**Automatización de Redes**

Enviar y Capturar datos

Índice:

1 – Cuadros de diálogo (breve repaso)

2 – Enviar Comandos

3 – Capturar datos

1. **Cuadros de diálogo (breve repaso):**

Como vimos en el primer capítulo anterior en el ejemplo de “Hola Mundo”:

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

# $language = "JScript"

# $interface = "1.0"

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

**function** main()

{

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

}

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

Podemos crear cuadros de diálogos llamando a la siguiente función:

crt.Dialog.MessageBox("**cualquier texto**"); // el texto debe estar entre “”

recordar cada vez que se termina escribir una línea de código hay que cerrarla con punto y coma “;”, como la línea anterior. Otro punto a repasar es como crear una función:

**function** main(){ // function NOMBRE\_de\_FUNCION(argumentos){

// aquí va el código de la función

} // cerramos la función

1. **Enviar comandos:**

Para hacerlo de una manera más rápida voy a tratar de explicar el código directamente comentando desde el ejemplo dado. A continuación “1 - Enviar comando.js”:

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

# $language = "JScript"

# $interface = "1.0"

// Este Script envia el texto "Network Automation Rules!!!"

**function** main()

{

// sincronizo con el gestor

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

// creo la variable COMANDO y le doy el valor "Network Automation Rules!!!"

**var** COMANDO = "**Network Automation Rules!!!**";

**// envia el comando**

**crt.Screen.Send( COMANDO );**

// creo un cuadro de dialogo indicando el fin del script

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

}

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

Como podrán ver la función a llamar para enviar comandos es:

crt.Screen.Send( COMANDO );

o lo que es lo mismo

crt.Screen.Send( “cualquier texto” );

Ahora vamos a hacer un poco más complejo el ejemplo para algo útil en la realidad, vamos a hacer un script para loggearnos en los equipos y que automáticamente envíe nuestro Usuario y Clave de LAN,

"2 - Loggearse.js":

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

# $language = "JScript"

# $interface = "1.0"

// script para ingresar a los equipos

// el usuario y password están guardados en variables globales

// deben ser modificados cada vez que cambiamos nuestra clave de LAN

**var** **ENTER** = **"\015"**; // varible global ENTER para indicar cuando enviar un enter

**var** **USUARIO**, **pass**; // creo las variables globales USUARIO y pass

// las variables globales pueden ser vistas por todas las funciones del código

**USUARIO** = "**u564508**"; // le asigno el valor de mi u564508 a USUARIO

**pass** = "**Pass\_de\_LAN**"; // le asigno el valor clave de LAN a pass

**function** main()

{

**var** **EQUIPO**; // creo la variable EQUIPO, esta variable es local a main()

**var** **leido**; // creo la variable leido

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

// pregunto por el equipo a ingresar

**EQUIPO** = crt.Dialog.Prompt("Ingrese el equipo", "loggearse", "RSC4MU", 0);

// los argumentos son ("texto del cuadro", "titulo", "ejemplo", tipo 0 sin ocultar)

// tipo 1 oculto

// creo la variable COMANDO con el valor "ttelnet " y concateno el EQUIPO

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

crt.Screen.Send( **COMANDO** + **ENTER** ); // envía el COMANDO y también un ENTER

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

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

if (**leido**!= 2){// si leído es distinto de 2 o sea encontró Username: hace..

crt.Screen.Send( **USUARIO** + **ENTER** ); // envía el USUARIO y un ENTER

crt.Screen.WaitForString( "Password:" ); // espera hasta leer "Password:"

crt.Screen.Send( **pass** + **ENTER** ); // entonces envía la pass y un ENTER

// espera por algunos de los siguientes "#",">","$"

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

crt.window.caption = **EQUIPO**; // pone el EQUIPO a la solapa del gestor

} // cierro la función if “si (condicion) entonces {}”

} // cierro la función main

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

En este caso usamos el recurso aprendido de enviar comandos:

crt.Screen.Send( **COMANDO** );

