

2024-2025



Universidad
Rey Juan Carlos

PRACTICA 1

INTELIGENCIA DE LA CIBERSEGURIDAD

TABLA DE CONTENIDO

1. Parte Windows - PowerShell	2
1. Realiza los siguientes ejercicios de localización de datos en Windows mediante órdenes únicas encadenadas con tuberías:	2
Realiza los siguientes scripts en PowerShell.	7
Parte Linux	13
Realiza los siguientes ejercicios de localización de datos en Linux:	13
2. Realiza el siguiente script en bash	20

1. PARTE WINDOWS - POWERSHELL

1. REALIZA LOS SIGUIENTES EJERCICIOS DE LOCALIZACIÓN DE DATOS EN WINDOWS MEDIANTE ÓRDENES ÚNICAS ENCADENADAS CON TUBERÍAS:

- 1) Muestra los nombres de los usuarios que están activos en el sistema y que han iniciado sesión alguna vez en el último mes.

Para la realización de este apartado se hace uso del comando `Get-LocalUser` para obtener los usuarios locales.

```

PS C:\Users\user\Desktop> Get-LocalUser | Get-member

TypeName: Microsoft.PowerShell.Commands.LocalUser

Name            MemberType Definition
-----
Clone           Method      Microsoft.PowerShell.Commands.LocalUser Clone()
Equals          Method      bool Equals(System.Object obj)
GetHashCode     Method      int GetHashCode()
GetType         Method      type GetType()
ToString        Method      string ToString()
AccountExpires  Property    System.Nullable[datetime] AccountExpires {get;set;}
Description     Property    string Description {get;set;}
Enabled         Property    bool Enabled {get;set;}
FullName        Property    string FullName {get;set;}
LastLogon       Property    System.Nullable[datetime] LastLogon {get;set;}
Name            Property    string Name {get;set;}
ObjectClass     Property    string ObjectClass {get;set;}
PasswordChangeableDate Property    System.Nullable[datetime] PasswordChangeableDate {get;set;}
PasswordExpires Property    System.Nullable[datetime] PasswordExpires {get;set;}
PasswordLastSet Property    System.Nullable[datetime] PasswordLastSet {get;set;}
PasswordRequired Property    bool PasswordRequired {get;set;}
PrincipalSource Property    System.Nullable[Microsoft.PowerShell.Commands.PrincipalSource] PrincipalSource {ge...
SID             Property    System.Security.Principal.SecurityIdentifier SID {get;set;}
UserMayChangePassword Property    bool UserMayChangePassword {get;set;}
  
```

Ilustración 1 Uso de `Get-member` para saber funciones

Para saber las funciones y campos disponibles de “`Get-LocalUser`” se puede hacer uso de un pipe y “`Get-member`”. Como se puede ver en la ilustración 1, aparecen los campos “`Enabled`” y “`LastLogon`” los cuales se usarán para el comando final.

Comando Resultante: `Get-LocalUser | Where-Object {$_.Enabled -eq $true -and $_.LastLogon -gt (Get-Date).AddMonths(-1)} | Select-Object Name,LastLogon`

La sentencia “`Where-Object`” se utiliza para hacer las comprobaciones que se deseen. En este caso se comprueba que el campo “`Enabled`” sea true y el campo “`LastLogon`” sea de un mes anterior a la fecha actual. Para esto se usa también la sentencia “`Get-Date`” que devuelve la fecha actual y la función “`AddMonths`” que podríamos ver si se hiciera “`Get-member`”. Por último para mostrar el resultado se hace uso de la sentencia “`Select-Object`” para mostrar los campos que nos interesen, en este caso el campo “`Name`” y el campo “`LastLogon`”.

- La sentencia “Select-Object” y “Where-Object” se utilizará en los ejercicios posteriores por lo que no se reiterará en la funcionalidad de estas sentencias explicadas anteriormente.

```
PS C:\Windows\system32> Get-LocalUser | Where-Object { $_.Enabled -eq $true -and $_.LastLogon -gt (Get-Date).AddMonths(-1)} | Select-Object Name, LastLogon

Name LastLogon
----
user 07/10/2024 16:02:38
```

Ilustración 2 Salida apartado 1

- 2) Localiza los diez procesos que más memoria están consumiendo en el sistema ordenando por cantidad de CPU utilizada en orden inverso.

Para la realización de este apartado se hace uso del comando “Get-Process” para mostrar procesos

```
PS C:\Users\user\Desktop> Get-Process | get-member

TypeName: System.Diagnostics.Process

Name MemberType Definition
----
Handles AliasProperty Handles = Handlecount
Name AliasProperty Name = ProcessName
NPM AliasProperty NPM = NonpagedSystemMemorySize64
PM AliasProperty PM = PagedMemorySize64
SI AliasProperty SI = SessionId
VM AliasProperty VM = VirtualMemorySize64
WS AliasProperty WS = WorkingSet64
Disposed Event System.EventHandler Disposed(System.Object, System.EventArgs)
ErrorDataReceived Event System.Diagnostics.DataReceivedEventHandler ErrorDataReceived(System.Object, System.EventArgs)
Exited Event System.EventHandler Exited(System.Object, System.EventArgs)
OutputDataReceived Event System.Diagnostics.DataReceivedEventHandler OutputDataReceived(System.Object, System.EventArgs)
BeginErrorReadLine Method void BeginErrorReadLine()
BeginOutputReadLine Method void BeginOutputReadLine()
CancelErrorRead Method void CancelErrorRead()
CancelOutputRead Method void CancelOutputRead()
Close Method void Close()
CloseMainWindow Method bool CloseMainWindow()
CreateObjRef Method System.Runtime.Remoting.ObjRef CreateObjRef(type requestedType)
Dispose Method void Dispose(), void IDisposable.Dispose()
Equals Method bool Equals(System.Object obj)
GetHashCode Method int GetHashCode()
GetLifetimeService Method System.Object GetLifetimeService()
GetType Method type GetType()
InitializeLifetimeService Method System.Object InitializeLifetimeService()
Kill Method void Kill()
Refresh Method void Refresh()
Start Method bool Start()
ToString Method string ToString()
WaitForExit Method bool WaitForExit(int milliseconds), void WaitForExit()
WaitForInputIdle Method bool WaitForInputIdle(int milliseconds), bool WaitForInputIdle()
_NounName NoteProperty string _NounName=Process
BasePriority Property int BasePriority {get;}
Container Property System.ComponentModel.IContainer Container {get;}
EnableRaisingEvents Property bool EnableRaisingEvents {get;set;}
ExitCode Property int ExitCode {get;}
ExitTime Property datetime ExitTime {get;}
Handle Property System.IntPtr Handle {get;}
HandleCount Property int HandleCount {get;}
HasExited Property bool HasExited {get;}
Id Property int Id {get;}
```

Ilustración 3 Get-Member de Get-Process

Después de realizar el Get-Member para saber funciones y campos del comando Get-Process, se hará uso de los campos WS (cantidad de memoria que consume) y CPU (cantidad de CPU).

