**Automatización de Redes**

Leer y Escribir Archivos

Índice:

1 – Leer Archivos

2 – Escribir Archivos

1. **Leer Archivos:**

Hasta ahora aprendimos a crear cuadros de diálogos, enviar comandos, y también capturar datos desde el gestor. El próximo paso que me parece algo importante es leer archivos. Si quisiéramos hacer una acción repetitiva en varios equipos, podríamos armar una lista de equipos dentro del código, pero no es la mejor forma de hacerlo. Una vez que nuestro código funciona bien lo mejor es no tocarlo, y además si somos nosotros quienes lo escribimos seguramente resultaría fácil entender el código, pero si le damos nuestro script a un compañero que no entiende nada de programación y le decimos cambia la lista de equipos dentro del código, tal vez podríamos marearlo y no sabría como hacerlo. Por eso este tipo de cuestiones externas al código es mejor manejarlo de otra manera como por ejemplo leer archivos. Aquí va un código de ejemplo, lo que hace es ingresar a los equipos HUAWEI a partir de una lista de equipos en un archivo llamado “inventory.txt” en este caso, envía usuario y password automáticamente, como en otras versiones anteriores que ya hemos hecho y pregunta que versión de equipo es:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# $language = "JScript"

# $interface = "1.0"

/\* "4 - Leer archivo de texto.js"

script para ingresar a los equipos HUAWEI

a partir de una lista de equipos en un archivo

envía usuario y password automáticamente

y pregunta que versión de equipo es \*/

//variables globales

**var** USUARIO = "u564508";

**var** pass = "PASS\_DE\_LAN";

**var** ENTER = "\015";

**var** **NombreArchivo\_Entrada** = "**inventory.txt**"; // nombre de donde lee los datos

**var** **ForReading** = 1, ForWriting = 2, ForAppending = 8; //constantes para archivos

// creo un objeto ActiveXObject

**var** FileOpener = new ActiveXObject ("Scripting.FileSystemObject");

// puntero de archivo de entrada

**var** filedatos = FileOpener.OpenTextFile (**NombreArchivo\_Entrada**, **ForReading**, **true**);

**function** main()

{

**var** EQUIPO, COMANDO; //variables locales a main

crt.Screen.Synchronous = **true**; // sincronizo con el gestor

**EQUIPO** = filedatos.Readline(); //acá leo el EQUIPO

// mientras no llegue al final del archivo...

**while** (filedatos.AtEndOfStream != **true**) {

//llamo a la funcion ingresar al equipo, paso el EQUIPO como un argumento

ingresarAlEquipo(**EQUIPO**);

// una vez adentro del equipo…

**COMANDO** = "**disp version | inc CX**";

crt.Screen.Send ( **COMANDO** + ENTER ); // envía el comando

crt.screen.WaitForString (">"); // espera por el string “>”

crt.Screen.Send ( "quit " + ENTER ); // sale del equipo

crt.screen.WaitForString ("$"); // espera por el string “$”

EQUIPO = filedatos.Readline(); // leo el siguiente equipo

} // cierro el while

crt.Dialog.MessageBox("Fin del script");

filedatos.close(); // cierro el archivo

} // cierro la función main

**function** ingresarAlEquipo(**Equipoaingresar**)

{

**var** COMANDO = "ttelnet " + **Equipoaingresar**;

crt.Screen.Send( COMANDO + ENTER );

// espera hasta que el gestor lea "Username:" o "$"

leido = crt.Screen.WaitForStrings( "Username:" , "$");

**if** ( leido == 1 ){ // si leido es igual a 1 entonces...

crt.Screen.Send( USUARIO + ENTER );

crt.Screen.WaitForString( "Password:" );

crt.Screen.Send( pass + ENTER );

crt.Screen.WaitForStrings("#",">","$");

crt.window.caption = **Equipoaingresar**;

} // cierro la función if

} // cierro la función ingresarAlEquipo

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

En este caso también debo mencionar como está compuesto el archivo “inventory.txt” que use para leer los equipos, su contenido es:

GLF4MU

RSC4MU

BEL5MU

RET5MU

POS3MU

Están elegidos HUAWEI a propósito ya que si alguno fuera un equipo CISCO el comando a enviar sería distinto. En este punto encontramos la primera critica al código:

**¿Podríamos armar un script igual para ambas marcas, CISCO y HUAWEI?**

La respuesta es sí, por supuesto, la imaginación es el límite. Pero sería un código más largo, y por cuestiones de simplicidad y didácticas solo lo hice con HUAWEI. Habría que tener en cuenta un par de cosas. Cuando entramos en un equipo HUAWEI nos damos cuenta porque el formato es “<EQUIPO>” y “>” podríamos simplificar que nos da la pauta que estamos en un HUAWEI, en el caso de CISCO el formato es “EQUIPO#” y es “#” lo que nos indica que estamos en un CISCO, y ¿si fuera un JUNIPER? Bueno en este caso no lo sé porque nunca trabaje con un equipo JUNIPER, y no sé ni un solo comando de este tipo de equipos, así que será un trabajo para ustedes en el futuro, pero tengan en cuenta que la manera de trabajar es la misma, identificar en qué tipo de equipo estamos y enviarle el comando correspondiente:

En HUAWEI será = “disp version | inc CX”

Y en CISCO será = “show version | inc ASR”

Creo que un buen ejercicio podría ser este, modificar el código de "4 - Leer archivo de texto.js" para que funcione tanto en CISCO como HUAWEI. Lo voy a dejar sobre el final de este capítulo para que lo hagan.

Ahora podría mencionarles que cosas nuevas aparecieron en este código:

**var** **NombreArchivo\_Entrada** = "**inventory.txt**"; // nombre de donde lee los datos

**var** **ForReading** = 1, ForWriting = 2, ForAppending = 8; //constantes para archivos

// creo un objeto ActiveXObject

**var** FileOpener = new ActiveXObject ("Scripting.FileSystemObject");

// puntero de archivo de entrada

**var** filedatos = FileOpener.OpenTextFile (**NombreArchivo\_Entrada**, **ForReading**, **true**);

El **NombreArchivo\_Entrada** es solo una variable global en este caso que nos permite escribir el nombre del archivo una sola vez en todo el código, en el resto del código si tuviéramos que llamarlo varias veces, vamos a hablar de simplemente de **NombreArchivo\_Entrada.** “inventory.txt” es un archivo de texto que está ubicado en la misma carpeta donde se encuentra el script, si estuviera ubicado en otra parte habría que decir la dirección completa en la PC, por ejemplo C:\Users\u564508\Documents\Nelson\Trabajo\Scripts\Capacitacion\inventory.txt

Antiguamente a mis archivos de entrada solía llamarlos “input.txt” pero ya que en el código Ansible está mucho más asociado a “Inventory” me pareció bueno introducir este concepto.

En la segunda línea creo unas variables globales **ForReading** = 1, ForWriting = 2, ForAppending = 8, de las cuales solo uso una **ForReading**, cuando abro el archivo de texto, es decir que lo estoy abriendo para lectura. Enviar un 1 en el argumento cuando abro la función o decir **ForReading**, en este caso es lo mismo, pero es mucho más entendible si lo veo de esta manera. Estas 3 variables siempre las tengo a mano cuando manejo archivos, así entiendo a simple vista si quiero usar un archivo para leerlo, escribirlo, o concatenarlo (escribir al final del archivo).

// creo un objeto ActiveXObject

**var** FileOpener = new ActiveXObject ("Scripting.FileSystemObject");

Esta línea de código no quiero entrar en detalle simplemente decir que es necesario crear un objeto ActiveXObject("Scripting.FileSystemObject") para manejar archivos, siempre que manejemos archivos la vamos a ver. Finalmente:

// puntero de archivo de entrada

**var** filedatos = FileOpener.OpenTextFile (**NombreArchivo\_Entrada**, **ForReading**, **true**);

creamos una variable filedatos (o puntero en este caso), donde le indicamos cual es el archivo a abrir, “inventory.txt” en este caso, le decimos que es abrimos para lectura, y el último **true** indica que se pude crear un nuevo archivo en caso que no exista, false es por default.

Algo que no habría que olvidar es que cuando terminamos de usar el archivo se debe cerrar como vemos al final de la función main:

filedatos.close(); // cierro el archivo

Así el script deja de utilizarlo, y no nos genera algún conflicto si dos aplicaciones usan el mismo archivo.

