**Automatización de Redes**

Getting Started

Índice:

1 - Comenzado

2 - Lenguajes de programación en Automatización de Redes

3 - El Primer Script “0 - Hola Mundo.js”

4 - Escribiendo código

5 - Corriendo el script

1. **Comenzando:**

La idea de este curso es entrar en el mundo de automatización de redes, abrir la mente hacia un mundo nuevo y una mejor forma de trabajar.

El cliente SSH utilizado en la mayoría de los casos en Telecom es **SecureCRT** creado por VanDyke Software. Así que voy a basar todos mis proyectos en este.

Una vez establecida nuestra conexión con el gestor, podemos empezar a enviar y recibir información de este, cualquier comando o texto que se escriba por teclado, puede ser automatizado para realizado directamente por la PC, también se puede manipular archivos (leer y escribir).

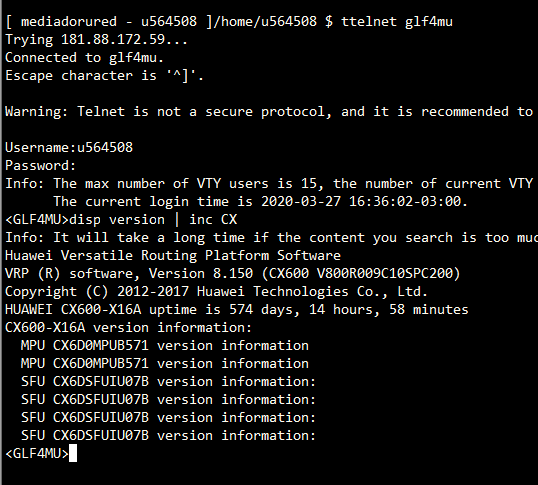


Figura 1

Cuando necesitamos realizar un análisis de la red, entender cómo funciona algún servicio tal vez sea mejor hacerlo a mano, y pensar cual es el mejor comando a utilizar, pero cuando tenemos algún proceso repetitivo una mejor alternativa puede ser automatizarlo. En la Figura 1, lo que hice fue ingresar al equipo GLF4MU (**ttelnet glf4mu**) y luego de loggearme (Username: u564508 , Password: XXXXXXX) preguntar cuál es la versión del equipo ( **disp version | inc CX** ).

Y está perfecto hacerlo manualmente si son pocos casos, pero ¿si lo queremos hacer en muchos equipos diferentes? Por ejemplo, tenemos la siguiente lista de equipos:

GLF4MU

BAL4MU

CLI4MU

CAP3MU

GLF5MU

TIR3MU

Tal vez se puede seguir pensando en hacerlo manualmente ¿pero y si son 200 equipos o 1000? La alternativa automatizada empieza a tener más sentido. Para no complicar las cosas, elegí en mi lista todos equipos HUAWEI, para que todos los comandos sean iguales.

Así que vamos a crear una receta (o algoritmo) para tratar de encontrar la solución a este problema. Para ver la versión del equipo son los siguientes pasos:

1. Ingresar al equipo
2. Loguearse
3. Tirar el comando para ver la versión
4. Salir del equipo

Una vez finalizado el paso último paso hay que repetir el proceso hasta el terminar con el ultimo equipo de la lista, en nuestro ejemplo TIR3MU. Ahora veamos cada uno de los pasos con más detalle, voy a explicarlos en orden inverso por la complejidad que presentan:

d) Salir del equipo

Aquí el comando es siempre el mismo por tratarse de equipos HUAWEI sería:

Enviar “**quit**”

c) Tirar el comando para ver la versión

Este es uno de los pasos más simples ya que el comando es siempre el mismo, aquí podríamos, por ejemplo, crear una variable llamada **COMANDO**:

**var COMANDO = “**disp version | inc CX”;

entonces lo que debemos hacer es:

enviar **COMANDO** y luego apretar (o enviar) un Enter.

b) Loguearse

Creo dos variables y les doy su valor correspondiente:

**var** **Usuario** = “u564508”;

**var** **Pass** = “password\_de\_ejemplo”;

Entonces cuando en el gestor aparezca “Username:” envió mi **Usuario** y cuando aparezca

“Password:” voy a enviar en mi **Pass**.