Pero también aparecieron otros nuevos conceptos como las variables globales, aquellas que pueden ser vistas por todas las funciones del código. Imaginemos el siguiente código con dos funciones:

# $language = "JScript"

# $interface = "1.0"

**var Global**; // creo la variable Global

Global = 25; // le doy el valor 25

**function** main(){

// aquí va el código de la función main por ejemplo:

**var otra\_variable = Global** - **10**; // creo otra variable local = 15

crt.Dialog.MessageBox(**Global**); //va a mostrar en pantalla 25

crt.Dialog.MessageBox(**otra\_variable**); //va a mostrar en pantalla 15

sumar(); // invoco a la función sumar()

} // cierro la función main

**function** sumar(){ // creo la función sumar

// aquí va el código de la función sumar:

**var variable\_local = Global - 20**; // creo una variable local = 5

crt.Dialog.MessageBox(**variable\_local**); //va a mostrar en pantalla 5

//si quisiera mostrar desde aca a **otra\_variable**

crt.Dialog.MessageBox(**otra\_variable**); //esta línea generaría error

/\* ya que otra variable es una variable local de la función main

No de esta función sumar, por lo tanto, no puede ser vista desde aquí\*/

}

Una alternativa nueva que apareció de hacer cuadros de dialogo fue para el caso que necesitemos ingresar datos por teclado, como en este caso el equipo:

// pregunto por el equipo a ingresar

**EQUIPO** = crt.Dialog.Prompt("Ingrese el equipo", "loggearse", "RSC4MU", 0);

// los argumentos son ("texto del cuadro", "titulo", "ejemplo", tipo 0 sin ocultar)

// tipo 1 oculto

El texto ingresa lo almacena en la variable EQUIPO, en los comentarios está explicado como es el formato de los argumentos, en caso de poner uno al final el ejemplo aparecería como “\*\*\*\*\*\*”, esta forma es útil cuando necesitamos ingresar una password por teclado.

Respecto a esperar hasta leer un texto especifico (un STRING) en el gestor aparecieron estas dos líneas de código:

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

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

notar que en el primer caso habla de WaitForStrings (esperar por los strings, lo dice en plural), si encuentra “Username:” leído toma el valor de 1 y si encuentra “$” leído toma el valor 2. En el segundo caso tenemos WaitForString (esperar por el string, en singular), así que aquí el código se detiene hasta leer en el gestor "Password:", en cuanto lo encuentra continúa ejecutándose las siguientes líneas de código.

También apareció por primera vez la función **if** , elementar en cualquier código de programación:

if (**leido**!= 2){}

Su estructura es la siguiente :

if (**condición**){ // si sucede la siguiente condición hacer … {

// aca va el código en caso que se cumpla la condición

// en nuestro caso que leído se diferente de 2

} // cierro la función if

Por último, tenemos la siguiente línea:

crt.window.caption = **EQUIPO**; // pone el EQUIPO a la solapa del gestor

como está indicado en los comentarios a la solapa del gestor donde estamos trabajando le pone el valor de EQUIPO que habíamos ingresado por teclado.

1. **Capturar datos:**

Veamos el siguiente código y luego paso a explicarlo, es simplemente una modificación del código anterior, "2 - Loggearse.js", donde capturamos la ip del equipo al que estamos ingresando. Para que el código sea más fácil de leer, solo voy a comentar aquellas cosas nuevas que aparecen:

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

# $language = "JScript"

# $interface = "1.0"

// "3 - IP del EQUIPO.js"

// script para ingresar a los equipos

// envía usuario y password automáticamente

// y muestra en pantalla cual es la IP del EQUIPO al que estamos ingresando

var ENTER = "\015";

var USUARIO, pass;

USUARIO = "u564508";

pass = "Golf0420";

**function** main()

{

**var** EQUIPO;

**var** leido;

crt.Screen.Synchronous = true;

EQUIPO = crt.Dialog.Prompt("Ingrese el equipo", "loggearse", "RSC4MU", 0);

var COMANDO = "ttelnet " + EQUIPO;

crt.Screen.Send( COMANDO + ENTER );

// espera hasta que el gestor lea "Trying " o "$"

leido = crt.Screen.WaitForStrings( "**Trying** " , "$");

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

**var** **fila** = crt.screen.CurrentRow; // fila actual donde está el cursor

crt.Screen.WaitForString( "..." ); // espera leer el string "..."