Una vez dentro de la función main lo que vemos de nuevo es:

**var** EQUIPO, COMANDO; //variables locales a main

Las variables son locales en este caso simplemente de manera didáctica para entender nuevos conceptos más adelante.

**EQUIPO** = filedatos.Readline(); //acá leo el EQUIPO

Esta línea me parece interesante es donde leemos una línea desde el archivo “inventory.txt” y esa línea leída la guardamos en la variable EQUIPO, también podríamos leer el archivo entero con la función Read(); pero había que trabajar el código de manera diferente.

// mientras no llegue al final del archivo...

**while** (filedatos.AtEndOfStream != **true**) { // while (condición){}

// código del while…

} // cierro el while

Luego aparece la función **while** típica de cualquier lenguaje de programación, lo que está diciendo es

“mientras el puntero no llegue al final del archivo hacer lo siguiente…”, filedatos.AtEndOfStream Solamente es verdadero (**true**) cuando llega al final del archivo entonces la condición del **while** estaría diciendo mientras filedatos.AtEndOfStream sea diferente de **verdadero**.

Una vez dentro del **while** tenemos:

//llamo a la funcion ingresar al equipo, paso el EQUIPO como un argumento

ingresarAlEquipo(**EQUIPO**);

Es la primera vez que vemos un código con dos funciones, main() e ingresarAlEquipo(), como ingresar a un equipo ya lo vimos en códigos anteriores así que lo voy a explicar muy rápido, aquí simplemente estoy llamando a una función a la cual le paso como argumento cual es el EQUIPO al que quiero entrar, esta realiza su acción y una vez terminado vuelve aquí y continua con el resto del script. Al final del código vemos cual es el código de esta función, lo analizaremos más adelante.

// una vez adentro del equipo…

**COMANDO** = "**disp version | inc CX**";

crt.Screen.Send ( **COMANDO** + ENTER ); // envía el comando

crt.screen.WaitForString (">"); // espera por el string “>”

crt.Screen.Send ( "quit " + ENTER ); // sale del equipo

crt.screen.WaitForString ("$"); // espera por el string “$”

EQUIPO = filedatos.Readline(); // leo el siguiente equipo

Esta parte del código se entiende sola con los comentarios no es nada raro que no hayamos hecho antes. Para mencionar tal vez, esta es la parte que está muy enfocada a HUAWEI, es interesante pensar cómo se modificaría primero para CISCO, y luego para que funcione en ambos casos.

} // cierro el while

Luego de esto se cierra el **while**,y por último en el main:

crt.Dialog.MessageBox("Fin del script");

filedatos.close(); // cierro el archivo

Que ya lo hablamos antes. Ahora vemos rápidamente la otra función

**function** ingresarAlEquipo(**Equipoaingresar**)

{

**var** COMANDO = "ttelnet " + **Equipoaingresar**;

crt.Screen.Send( COMANDO + ENTER );

// espera hasta que el gestor lea "Username:" o "$"

leido = crt.Screen.WaitForStrings( "Username:" , "$");

**if** ( leido == 1 ){ // si leido es igual a 1 entonces...

crt.Screen.Send( USUARIO + ENTER );

crt.Screen.WaitForString( "Password:" );

crt.Screen.Send( pass + ENTER );

crt.Screen.WaitForStrings("#",">","$");

crt.window.caption = **Equipoaingresar**;

} // cierro la función if

} // cierro la función ingresarAlEquipo

Esta función es prácticamente una copia de lo que ya hicimos en “2 - Loggearse.js”, es una función creada por mí que yo le di este nombre, no es propia de javascript o el gestor, por lo que para que pueda ejecutarse necesitamos indicar como funciona, cuál es su código. Lo que hace es recibir como argumento cual es el equipo a ingresar (**Equipoaingresar**), e ingresar al equipo pedido y en vez de volver a escribir estas 12 líneas de código cuando queramos entrar a un equipo con simplemente llamar a nuestra función ingresarAlEquipo(**EQUIPO**) ya estaría resuelto.