1. Ingresar al equipo

Se debe enviar el comando: ttelnet **EQUIPO**

Puesto que no tenemos un único equipo, **EQUIPO** representa una variable, o mejor dicho una lista de variables (conocido como array), donde:

**var** **EQUIPO** = [ // var de variable, en este caso representa un array llamado EQUIPO

“GLF4MU”, // GLF4MU es la primera variable de la lista en la posición 0

“BAL4MU”, // BAL4MU es la 2da variable del array en la posición 1

“CLI4MU”, // CLI4MU es la 2da variable del array en la posición 2

“CAP3MU”, // CAP3MU es la 2da variable del array en la posición 3

“GLF5MU”, // GLF5MU es la 2da variable del array en la posición 4

“TIR3MU” // TIR3MU es la 2da variable del array en la posición 5

];

Por lo tanto en la primera vuelta debemos tomar la primera posición del array en este caso EQUIPO[0] que representa a GLF4MU, en la segunda vuelta sería EQUIPO[1] que representa a BAL4MU y así sucesivamente.

Una vez realizados todos estos pasos en el orden correcto a, b, c, d volvemos a repetir el proceso hasta terminar con la última posición de la lista de **EQUIPO**.

1. **Lenguajes de programación en Automatización de Redes:**

Si entendimos un poco como funcionaba nuestra receta, o algoritmo del apartado anterior ya podríamos animarnos a hacer nuestro primer Script, y antes de profundizar demasiado en la teoría a mí me gusta tomar ejemplos concretos que ya funcionan, y empezar a modificarlos según mis necesidades.

En la siguiente web de Vandyke, creador del SecureCRT:

<https://www.vandyke.com/support/index.html>

podrán ver una sección **Automation Academy** dedicada a este tema y con ejemplos en diferentes lenguajes. Yo voy a mencionarles 4:

* Ansible
* Visual Basic Script
* Python
* Javascript

También si tengo que hablar de una página puramente de código, ya por el fuera de automatización de redes les recomiendo:

<https://www.w3schools.com/>

Todos estas páginas y temas están en inglés por si necesitan traducirlas.

Volviendo a los lenguajes:

**Ansible:**

Ansible es una plataforma de software libre para configurar y administrar ordenadores. Nativamente utiliza YAML para describir configuraciones reusables de los sistemas.