Comando resultante: `Get-Process | Sort-Object WS -Descending | Sort-Object -First 10 | Sort-Object CPU -Descending | Select-Object Name, id, WS, CPU`

En este comando se hace uso de la sentencia “Sort-Object” para ordenar la salida del subcomando anterior al pipe. En el primer caso se ordenará de mayor a menor los procesos procedentes de la salida de Get-Process. Posteriormente se volverá a usar la sentencia “Sort-Object para coger solo los 10 primeros procesos de la salida y posteriormente se ordenarán esos 10 procesos de mayor a menor en función de la CPU que consumen. Por último se hace uso de “Select-Object” para mostrar los campos necesarios.

```
PS C:\Windows\system32> Get-Process | Sort-Object WS -descending | Select-Object -First 10 | Sort-Object CPU -Descending
| Select-Object Name, id, WS, CPU
```

Name	Id	WS	CPU
explorer	580	127307776	20,734375
SearchApp	4624	210280448	17,140625
explorer	4340	172732416	14,78125
MsMpEng	3284	201998336	14,5625
msedge	8868	130174976	9,546875
OneDrive	7160	105000960	6,09375
msedge	6188	109453312	5,0625
OneDrive	1108	93827072	4,546875
powershell	3924	91168768	3,796875
SearchApp	8752	153710592	2,703125

Ilustración 4 Salida del apartado 2

3) Obtén mediante un comando el nombre del Fabricante de la placa base de tu ordenador, la versión del sistema operativo y la versión de la BIOS de tu equipo.

Para la realización de este apartado es necesario utilizar el comando “Get-WmiObject”. En general este comando permite administrar y consultar información sobre dispositivos, configuraciones, servicios, etc.

Puesto que se pretende obtener información de la placa base, es preciso usar el argumento “win32_baseboard”.

```

PS C:\Users\user\Desktop> Get-WmiObject win32_baseboard | Get-member

TypeName: System.Management.ManagementObject#root\cimv2\Win32_BaseBoard

Name      MemberType Definition
-----
PSComputerName AliasProperty PSComputerName = __SERVER
IsCompatible Method      System.Management.ManagementBaseObject IsCompatible(System.String ElementToCheck)
Caption     Property    string Caption {get;set;}
ConfigOptions Property    string[] ConfigOptions {get;set;}
CreationClassName Property    string CreationClassName {get;set;}
Depth       Property    float Depth {get;set;}
Description  Property    string Description {get;set;}
Height      Property    float Height {get;set;}
HostingBoard Property    bool HostingBoard {get;set;}
HotSwappable Property    bool HotSwappable {get;set;}
InstallDate  Property    string InstallDate {get;set;}
Manufacturer Property    string Manufacturer {get;set;}
Model        Property    string Model {get;set;}
Name         Property    string Name {get;set;}
OtherIdentifyingInfo Property    string OtherIdentifyingInfo {get;set;}
PartNumber   Property    string PartNumber {get;set;}
PoweredOn    Property    bool PoweredOn {get;set;}
Product      Property    string Product {get;set;}
Removable     Property    bool Removable {get;set;}
Replaceable  Property    bool Replaceable {get;set;}
RequirementsDescription Property    string RequirementsDescription {get;set;}
RequiresDaughterBoard Property    bool RequiresDaughterBoard {get;set;}
SerialNumber  Property    string SerialNumber {get;set;}
SKU           Property    string SKU {get;set;}
SlotLayout    Property    string SlotLayout {get;set;}
SpecialRequirements Property    bool SpecialRequirements {get;set;}
Status        Property    string Status {get;set;}
Tag           Property    string Tag {get;set;}
Version       Property    string Version {get;set;}
Weight        Property    float Weight {get;set;}
Width         Property    float Width {get;set;}
__CLASS       Property    string __CLASS {get;set;}
__DERIVATION  Property    string[] __DERIVATION {get;set;}
__DYNASTY     Property    string __DYNASTY {get;set;}
__GENUS       Property    int __GENUS {get;set;}
__NAMESPACE  Property    string __NAMESPACE {get;set;}
__PATH        Property    string __PATH {get;set;}
__PROPERTY_COUNT Property    int __PROPERTY_COUNT {get;set;}
__RELPATH     Property    string __RELPATH {get;set;}
__SERVER      Property    string __SERVER {get;set;}
__SUPERCLASS  Property    string __SUPERCLASS {get;set;}
PSStatus      PropertySet  PSStatus {Status, Name, PoweredOn}
ConvertFromDateTime ScriptMethod System.Object ConvertFromDateTime();
  
```

Ilustración 5 Get-member del comando Get-WmiObject

Después de comprobar los campos y funciones del comando anterior, se contempla la existencia del campo “Manufacturer”. Por tanto, el comando resultante es el siguiente:

Get-WmiObject win32_baseboard | Select-Object Manufacturer

```

PS C:\Windows\system32> Get-WmiObject win32_baseboard | Select-Object Manufacturer

Manufacturer
-----
Oracle Corporation
  
```

Ilustración 6 Salida del comando Get-WmiObject

Puesto que la versión del sistema operativo y la versión de la BIOS no se encuentra en “win32_baseboard”, no se puede obtener toda esta información con un solo comando encadenado por pipes, por lo que se harán 3 comandos independientes (se puede poner en una sola línea usando punto y coma, pero siguen siendo 3 comandos)

Para obtener información sobre la versión del sistema operativo se hará uso del argumento “Win32_operationsystem”

Comando resultante: *“Get-WmiObject win32_operationSystem | Select-Object version”*

```
PS C:\Users\user\desktop> Get-WmiObject Win32_OperatingSystem | Select-Object Version
Version
-----
10.0.19044
```

Ilustración 7 Salida comando version

Para obtener la version de la BIOS, se utilizará el argumento “Win32_Bios” y el comando resultante será similar a los anteriores.

Comando resultante: *“Get-Win32_BIOS | Select-Object Version”*

```
PS C:\Users\user\desktop> Get-WmiObject Win32_Bios | Select-Object Version
Version
-----
VBOX - 1
```

Ilustración 8 Salida comando version Bios

4) Muestra los 5 últimos eventos del sistema que indiquen un acceso por escritorio remoto

Para realizar este apartado se hará uso de nuevo del comando “Get-WinEvent”. Los inicios de sesión se guardan en un log llamado “Security” por lo que se especificará en el argumento “-logName”.

Por otro lado, los inicios de sesión exitosos se registran con id igual a 4624 por lo que se buscarán inicios de sesión con este id, mientras que para buscar acceso por escritorio remoto se deberá consultar que el valor 8 de las “properties” sea igual a 10 (sabemos esto mirando la documentación).