1. **Escribir Archivos:**

Vayamos derecho al código y después trato de explicarlo:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# $language = "JScript"

# $interface = "1.0"

/\* "5 - Escribir archivo de texto.js"

script para ingresar a los equipos HUAWEI

a partir de una lista de equipos en un archivo

envia usuario y password automáticamente

y pregunta que version de equipo es

y guarda esa información en un archivo de texto llamado "output.txt" \*/

//variables globales

**var** USUARIO = "u564508";

**var** pass = "PASS\_DE\_LAN";

**var** ENTER = "\015";

**var** NombreArchivo\_Entrada = "inventory.txt"; // nombre de donde lee los datos

**var** NombreArchivo\_Salida = "output.txt"; // nombre de donde escribe los datos

**var** ForReading = 1, ForWriting = 2, ForAppending = 8; //constantes para archivos

// creo un Objecto ActiveXObject

**var** FileOpener = new ActiveXObject ("Scripting.FileSystemObject");

// puntero de archivo de entrada

**var** filedatos = FileOpener.OpenTextFile (NombreArchivo\_Entrada, ForReading, true);

// puntero de archivo de Salida

**var** fileSalida = FileOpener.OpenTextFile (NombreArchivo\_Salida, ForWriting, true);

**function** main()

{

**var** EQUIPO, COMANDO; //variables locales a main

crt.Screen.Synchronous = **true**; //sincronismo

// mientras no llegue al final del archivo...

**while** ( filedatos.AtEndOfStream != **true**) {

**EQUIPO** = filedatos.Readline(); // leo el equipo desde el archivo

//llamo a la funcion ingresar al equipo, paso el EQUIPO como un argumento

ingresarAlEquipo(**EQUIPO**);

**COMANDO** = "disp version | inc CX";

crt.Screen.Send ( **COMANDO** + ENTER ); // envía el comando

capturaryescribir(**EQUIPO**); // **llamo a la función capturar y escribir**

crt.Screen.Send ( "quit " + ENTER ); // saliendo del equipo

crt.screen.WaitForString ("$");

}

crt.Dialog.MessageBox("Fin del script");

filedatos.close(); // cierro los archivos

fileSalida.close();

}

**function** capturaryescribir(**EQUIPO**)

{

**var** Salir = 0;

**var** re = /HUAWEI (\w+-\w+) uptime is [\S\s]+/g; // mi expresión regular "re"

// espera un \n (ENTER) o ">"

Salir = crt.screen.WaitForStrings (String.fromCharCode(13), ">");

**while** ( Salir != 2 ) // mientras Salir sea diferente de 2 hacer..

{

fila = crt.screen.CurrentRow; // fila actual

readline = crt.Screen.Get(fila, 1, fila, 200); // capturo la línea entera

Salir = crt.screen.WaitForStrings (String.fromCharCode(13), ">");

hallado = readline.match(re); // hay coincidencia entre mi re y readline?

**if** ( hallado != null ){ // si hay coincidencia…

// extraigo aquello que está entre paréntesis "()" de readline y lo guardo en DATO

**var** DATO = readline.replace(re, "$1");

// escribo en el archivo de salida el EQUIPO junto con el DATO en este caso el modelo

fileSalida.WriteLine(EQUIPO +" "+ DATO);

} // cierro el if

} // cierro el while

Salir = 0;

} // cierro la función capturaryescribir

**function** ingresarAlEquipo(**Equipoaingresar**)

{

**var** COMANDO = "ttelnet " + **Equipoaingresar**;

crt.Screen.Send( COMANDO + ENTER );

crt.Screen.WaitForString( "Username:" );

crt.Screen.Send( USUARIO + ENTER );

crt.Screen.WaitForString( "Password:" );

crt.Screen.Send( pass + ENTER );

crt.Screen.WaitForStrings("#",">","$");

crt.window.caption = Equipoaingresar;

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Bueno de a poco se va poniendo más complejo, ya estamos en un script de más de 80 líneas de código, pero empieza a ser todo más de lo mismo. Ya no hay una sola función sino que hay 3, main(), capturaryescribir () e ingresarAlEquipo(). Solo tengamos el enfoque en las cosas nuevas:

**var** NombreArchivo\_Entrada = "inventory.txt"; // nombre de donde lee los datos

**var** NombreArchivo\_Salida = "output.txt"; // nombre de donde escribe los datos

al igual que en el código anterior aca simplemente creo una variable global para darle un nombre a mi archivo de salida.