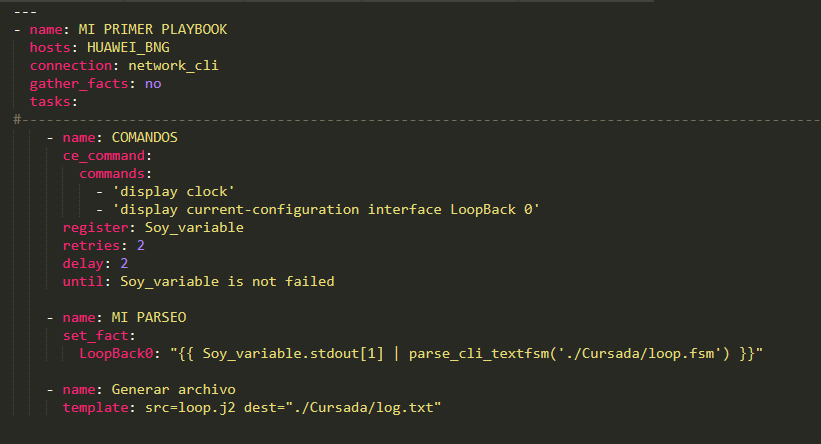


Figura 2

La Figura 2 muestra un ejemplo en Ansible, el formato en que se escribe YAML, está creado de tal manera que sea bastante entendible al ojo humano. Ansible en un lenguaje enfocado totalmente a automatización de redes y tal vez en unas simples líneas de código se puede resolver gran cantidad de problemas, pero YAML presenta un formato al cual no estoy acostumbrado, es muy riguroso ya que la cantidad de espacios que hay que dejar por ejemplo tiene que ser precisa. Yo no soy un experto en este tema solo hice algunos casos de prueba para entender cómo funciona. Así que prefiero antes de hablar de Ansible dejarles los referentes en Telecom que saben mejor de este lenguaje:

* Alejandro Ariel Medicina
* Bruno D'Alia
* Carlos Leandro Sampayo Baus

A continuación, el manual de Ansible creado por Alejandro Ariel Medicina con el que me inicie en este tema:



**Ansible fue el lenguaje elegido por Telecom para realizar tareas de automatización**, todos en la empresa deberíamos ir por este camino. Como les mencionaba antes tiene algunos beneficios como ser bastante entendible a simple vista, y tener funciones concretas para redes simplificándolo a pocas líneas de código, pero también presenta algunas complicaciones, tal vez más burocráticas, para trabajar con Ansible deben pedir acceso al servidor de Ansible (se pide a Bruno D'Alia o Carlos Sampayo), necesitan una ip fija para trabajarlo, o trabajar desde su máquina virtual (Vmware).

Así que siguiendo mi propio criterio preferiría no entrar en el tema Ansible, por más que Telecom diga lo contrario y si quieren conocer más de este lenguaje pueden consultarlo con los referentes antes mencionados.

**Visual Basic Script (.vbs):**

En la página de ejemplos de scripts de vandyke

<https://www.vandyke.com/support/securecrt/scripting_examples.html>

hay un montón de documentación al respecto, fue el lenguaje con que yo entré al mundo script, y te da algunos ejemplos como:

- Crear cuadros de dialogo

- Enviar comandos e información a un Server

- Obtener datos de la salida de un Server

…entre otros.

La mayoría de los scripts que yo creé (Nelson) están hechos en este lenguaje, si tienen algunos de mis scripts en sus máquinas seguramente será es un “.vbs”.

Punto a favor que tiene visual basic script es que hay mucha documentación en internet dando vuelta para tener de referencia, a diferencia de Ansible el lenguaje “.vbs” podrías programarlo hasta en un simple bloc de notas y luego cambiar la extensión de “.txt” a “.vbs”, yo prefiero utilizar Notepad++ ya que te colorea las palabras claves del lenguaje elegido y te numera cada línea de código. Punto en contra que yo le veo al lenguaje, es que es algo viejo, y si bien funciona bien en automatización de redes, cuando lo queremos usar para otros propósitos no nos sirve demasiado.

Pueden mirar cómo están hechos mis .vbs pero tampoco quiero enfocarme en este lenguaje, prefiero estar un poco más a la vanguardia y usar cosas más modernas.

**Python (.py):**

Como podrán ver en la web de vandyke

<https://www.vandyke.com/support/securecrt/python_examples.html>

Se pueden hacer scripts con Python, creo que mucho del futuro de la programación pasa mucho por este lenguaje, sería mi lenguaje preferido para enseñarles salvo por el único detalle que la versión que tenemos en nuestras máquinas no acepta Python. La Versión del SecureCRT que yo tengo en mi máquina es la 5.2.2, si ustedes tienen alguna versión superior u otro cliente SSH que no sea el SecureCRT podríamos probar si se puede correr algún script de Python.

Lo dejo solo como comentario entonces, que Python permite hacer scripts de automatización de redes, por si en algún futuro se les ocurre utilizarlo.

**Javascript (.js):**

En la misma página que el caso de vbs

<https://www.vandyke.com/support/securecrt/scripting_examples.html>

encontramos ejemplos de este lenguaje, pero ya en menor cantidad.