Comando resultante: *Get-WinEvent -LogName “Security” | Where-Object { \$_.Id -eq 4624 -and \$_.Properties[8].Value -eq “10” } | Select-Object -first 5*

```

PS C:\Users\user\desktop> Get-WinEvent -LogName "Security" | Where-Object { $_.Id -eq 4624 -and $_.Properties[8].Value -eq '2' } | Select-Object -First 5

ProviderName: Microsoft-Windows-Security-Auditing

TimeCreated          Id LevelDisplayName Message
-----
24/10/2024 18:08:05    4624 Información Se inició sesión correctamente en una cuenta....
24/10/2024 18:08:05    4624 Información Se inició sesión correctamente en una cuenta....
24/10/2024 18:07:51    4624 Información Se inició sesión correctamente en una cuenta....
24/10/2024 18:07:51    4624 Información Se inició sesión correctamente en una cuenta....
24/10/2024 18:07:51    4624 Información Se inició sesión correctamente en una cuenta....
  
```

Ilustración 9 Salida comando escritorio remoto

Puesto que no se ha hecho ningún acceso desde escritorio remoto, no se muestra ningún resultado pero la ilustración 7 muestra la salida cuando se consulta por los accesos normales.

REALIZA LOS SIGUIENTES SCRIPTS EN POWERSHELL.

- 1) Dado el nombre de un usuario por teclado, obtén la fecha y hora de sus últimos tres accesos al ordenador indicando la IP del equipo desde el que se ha conectado

```

$usuario = Read-Host "Introduce el nombre de usuario"

$InicioSesion = Get-WinEvent -LogName Security | Where-Object { $_.Id -eq 4624 -and
$_.Properties[5].Value -eq $usuario } | Select-Object -First 3

foreach ($sesion in $InicioSesion) {
    $loginTime = $sesion.TimeCreated
    $loginIp = $sesion.Properties[18].Value

    Write-Host "Fecha Inicio de Sesion: $loginTime, Direccion IP: $loginIp"
}
  
```

Ilustración 10 Script


```

PS C:\Windows\system32> Get-WinEvent -LogName Security | Where-Object {$_.Id -eq 4624 -and $_.Properties[5].value -eq "user"} | ForEach-Object
{$_} | Format-List *
Value : S-1-5-18
Value : DESKTOP-62V3LQQ5
Value : WORKGROUP
Value : 999
Value : S-1-5-21-486287444-1274488965-2924374261-1001
Value : user
Value : DESKTOP-62V3LQQ
Value : 236063
Value : 2
Value : User32
Value : Negotiate
Value : DESKTOP-62V3LQQ
Value : 00000000-0000-0000-0000-000000000000
Value : -
Value : -
Value : 0
Value : 1412
Value : C:\Windows\System32\svchost.exe
Value : 127.0.0.1
  
```

Ilustración 11 Propiedades

Para la realización del Script se comienza utilizando la función “Read-Host” que sirve para que el usuario introduzca por teclado el valor necesario, en este caso un nombre de usuario.

Para poder hacer la consulta de los inicios de sesión, se hace uso del comando powershell “Get-WinEvent”. Con este comando se pueden consultar diferentes logs pero en este caso el que interesa es el que se llama Security. Para saber en que campo de las propiedades está el usuario y posteriormente la IP se hace uso del siguiente comando “Get-WinEvent -logName Security | Where-Object {\$_.Id -eq 4624} | ForEach-Object {\$_.Properties} | Format-List *” representado en la ilustración 11. Una vez conocidos los campos de las propiedades de interés se realiza el script representado en la figura 10.

```

PS C:\Users\user\Desktop> ./script1.ps1
Enter the username: user
Logon Time: 10/10/2024 15:35:31, IP Address: 127.0.0.1
Logon Time: 10/10/2024 15:35:31, IP Address: 127.0.0.1
Logon Time: 10/10/2024 15:34:27, IP Address: 127.0.0.1
  
```

Ilustración 12 Salida Script1

- 2) **Añade a un fichero de texto los nombres de las aplicaciones instaladas durante la última hora junto con sus rutas y el usuario que se ha utilizado para instalarse. Crea una tarea programada que lance dicho script cada hora para hacer una monitorización continua de estos datos.**

```
$fechaAct = Get-Date
$fechaFinal = $fechaAct.AddHours(-1)

$AppsInstaladas = Get-WinEvent -LogName 'Application' | Where-Object {
    $_.Id -eq 11707 -and $_.TimeCreated -ge $fechaFinal -and $_.TimeCreated -le $fechaAct
}

$Fichero = "C:\Users\user\Documents\installed_apps.txt"

$DetallesApp = $AppsInstaladas | ForEach-Object {
    if ($nombre = $_.Properties[0].Value -match ':\s*(.*)\s* --'){
        $nombreApp = $matches[1]
    }
    $rutaApp = (Get-ItemProperty "HKLM:\Software\Microsoft\Windows\CurrentVersion\Uninstall/*" | Where-Object { $_.DisplayName -eq $nombreApp }).InstallLocation
    if (-not $appPath){
        $appPath="No hay ruta"
    }
    $userId = $_.UserId
    $userName = (Get-LocalUser | Where-Object{$_._SID -eq "$userId"}).Name
    $fecha = $_.TimeCreated

    "Nombre APP: $nombreApp, PATH: $appPath, Nombre Usuario: $userName, Fecha: $fecha"
}

$DetallesApp | Out-File -FilePath $Fichero -Append
```

Ilustración 13 Script2

***NOTA:** Para que este script funcione debe crearse previamente un fichero llamado `installed_apps.txt` en documentos

En el desarrollo de este script se hará uso del comando “Get-WinEvent” para obtener las aplicaciones instaladas. En este caso, las aplicaciones se registran en un log llamado “Application” y las que interesan son las que tienen “id” igual a 11707 que son las que se han instalado correctamente. Para comprobar que se han instalado en la última hora, se hace uso de “Get-Date” para obtener la fecha actual y la función “AddHours” para obtener la fecha de 1 hora anterior. Una vez conocemos las 2 fechas se hace uso de la sentencia “Where-Object” para hacer las comprobaciones correspondientes.

Para la ruta de las aplicaciones se hace uso de “Get-ItemProperty” seguido del directorio “HKLM:\Software\Microsoft\Windows\CurrentVersion\Uninstall/*” que contiene información de la ruta de las aplicaciones instaladas en la propiedad “InstallLocation”.

En el caso del nombre de usuario, no sirve con consultar un campo o propiedad de un objeto del estilo “Get-WinEvent” ya que lo que se guarda en este tipo de objetos es el SID de los usuarios no el nombre. Por ello, es necesario usar el comando “Get-LocalUser” para relacionar los SID con los nombres de usuario.

