #
=====

Histograma de acceso por dominio entrante:

[Dominio] [Nº acceso] [Histograma]
192.168.1.200 3220 #####
200.236.31.4 1174 #####
91.189.91.23 470 #####
213.186.33.87 56 ##
208.67.220.220 10 #
192.168.1.4 9 #
192.168.1.130 4 #
127.0.0.1 4 #
0.0.0.0 2 #

Histograma de acceso por dominio saliente:

[Dominio] [Nº acceso] [Histograma]
192.168.1.200 2558 #####
200.236.31.4 868 #####
91.189.91.23 328 #####
213.186.33.87 51 ##
208.67.220.220 13 #
192.168.1.255 4 #
127.0.0.1 4 #

server@pruebas:~/ej_8$
```