Si bien es más difícil encontrar ejemplos e información de Automatización de Redes con Javascript que con Visual Basic Script, Javascript es un lenguaje muy usado en el mundo de la programación y que todavía sigue muy vigente.

Hay infinidad de documentación de javascript en internet, como lenguaje, no de Automatización, ya que por ejemplo un gran porcentaje de las páginas web se programan en Javascript. También es simple de programar, podríamos hacerlo al igual que .vbs en un bloc de notas o en notepad++.

**Por todas características, me gustaría enseñarles el mundo de Network Automation con el lenguaje Javascript.**

1. **El Primer Script “0 - Hola Mundo”:**

Como en la gran mayoría de casos lenguajes de programación, vamos a arrancar con el ejemplo de “Hola Mundo”, el objetivo de este ejemplo escribir en pantalla precisamente eso “Hola Mundo”, vamos al código:

# $language = "JScript"

# $interface = "1.0"

// el objetivo de este ejemplo escribir en pantalla “Hola Mundo”

**function** main()

{

crt.Dialog.MessageBox("**Hola Mundo**");

}

Ahora voy a explicar línea por línea para que entiendan que es lo que está haciendo.

**Líneas 1 y 2:**

# $language = "JScript"

# $interface = "1.0"

Esto siempre va estar en todos los scripts, por lo que entiendo la línea 1 dice el lenguaje en el que va a trabajar (ya que como vimos antes podemos programar en distintos lenguajes) y el “interface 1.0” lo vi siempre, así que estas dos líneas tienen que ir sí o sí.

**Línea 3:**

// el objetivo de este ejemplo escribir en pantalla “Hola Mundo”

Lo que vemos acá es un **comentario,** esto yo lo uso muchísimo, cuando se está corriendo un script, la máquina ignora los comentarios, para la PC da igual que esté o no esté, pero para el ojo humano ayuda muchísimo a entender que es lo que se está haciendo. El formato de los comentarios en **Javascript** es el siguiente:

// esto es un comentario de una sola línea

Cualquier\_codigo () // se agrega “//” al inicio y el resto de la línea será ignorada

/\* esto es un comentario multilínea

A partir que en el código se encuentra “/\*” empieza a ignorar todo el resto hasta encontrar el cierre “\*/”

\*/

**Líneas 4 y 5:**

**function** main()

{

Acá está declarando donde empieza la función principal “main()”, es la base de todo el código. Y ya que estamos en este punto podemos decir como es la estructura de una función, arranca con **function** luego viene el nombre de la función en este caso es **main** pero podría ser cualquiera Nelson, Telecom, Huawei, CISCO, sumarNumeros. Luego viene “()” lo que va ahí son los argumentos de la función en este caso no tenemos ninguno por esto está vacío. Pero si quisiéramos sumar dos números que ingresamos por teclado por ejemplo podríamos crear la siguiente función:

**function** sumarNumeros(n1, n2) { // la function sumarNumeros tiene argumentos n1 y n2

return n1 + n2; // aquí suma n1 y n2 y retorna el resultado

} // se cierra la function sumarNumeros

Entonces luego de los argumentos tenemos **“{“** que indica donde va a arrancar el código propio de esa función.

**Líneas 4:**

crt.Dialog.MessageBox("**Hola Mundo**");

La línea 4 lo que hace es crear un cuadro de dialogo con el texto “Hola Mundo”, cualquier texto que queramos ingresar va entre “”. Notar también lo que hablaba anteriormente crt.Dialog.MessageBox podría decirse que es una función y su argumento es el texto (o string) “Hola Mundo”, otra alternativa de hacer lo mismo es crear una variable cualquiera asignarle un valor, algún texto y pasar esa variable como argumento de la función, como en este caso:

**var** **texto** = "**Automatización**"; // creo la variable texto y le asigno un valor

crt.Dialog.MessageBox(**texto**); // Le paso la variable texto como argumento

// estas dos líneas de código hacen lo mismo que lo anterior

**Línea 5:**

}

La última línea lo que hace es cerrar la función main, como vimos antes la estructura de una función es:

**function** Nombre\_de\_la\_funcion(argumentos) { // declaro la función

// aquí va el código de la función

} // aquí cierro la función

1. **Escribiendo código:**

Para hacer funcionar nuestro código podemos simplemente copiar el código:

# $language = "JScript"

# $interface = "1.0"

// el objetivo de este ejemplo escribir en pantalla “Hola Mundo”

**function** main() {

crt.Dialog.MessageBox("**Hola Mundo**");

}

Y pegarlo en un bloc de notas, lo guardamos y ¡listo!, el gestor entiende el script por más que lo guardemos como un archivo “.txt”. Por una cuestión de prolijidad voy a explicarlo como habría que hacerlo bien, los archivos JavaScript usan la extensión “.js”, para escribir código yo uso el **Notepad++** es un software súper sencillo y lo pueden encontrar en la web de aplicaciones de Telecom **Tapps:**

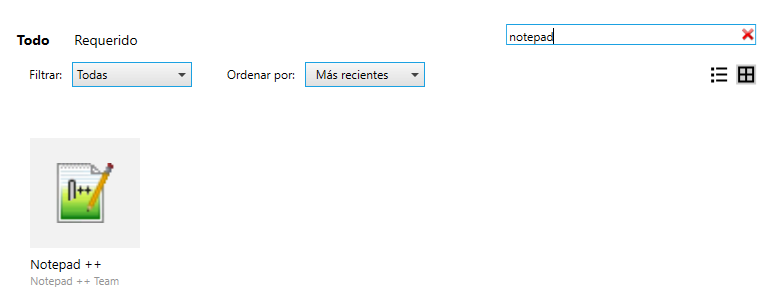


Figura 3

Una vez instalado pueden ver la interface bien simple, es muy parecido al bloc de notas, con la ventaja que si van a la solapa **Language >> J >> Javascript** el Notepad++ les colorea directamente las palabras claves propias del código, y los comentarios, otra ventaja es que te numera cada una de las líneas a la izquierda del texto como vemos en la siguiente imagen.

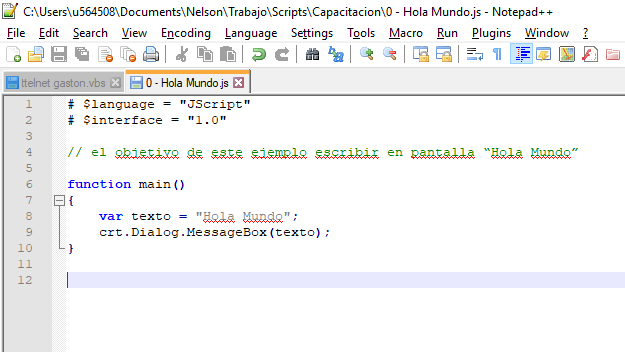


Figura 4

En caso de abrir un archivo ya existente, notepad++ detecta automáticamente de que lenguaje estamos hablando por la extensión del mismo, en este caso al ser de extensión “.js” sabe que es código Javascript.

Muy bien guardamos entonces, y ya tenemos el código en un archivo.

1. **Corriendo el Script:**

Esto seguramente ya lo usan todo el tiempo, pero por las dudas lo explico para completar información de esta capacitación. Una vez que tenemos nuestro código finalizado y guardado, vamos al gestor a la solapa **Script >> Run…**