Por último para la fecha basta con consultar el campo “TimeCreated”.

```

Nombre APP: Java Auto Updater, PATH: No hay ruta, Nombre Usuario: user, Fecha: 10/24/2024 15:35:26
Nombre APP: Java 8 Update 431, PATH: No hay ruta, Nombre Usuario: user, Fecha: 10/24/2024 15:35:24
Nombre APP: Python 3.13.0 pip Bootstrap (64-bit), PATH: No hay ruta, Nombre Usuario: user, Fecha: 10/24/2024 16:02:00
Nombre APP: Python Launcher, PATH: No hay ruta, Nombre Usuario: user, Fecha: 10/24/2024 16:01:40
Nombre APP: Python 3.13.0 Tcl/Tk Support (64-bit), PATH: No hay ruta, Nombre Usuario: user, Fecha: 10/24/2024 16:01:37
Nombre APP: Python 3.13.0 Documentation (64-bit), PATH: No hay ruta, Nombre Usuario: user, Fecha: 10/24/2024 16:01:08
Nombre APP: Python 3.13.0 Test Suite (64-bit), PATH: No hay ruta, Nombre Usuario: user, Fecha: 10/24/2024 16:00:55
Nombre APP: Python 3.13.0 Standard Library (64-bit), PATH: No hay ruta, Nombre Usuario: user, Fecha: 10/24/2024 16:00:12
Nombre APP: Python 3.13.0 Development Libraries (64-bit), PATH: No hay ruta, Nombre Usuario: user, Fecha: 10/24/2024 15:59:59
Nombre APP: Python 3.13.0 Executables (64-bit), PATH: No hay ruta, Nombre Usuario: user, Fecha: 10/24/2024 15:59:52
Nombre APP: Python 3.13.0 Core Interpreter (64-bit), PATH: No hay ruta, Nombre Usuario: user, Fecha: 10/24/2024 15:59:49
Nombre APP: Java Auto Updater, PATH: No hay ruta, Nombre Usuario: user, Fecha: 10/24/2024 15:35:26
Nombre APP: Java 8 Update 431, PATH: No hay ruta, Nombre Usuario: user, Fecha: 10/24/2024 15:35:24
Nombre APP: Python 3.13.0 pip Bootstrap (64-bit), PATH: No hay ruta, Nombre Usuario: user, Fecha: 10/24/2024 16:02:00
Nombre APP: Python Launcher, PATH: No hay ruta, Nombre Usuario: user, Fecha: 10/24/2024 16:01:40
Nombre APP: Python 3.13.0 Tcl/Tk Support (64-bit), PATH: No hay ruta, Nombre Usuario: user, Fecha: 10/24/2024 16:01:37
Nombre APP: Python 3.13.0 Documentation (64-bit), PATH: No hay ruta, Nombre Usuario: user, Fecha: 10/24/2024 16:01:08
Nombre APP: Python 3.13.0 Test Suite (64-bit), PATH: No hay ruta, Nombre Usuario: user, Fecha: 10/24/2024 16:00:55
Nombre APP: Python 3.13.0 Standard Library (64-bit), PATH: No hay ruta, Nombre Usuario: user, Fecha: 10/24/2024 16:00:12
Nombre APP: Python 3.13.0 Development Libraries (64-bit), PATH: No hay ruta, Nombre Usuario: user, Fecha: 10/24/2024 15:59:59
Nombre APP: Python 3.13.0 Executables (64-bit), PATH: No hay ruta, Nombre Usuario: user, Fecha: 10/24/2024 15:59:52
Nombre APP: Python 3.13.0 Core Interpreter (64-bit), PATH: No hay ruta, Nombre Usuario: user, Fecha: 10/24/2024 15:59:49
Nombre APP: Java Auto Updater, PATH: No hay ruta, Nombre Usuario: user, Fecha: 10/24/2024 15:35:26
Nombre APP: Java 8 Update 431, PATH: No hay ruta, Nombre Usuario: user, Fecha: 10/24/2024 15:35:24

```

Ilustración 14 Salida del Script2

```

PS C:\Users\user\Desktop> (Get-ItemProperty "HKLM:\Software\Microsoft\Windows\CurrentVersion\Uninstall/*" | Where-Object { $_.DisplayName -eq "Python 3.13.0 Test Suite (64-bit)" }).Installation
PS C:\Users\user\Desktop>

```

Ilustración 15

Como se puede observar en la ilustración 14, las aplicaciones instaladas no muestran la ruta esto puede deberse a 2 motivos. El primero de ellos es la mostrada en la ilustración 15 donde la aplicación aparece al ejecutar el comando “*Get-ItemProperty "HKLM:\Software\Microsoft\Windows\CurrentVersion\Uninstall/*"*” pero no tiene ruta; y la segunda opción es que la aplicación aparezca al ejecutar el comando

“*Get-WinEvent -LogName 'Application' | Where-Object { \$_.Id -eq 11707 -and \$_.TimeCreated -ge \$oneHourAgo -and \$_.TimeCreated -le \$currentTime}*” pero no aparezca al ejecutar el comando anterior, de modo que aunque la aplicación tenga ruta no se puede acceder a ella.

Para crear una tarea programada que lance el script cada hora se ha hecho uso del programador de tareas.