// puntero de archivo de entrada

**var** filedatos = FileOpener.OpenTextFile (NombreArchivo\_Entrada, ForReading, **true**);

// puntero de archivo de Salida

**var** fileSalida = FileOpener.OpenTextFile (NombreArchivo\_Salida, ForWriting, **true**);

y también debemos crearle su puntero fileSalida para trabajar durante el código, como podrán ver, el primero argumento es el nombre del archivo, el segundo en este caso indico que es para escritura (ForWriting o 2 sería lo mismo), y el último **true** indica que si el archivo no existe lo va a crear.

**COMANDO** = "disp version | inc CX";

crt.Screen.Send ( **COMANDO** + ENTER ); // envía el comando

capturaryescribir( **EQUIPO** ); // **llamo a la función capturar y escribir**

El código viene igual que el anterior, hasta llegar a este punto, una vez que enviamos el comando debemos capturar correctamente la información que queremos, y escribirla en el archivo de salida. Lo único que estoy haciendo aquí es llamar a la función capturaryescribir, es una función creada por mí que debo pasarle como argumento el EQUIPO, porque yo definí el EQUIPO como una variable local a main, también para escribir en el archivo necesitamos el puntero del archivo al cual queremos escribir, es decir fileSalida, pero este puntero al ser una variable global puede ser vistas por todas las funciones del código.

crt.Screen.Send ( "quit " + ENTER ); // saliendo del equipo

crt.screen.WaitForString ("$");

}

crt.Dialog.MessageBox("Fin del script");

filedatos.close(); // cierro los archivos

fileSalida.close();

} // cierro la función main

El resto del código de la función main() no presenta ninguna dificultad, es casi idéntico al código anterior, no hay que olvidar cerrar los archivos una vez que terminamos de usarlos. Ahora continuemos con la función capturaryescribir:

**var** Salir = 0;

**var** re = **/HUAWEI (\w+-\w+) uptime is [\S\s]+/g**; // mi expresión regular "re"

Primero creo que una variable Salir que me servirá para el while. Luego creo mi variable “re”, la expresión regular que coincida con mi dato de interés. Como había mencionado antes la web para probar las regular expression es <https://regexr.com/> y tomé como ejemplo:

HUAWEI CX600-X16 uptime is 875 days, 15 hours, 55 minutes

HUAWEI CX600-X16A uptime is 585 days, 16 hours, 29 minutes

HUAWEI CX600-X16A uptime is 517 days, 17 hours, 51 minutes

Básicamente mi expresión regular, ”/HUAWEI (\w+-\w+) uptime is [\S\s]+/g”, lo que dice es:

* Que debe empezar con “HUAWEI “, el espacio del final cuenta también
* Entre paréntesis, mi dato de interés (\w+-\w+):
  + “\w” es un carácter alfanumérico, “\w+” son varios caracteres te este tipo seguidos, es decir una palabra.
  + “-“ simplemente es un guion medio
  + Y nuevamente “\w+” otra palabra
* Luego continuamos con “ uptime is “, debe coincidir con eso exactamente
* Por último [\S\s]+:
  + Los corchetes [] representan un grupo
  + “\S” en mayúscula, es cualquier carácter que no sea un espacio en blanco
  + “\s” en minúscula, es un espacio en blanco
  + El “+” se usa para extender el grupo
  + Pero en su conjunto “[\S\s]+” estoy diciendo esto para que coincida con cualquier texto sea un espacio en blanco o no.

Continuando con el resto del código de la función capturaryescribir:

// espera un \n (ENTER) o ">"

Salir = crt.screen.WaitForStrings (String.fromCharCode(13), ">");

Esta función ya la conocemos, pero tiene un nuevo detalle, String.fromCharCode(13) lo que hace es transformar el carácter ASCII 13 (retorno de carro, o un ENTER dicho mal y pronto) en un string.

**while** ( Salir != 2 ) // mientras Salir sea diferente de 2 hacer..

{

fila = crt.screen.CurrentRow; // fila actual

readline = crt.Screen.Get(fila, 1, fila, 200); // capturo la línea entera

Salir = crt.screen.WaitForStrings (String.fromCharCode(13), ">");

hallado = readline.match(re); // hay coincidencia entre mi re y readline?