Buscamos nuestro archivo script recién creado y le damos **Run**

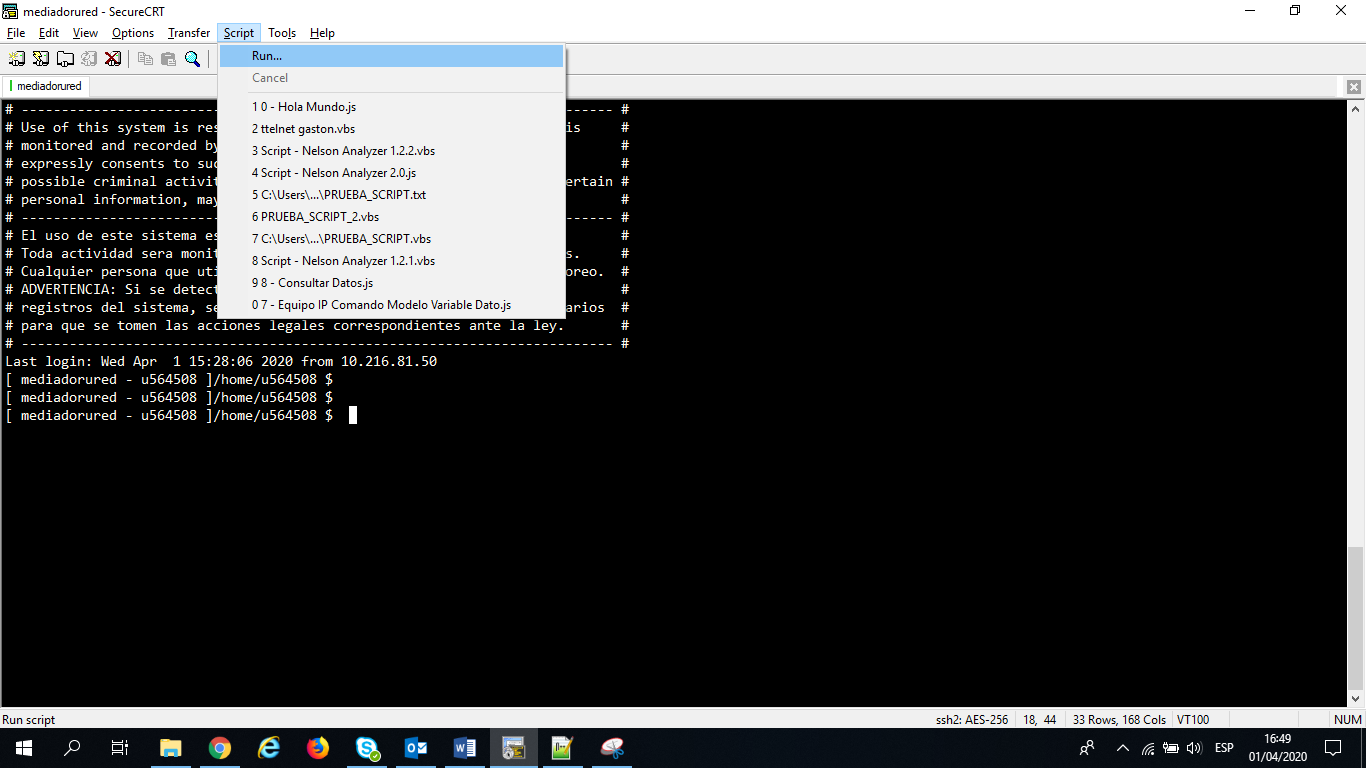


Figura 5

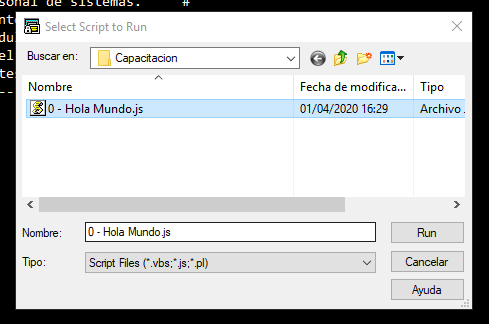


Figura 6

Y el resultado que obtenemos es el siguiente:

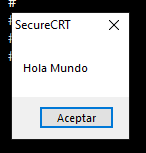


Figura 7

**FIN**