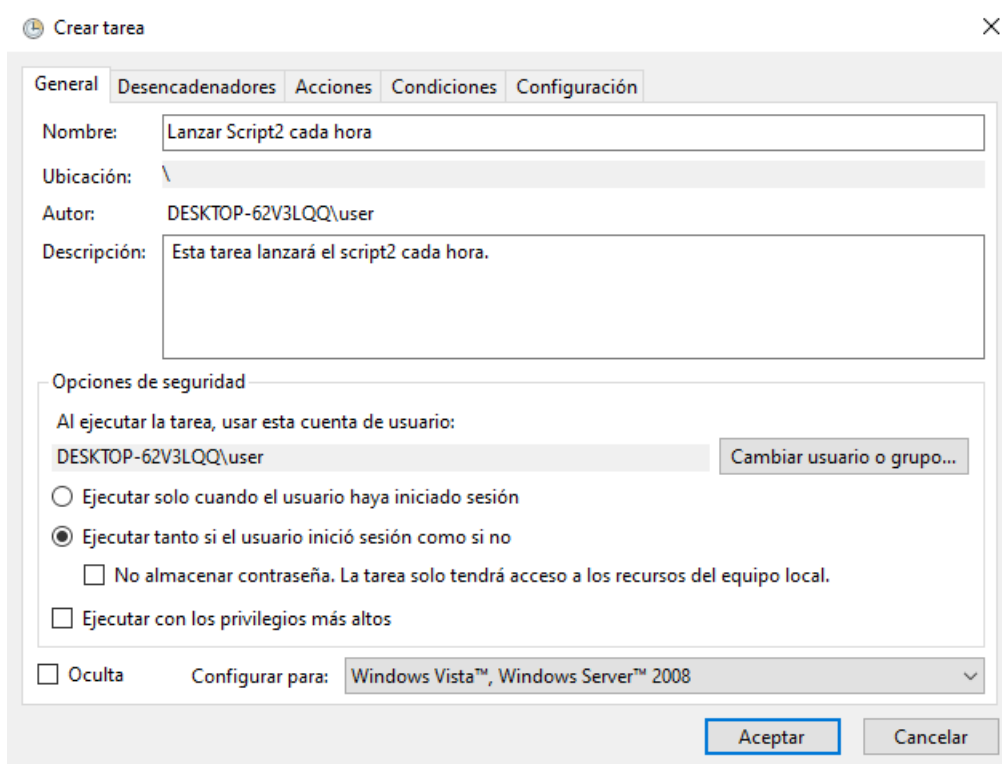
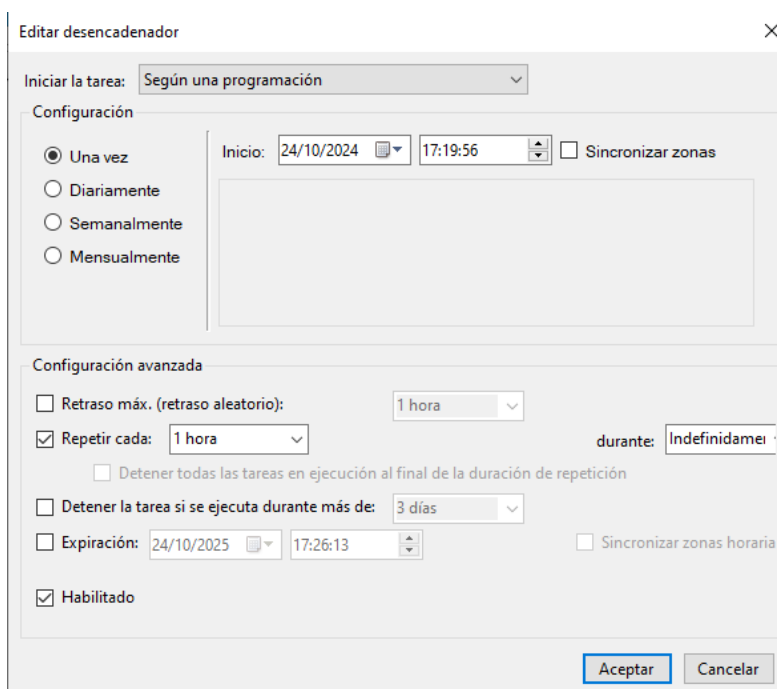


Ilustración 16 Programador de tareas paso 1

Lo primero que se ha realizado es crear la tarea, asignando en primer lugar un nombre, una descripción y habilitando que se ejecute tanto si el usuario ha iniciado sesión como si no.



Posteriormente hay que configurar el desencadenante o “trigger” que lance la tarea. En este caso se ha configurado para que se active una sola vez pero una vez activado, se repita cada hora durante indefinidamente, de modo que la tarea se ejecutará cada hora indefinidamente aunque el trigger se active 1 sola vez.

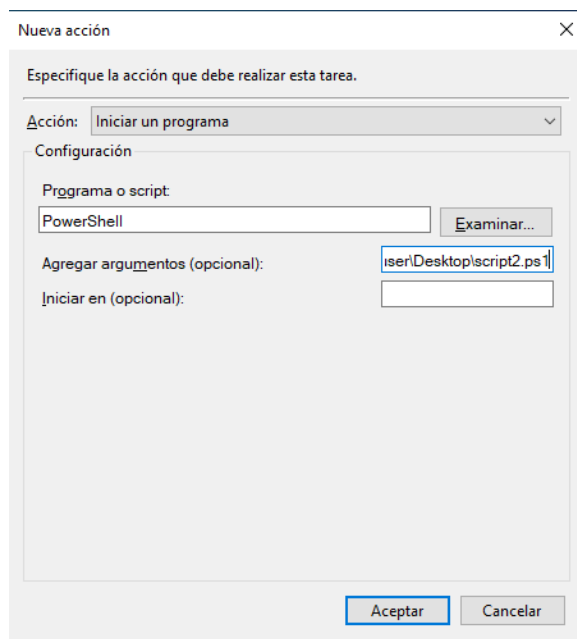


Ilustración 18 Ruta al script y tipo de programa

Por último, se ha especificado el tipo de programa que se va a ejecutar, en este caso powershell; y la ruta al script correspondiente.







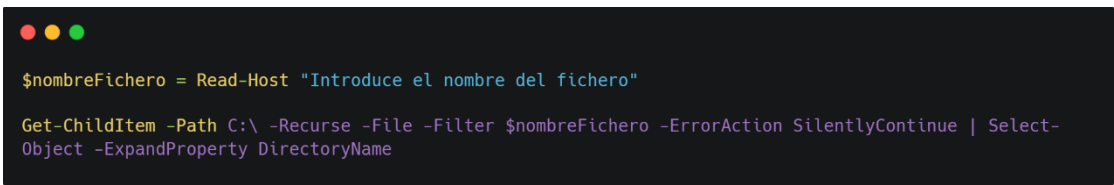
Nombre	Estado	Desencadenadores
 Lanzar Script...	En ejecución	A las 17:28 el 24/10/2024 - Tras desencadenar
 MicrosoftEd...	En ejecución	Se definieron varios desencadenadores
 MicrosoftEd...	Listo	A las 15:46 todos los días - Tras desencadenar
 OneDrive Re...	Listo	A las 16:31 el 23/10/2024 - Tras desencadenar
 OneDrive St...	Listo	A las 15:00 el 01/05/1992 - Tras desencadenar
 PostponeDe...	Listo	A las 16:31 el 27/10/2024 - El desencadenador

Ilustración 19 Tarea en ejecución

Una vez realizado todos los pasos anteriores, la tarea programada se ejecutará cuando corresponda según lo configurado anteriormente. En este caso, cada hora.