Entonces volviendo un poco para atrás ya habíamos enviando el comando, llamamos a nuestra función capturaryescribir, ahora entramos en el while, capturamos la línea desde el gestor, y lo que debemos hacer tomar nuestro dato de importancia en este caso la versión del EQUIPO, pero cuando enviamos el comando la respuesta del gestor es algo larga, así como:

<EQUIPO >disp version | inc CX

Info: It will take a long time if the content you search is too much or the string you input is too long, you can press CTRL\_C to break.

Huawei Versatile Routing Platform Software

VRP (R) software, Version 8.150 (CX600 V800R009C10SPC200)

Copyright (C) 2012-2017 Huawei Technologies Co., Ltd.

HUAWEI CX600-X16A uptime is 517 days, 17 hours, 51 minutes

CX600-X16A version information:

MPU CX6D0MPUB571 version information

MPU CX6D0MPUB571 version information

SFU CX6DSFUIU07B version information:

SFU CX6DSFUIU07B version information:

SFU CX6DSFUIU07B version information:

SFU CX6DSFUIU07B version information:

Para tomar el dato correctamente haber una coincidencia, un match, entre nuestra regular expresión y la línea que estamos leyendo desde el gestor, por ejemplo:

input is too long, you can press CTRL\_C to break.

No tiene nada que ver con

**/HUAWEI (\w+-\w+) uptime is [\S\s]+/g**

En cambio, la línea coloreada en amarillo si lo tiene. Entonces debemos leer cada una de las líneas que nos trae el gestor y compararla contra nuestra regular expression. Eso es lo que está haciendo esta línea:

hallado = readline.match(re); // hay coincidencia entre mi re y readline?

Si encontramos una coincidencia:

**if** ( hallado != null ){ // si hay coincidencia…

Estrictamente lo que está diciendo que la variable hallado no sea nula, la única forma que no sea nula es que la función match haya encontrado una coincidencia.

// extraigo aquello que está entre paréntesis "()" de readline y lo guardo en DATO

**var** DATO = readline.replace(re, "$1");

Ahora sí una vez que tuvimos una coincidencia podemos extraer nuestro dato, esta función replace ya la vimos en “3 - IP del EQUIPO.js”

// escribo en el archivo de salida el EQUIPO junto con el DATO en este caso el modelo

fileSalida.WriteLine(EQUIPO +" "+ DATO);

Finalmente con nuestro dato obtenido escribimos en el archivo de salida con la función WriteLine(“TEXTO a escribir”), funciona prácticamente igual que cuando enviamos un comando crt.Screen.Send ( COMANDO + ENTER ) o crt.Screen.Send ( “TEXTO A ENVIAR” );

} // cierro el if

} // cierro el while

Salir = 0;

} // cierro la función capturaryescribir

Esto último se explica solo con los comentarios. La última función es casi una copia de la que teníamos en el código anterior:

**function** ingresarAlEquipo(**Equipoaingresar**)

{

**var** COMANDO = "ttelnet " + **Equipoaingresar**;

crt.Screen.Send( COMANDO + ENTER );

crt.Screen.WaitForString( "Username:" );

crt.Screen.Send( USUARIO + ENTER );

crt.Screen.WaitForString( "Password:" );

crt.Screen.Send( pass + ENTER );

crt.Screen.WaitForStrings("#",">","$");

crt.window.caption = Equipoaingresar;

}

Con los conceptos vistos hasta ahora:

* Crear cuadros de dialogo
* Enviar y Capturar datos
* Leer y Escribir archivos

Ya tienen una base muy fuerte, para realizar una inmensidad de trabajos de automatización. La mejor forma de aprender es dejar volar un poco la imaginación y meterse a programar. ¿Qué cosas podríamos hacer con lo que aprendimos?

* Imaginemos que tenemos una VRF nueva para desplegar en varios equipos de la red, tanto CISCO como HUAWEI, solo habría que generar un Script con unos comandos específicos para cada marca. Leer los equipos desde nuestro “inventory.txt”, y una vez identificada la marca del equipo, enviar los comandos correspondientes.
* Podríamos crear nuestro propio CORE, hacer una bajada de la configuración de todos los equipos de la red y dejarlo almacenado en una carpeta, para trabajar la información más tarde cuando la necesitemos.
* Podríamos hacer monitoreo del tráfico de la red.
* O incluso hacer un listado único de todos los service policy y qos profile de la RED.

Como había mencionado antes “la imaginación es el límite”.

**FIN**