- 3) **Pide por teclado el nombre de un fichero y muestra por pantalla todos los nombres de los directorios en los que se encuentra un fichero con dicho nombre.**



```
$nombreFichero = Read-Host "Introduce el nombre del fichero"
Get-ChildItem -Path C:\ -Recurse -File -Filter $nombreFichero -ErrorAction SilentlyContinue | Select-Object -ExpandProperty DirectoryName
```

Ilustración 20 Script2

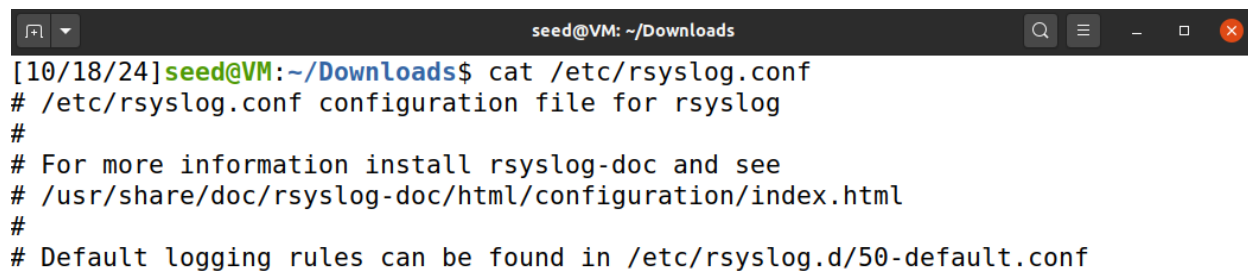
Para la realización del script representado en la ilustración 20 se hace uso del comando “Read-Host” para leer un valor por teclado y de “Get-ChildItem” para obtener los directorios. En este caso se utiliza el argumento “Path” para indicar el directorio desde donde se empezará a buscar, seguido de “Recurse” para hacer una búsqueda recursiva, “File” para que solo se busquen archivos y “Filter” para indicar que el nombre del fichero devuelto debe coincidir con el nombre de fichero introducido por teclado. Puesto que no se dispone de permisos para consultar ciertos directorios, se debe hacer uso también del parámetro “ErrorAction SilentlyContinue”. En caso contrario, el script daría un error al consultar un directorio sin los permisos pertinentes y acabaría la ejecución. Una vez encontrado los resultados, se utiliza un pipe para mostrar el nombre de los directorios, usando para ello el argumento “ExpandProperty DirectoryName”

PARTE LINUX

REALIZA LOS SIGUIENTES EJERCICIOS DE LOCALIZACIÓN DE DATOS EN LINUX:

- a) **Analiza y haz un resumen de cómo se están guardando los logs en tu sistema e indica de dónde saca dicha configuración.**

En este caso para la práctica utilizamos un Ubuntu 20.04, y la mayoría de los logs en esta distribución, así como en los Ubuntu modernos, se ubican en la carpeta /var/log, ya sean del sistema o de servicios como apache (como excepción mysql). Su configuración procede de los dos componentes principales, journald y, en esta distribución de Linux, el otro es rsyslogd. En concreto, los logs para journald encontrar en /var/log/journal, mientras que rsyslog guarda distintos logs como auth.log (autenticación) o kern.log (kernel). Los archivos de configuración se pueden encontrar en /etc/rsyslog.conf y en /etc/systemd/journald.conf. Abajo podemos observar la salida de los comandos:



```
[10/18/24]seed@VM:~/Downloads$ cat /etc/rsyslog.conf
# /etc/rsyslog.conf configuration file for rsyslog
#
# For more information install rsyslog-doc and see
# /usr/share/doc/rsyslog-doc/html/configuration/index.html
#
# Default logging rules can be found in /etc/rsyslog.d/50-default.conf

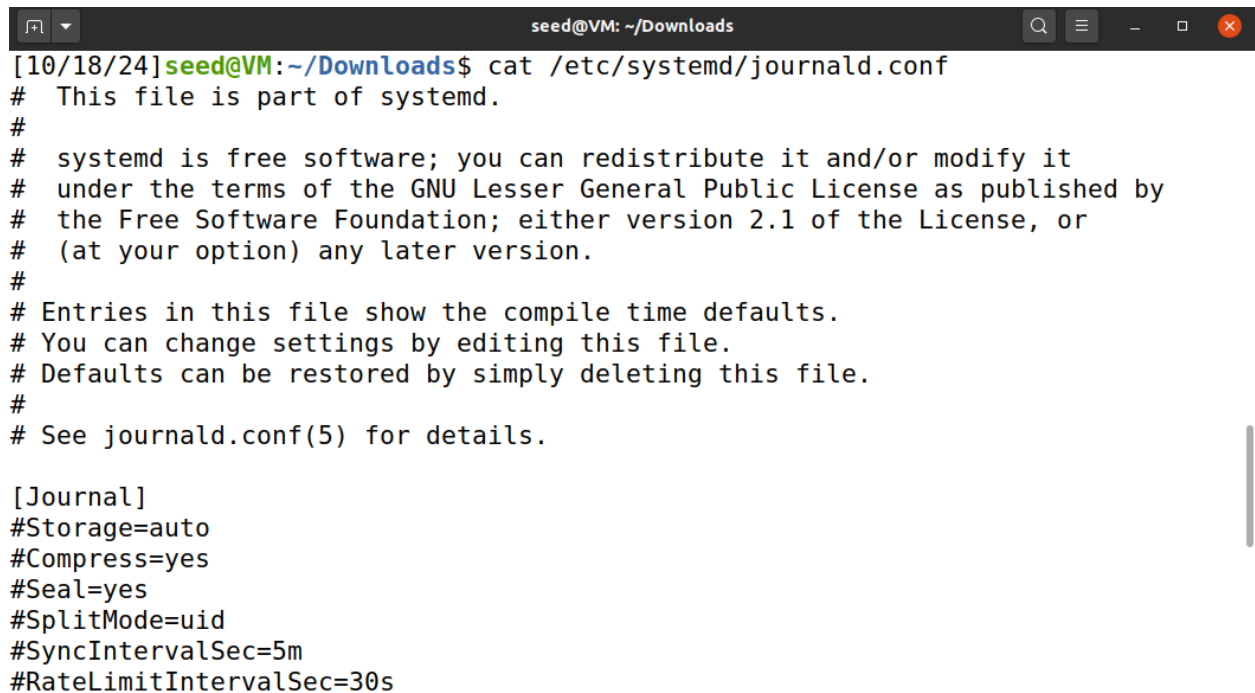
#####
#### MODULES ####
#####

module(load="imuxsock") # provides support for local system logging
#module(load="immark") # provides --MARK-- message capability

# provides UDP syslog reception
#module(load="imudp")
#input(type="imudp" port="514")

# provides TCP syslog reception
```

Ilustración 21 Salida Comandos



```
seed@VM: ~/Downloads
[10/18/24]seed@VM:~/Downloads$ cat /etc/systemd/journald.conf
# This file is part of systemd.
#
# systemd is free software; you can redistribute it and/or modify it
# under the terms of the GNU Lesser General Public License as published by
# the Free Software Foundation; either version 2.1 of the License, or
# (at your option) any later version.
#
# Entries in this file show the compile time defaults.
# You can change settings by editing this file.
# Defaults can be restored by simply deleting this file.
#
# See journald.conf(5) for details.

[Journal]
#Storage=auto
#Compress=yes
#Seal=yes
#SplitMode=uid
#SyncIntervalSec=5m
#RateLimitIntervalSec=30s
```

Ilustración 22 Salida Comandos

- b) Indica cual es la forma de rotado que está programada para cada uno de ellos y cada cuanto se hace**

En este caso, la configuración para el rotado de logs se encuentra en el archivo de configuración logrotate.conf, que se visualiza más abajo. Podemos observar que el rotado de logs se realiza de forma semanal, que se realiza con el usuario de root (administrador), se mantienen los logs durante cuatro semanas, se crean nuevos archivos.


```
[10/07/24]seed@VM:~$ cat /etc/logrotate.conf
# see "man logrotate" for details
# rotate log files weekly
weekly

# use the adm group by default, since this is the owning group
# of /var/log/syslog.
su root adm

# 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
```

Ilustración 23 Configuración rotado

c) Configura un fichero de log nuevo donde se registren sólo los registros de prioridad entre notice y err de cron y de ath

En este caso, primero tenemos que saber que los registros de prioridad entre notice y err son notice, err y warning, en este caso nos piden cron y auth, por lo que creamos un nuevo archivo de configuración en /etc/rsyslogd que incluya dichos registros, que se guardarán en un archivo llamado milog. El archivo se crea con el patrón visto en clase que tienen los archivos de configuración, mensajes de auth y cron, punto como selector para indicar la prioridad y posteriormente el archivo en el que se guardan.

```

seed@VM: ~/rsyslog.d
GNU nano 4.8 miconfiguracion.conf
cron.notice /var/log/milog.log
cron.warning /var/log/milog.log
cron.err /var/log/milog.log
auth.notice /var/log/milog.log
auth.warning /var/log/milog.log
auth.err /var/log/milog.log

[ Wrote 6 lines ]
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Paste Text ^T To Spell  ^_ Go To Line
  
```

Ilustración 24 Configuración Nueva

Luego le damos los permisos necesarios y creamos el archivo de logs en el directorio /var/log (donde se almacenan los logs como se comentó antes), le damos los permisos necesarios (r+w) para que el usuario root pueda escribir en ellos. Después de reiniciar rsyslog, debería funcionar adecuadamente.

```

[10/07/24]seed@VM:/$ cd /etc/rsyslog.d
[10/07/24]seed@VM:~/rsyslog.d$ ls
20-ufw.conf  50-default.conf
[10/07/24]seed@VM:~/rsyslog.d$ nano 50-default.conf
[10/07/24]seed@VM:~/rsyslog.d$ nano miconfiguracion.conf
[10/07/24]seed@VM:~/rsyslog.d$ touch miconfiguracion.conf
touch: cannot touch 'miconfiguracion.conf': Permission denied
[10/07/24]seed@VM:~/rsyslog.d$ sudo nano miconfiguracion.conf
[10/07/24]seed@VM:~/rsyslog.d$ cd /var/log
[10/07/24]seed@VM:~/log$ touch milog.log
touch: cannot touch 'milog.log': Permission denied
[10/07/24]seed@VM:~/log$ sudo touch milog.log
[10/07/24]seed@VM:~/log$ sudo chmod 640 /var/log/milog.log
[10/07/24]seed@VM:~/log$
  
```

Ilustración 25 Asignacion de permisos

d) —Obtén por pantalla las fechas y horas de los últimos 5 reinicios de la máquina.

Intenta mostrar sólo por pantalla la fecha y hora.

Se trata de un comando sencillo, para mostrar los reinicios podemos usar `journalctl` en su opción `list-boots` como se vió en clase. Para mostrar los 10 primeros, es suficiente con el comando `head -n 10`. Además, podemos observar que hasta la fecha hay siempre 38 caracteres, por lo que podemos desprendernos de estos con el comando `cut` para mostrarlos así por pantalla.

```
[10/07/24]seed@VM:~/log$ journalctl --list-boots | cut -c 38- | head -n 10
Tue 2020-11-24 10:38:28 EST-Tue 2020-11-24 10:47:36 EST
Tue 2020-11-24 10:47:42 EST-Tue 2020-11-24 10:48:28 EST
Tue 2020-11-24 11:02:35 EST-Tue 2020-11-24 11:54:16 EST
Tue 2020-11-24 11:54:45 EST-Tue 2020-11-24 11:59:02 EST
Tue 2022-10-18 05:46:57 EDT-Tue 2022-10-18 05:52:15 EDT
Wed 2022-10-19 03:18:51 EDT-Wed 2022-10-19 04:41:42 EDT
Wed 2022-10-26 03:10:05 EDT-Wed 2022-10-26 03:54:00 EDT
Wed 2022-10-26 03:54:15 EDT-Wed 2022-10-26 04:47:48 EDT
Wed 2022-10-26 05:05:59 EDT-Wed 2022-10-26 05:56:53 EDT
Wed 2022-10-26 05:57:10 EDT-Wed 2022-10-26 05:59:19 EDT
```

Ilustración 26 Fecha y Hora

e) Muestra por pantalla los registros del diario de hace 30 días que corresponden con el cron.

Podemos utilizar `journalctl` con los argumentos vistos en clase, `-S`, indicando que son los de hace 30 días, y `-u cron`, en este caso filtrando por la unidad `cron`, obteniendo así los siguientes resultados:

```
[10/07/24]seed@VM:.../log$ journalctl -S "30 days ago" -u cron
-- Logs begin at Tue 2020-11-24 10:38:28 EST, end at Mon 2024-10-07 10:30:41 EDT.
Oct 07 09:36:59 VM systemd[1]: Started Regular background program processing daemon.
Oct 07 09:37:00 VM cron[613]: (CRON) INFO (pidfile fd = 3)
Oct 07 09:37:01 VM cron[613]: (CRON) INFO (Running @reboot jobs)
Oct 07 10:17:01 VM CRON[9183]: pam_unix(cron:session): session opened for user root
Oct 07 10:17:01 VM CRON[9191]: (root) CMD ( cd / && run-parts --report /etc/cron.d)
Oct 07 10:17:01 VM CRON[9183]: pam_unix(cron:session): session closed for user root
Oct 07 10:30:01 VM CRON[9279]: pam_unix(cron:session): session opened for user root
Oct 07 10:30:01 VM CRON[9280]: (root) CMD ([ -x /etc/init.d/anacron ] && if [ ! -f /etc/anacrontab ]; then :; else :; fi)
Oct 07 10:30:01 VM CRON[9279]: pam_unix(cron:session): session closed for user root
```

Ilustración 27 Registros de hace 30 días

- f) **Obtén por pantalla en formato JSON los registros de hoy entre las 8 y las 12 de la mañana del servidor mysql junto con los registros del kernel y que correspondan con una prioridad de tipo "info"**

Este script tiene una dificultad mayor, porque journalctl no proporciona un marco para poder indicar día exacto (today, ya que podemos indicar una fecha, pero habría que cambiar siempre el comando de hoy), por lo que podemos emplear el comando date de Linux para insertar la fecha de hoy. Para indicar la unidad mysql utilizamos de nuevo la opción -u, para los registros del kernel -k, y para la prioridad -p info, que es la prioridad que se nos indica en este caso. Luego indicamos desde cuando y hasta cuando lo queremos, la hora va directamente en el texto y la fecha con el comando date y en formato año-mes-día que es como se pide en journalctl, y por último empleamos el comando -o para el output. Como podemos observar el archivo se crea, pero en nuestro caso está vacío puesto que no hay registros para dicha fecha.

```
[10/07/24]seed@VM:.../log$ sudo su
root@VM:/var/log# journalctl -u mysql -k -p info --since "$(date +%Y-%m-%d) 08:00:00" --until "$(date +%Y-%m-%d) 12:00:00" -o json > minuevolog.json
root@VM:/var/log# ls
alternatives.log      bootstrap.log         gdm3                 speech-dispatcher
alternatives.log.1    bttmp                gpu-manager.log     syslog
apache2               bttmp.1              hp                   syslog.1
apt                   cups                 installer            syslog.2.gz
auth.log              dist-upgrade         journal              syslog.3.gz
auth.log.1            dmesg               kern.log             syslog.4.gz
auth.log.2.gz         dmesg.0              kern.log.1           syslog.5.gz
auth.log.3.gz         dmesg.1.gz          kern.log.2.gz        syslog.6.gz
auth.log.4.gz         dmesg.2.gz          kern.log.3.gz        syslog.7.gz
boot.log              dmesg.3.gz          kern.log.4.gz        ubuntu-advantage.log
boot.log.1            dmesg.4.gz          lastlog              unattended-upgrades
boot.log.2            dpkg.log             milog.log            vsftpd.log
boot.log.3            dpkg.log.1           minuevolog.json     vsftpd.log.1
```

Ilustración 28 Registros

2. REALIZA EL SIGUIENTE SCRIPT EN BASH

El script lo realizamos de una manera muy sencilla, primero chequeando los argumentos, si no hay se piden el archivo y la ip, en el resto de los casos se comprueba si el archivo existe y se asignan las variables. Si se introduce un número incorrecto de argumentos, se muestra el uso adecuado del comando y el programa finaliza. En el caso de que la variable ip no exista (y, por lo tanto, 1 argumento solo), se muestra la salida de las IPs con las correspondientes solicitudes correctas e incorrectas (como se indicó en clase, se considera solo un 200 OK como una solicitud correcta). Se emplea un patrón sencillo para la detección de los códigos de las IPs, buscando 4 números con 3 puntos entre medias y 3 números después de las comillas para el código de solicitud. En el caso de proporcionar una IP, se itera por el archivo, y si la IP se encuentra en dicha línea (si se introduce una ip no válida o no presente en el archivo no habrá coincidencias, y por lo tanto, no se muestra nada) se procede a extraer la información solicitada, buscando http para encontrar la url de referencia hasta el cierre de las comillas y el formato adecuado para la hora en los logs (DD/Mes/AAAA:HH:MM:SS). La última parte de los logs es el user-agent, y este, según el navegador y el dispositivo puede variar en formato, por lo que optamos por ver si hay una coincidencia con los navegadores y sistemas operativos presentes en el log (y algunos que no lo están) que son los utilizados en el 99.9% de los casos y que no nos han dado ningún problema con el archivo, y que resulta más fácil que extraer el user-agent, lo que puede ser un poco complejo. Según se va iterando por el archivo, se imprimen los valores solicitados. Si los logs siguen el formato del proporcionado en la práctica (apache) debería funcionar para cualquiera de ellos. Más abajo podemos observar el código fuente.

```
#!/bin/bash

if [[ $# -eq 0 ]]; then
    read -p "Dime el archivo: " archivo
    if [[ ! -f $archivo ]]; then
        echo "No existe el archivo"
        exit 1
    fi
    read -p "Dime la IP: " ip
elif [[ $# -eq 1 ]]; then
```

```
archivo=$1

if [[ ! -f $archivo ]]; then
    echo "No existe el archivo"
    exit 1
fi

elif [[ $# -eq 2 ]]; then
    archivo=$1
    ip=$2

    if [[ ! -f $archivo ]]; then
        echo "No existe el archivo"
        exit 1
    fi
else
    echo "Uso: $0 [archivo] [IP]"
    exit 1
fi

if [[ -z $ip ]]; then
    ips=()
    bien=()
    mal=()
    total=0

    while read -r linea; do
        if [[ $linea =~ ([0-9]+\.[0-9]+\.[0-9]+\.[0-9]+) ]]; then
            ip_encontrada="${BASH_REMATCH[1]}"
            if [[ $linea =~ "\" ([0-9]{3})\" ]]; then
                codigo="${BASH_REMATCH[1]}"
```

```
    encontrado=0

    for ((i=0; i<total; i++)); do

        if [[ ${ips[i]} == "$ip_encontrada" ]]; then

            encontrado=1

            if [[ $codigo -eq 200 ]]; then

                ((bien[i]++))

            else

                ((mal[i]++))

            fi

            break

        fi

    done

    if [[ $encontrado -ne 1 ]]; then

        ips[total]="$ip_encontrada"

        bien[total]=0

        mal[total]=0

        if [[ $codigo -eq 200 ]]; then

            ((bien[total]++))

        else

            ((mal[total]++))

        fi

        ((total++))

    fi

fi

done < "$archivo"
```

```
for ((i=0; i<total; i++)); do
    echo "IP: ${ips[i]} - Bien: ${bien[i]} - Mal: ${mal[i]}"
done
fi

if [[ -n $ip ]]; then
    while read -r linea; do
        if [[ $linea =~ $ip ]]; then
            if [[ $linea =~ \"(http[^\"]*)\" ]]; then
                url="${BASH_REMATCH[1]}"
            fi
            if [[ $linea =~ ([0-9]{2}/[A-Za-z]{3}/[0-9]{4}:[0-9]{2}:[0-9]{2}:[0-9]{2}) ]];
then
                fecha_hora="${BASH_REMATCH[1]}"
            fi
            if [[ $linea =~ (Windows\ NT|Mac\ OS\ X|Linux|Ubuntu|Android|iPhone) ]]; then
                os="${BASH_REMATCH[1]}"
            fi
            if [[ $linea =~ (Firefox|Chrome|Safari|Opera|Edge|MSIE) ]]; then
                navegador="${BASH_REMATCH[1]}"
            fi
            echo "$url $fecha_hora $os $navegador"
        fi
    done < "$archivo"
fi
```