**var** **readline** = crt.Screen.Get(fila, 1, fila, 200);

// captura la línea leída en readline desde la columna 1 hasta la 200

// y sobre la fila actual donde está el cursor

**var** **re** = **/Trying\s(\d+\.\d+\.\d+\.\d+)...\s+/g**;

/\* creo la expresión regular "re" y le doy un valor particular de lo que quiero capturar, más adelante explico en detalle de que trata esto\*/

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

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

crt.Dialog.MessageBox(**DATO**); // muestro el DATO en pantalla, o sea la IP

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

crt.Screen.Send( USUARIO + ENTER );

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

crt.Screen.Send( pass + ENTER );

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

crt.window.caption = EQUIPO;

}

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

} // cierro la función main()

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

Analicemos ahora lo nuevo que aparece:

// espera hasta que el gestor lea "Trying " o "$"

leido = crt.Screen.WaitForStrings( "**Trying** " , "$");

En lugar de esperar por **"Username: "** en este caso esperamos por **"Trying "** que es la palabra que aparece cuando está intentando acceder a un equipo y nos muestra su IP.

if ( leido == 1 ){}

Lo que dice ahí es “si leído es igual a 1 entonces hacer…” a diferencia del caso anterior la condición del if pregunta si es igual y no diferente como el caso anterior. Cuando necesitamos determinar si una condición es igual se usa doble igual “==”, y para el resto de los casos les dejo esta pequeña tabla:

**Operador** **Descripción**

== Es igual a

=== Igual valor e igual tipo

!= Distinto

!== Distinto valor o distinto tipo

> Más grande que

< Más pequeño que

>= Mayor o igual

<= Menor o igual

**var** **fila** = crt.screen.CurrentRow; // fila actual donde está el cursor

Como indica en los comentarios estamos guardando en la variable fila cual es la fila donde estamos parados actualmente con el cursor.

crt.Screen.WaitForString( "..." ); // espera leer el string "..."

**var** **readline** = crt.Screen.Get(fila, 1, fila, 200);

// captura la línea leída en readline desde la columna 1 hasta la 200

// y sobre la fila actual donde está el cursor

Aquí es donde capturamos la información, en la fila actual, desde la columna 1 hasta la 200, de manera resumida podríamos decir que es toda la fila donde está parado el cursor. Para capturar un dato específico podríamos decir que lo haga desde la columna 10 hasta la 23 por ejemplo, pero como los datos pueden ser muy diversos y no tener una cantidad de caracteres exacta en todos los casos es mejor trabajarlo con **Expresiones Regulares**:

**var** **re** = **/Trying\s(\d+\.\d+\.\d+\.\d+)...\s+/g**;

/\* creo la expresión regular "re" y le doy un valor particular de lo que quiero capturar, más adelante explico en detalle de que trata esto\*/

Bueno aquí me tengo que detener un poco a explicar que es una expresión regular. Supongamos estos 3 casos:

GLF4MU = Trying 181.88.172.59...

BEL5MU = Trying 181.88.172.181...

RSC4MU = Trying 181.88.69.172...

¿Cómo hacemos para entender que es una IP? Porque la vemos a simple vista y nuestro ojo humano puede decir “¡ha! Eso es una IP”, pero para que lo entienda la máquina tenemos que ser más detallista. Una IP tiene la siguiente forma o estructura **“un número, un punto, un número, un punto, un número, un punto y por último un número”**, podríamos ponernos más riguroso y decir que ese número debe estar entre 0 y 255 pero no quiero entrar en tanto detalle.

Entonces la expresión regular que estamos buscando tiene la siguiente forma “una palabra o texto, un espacio, (**un número, un punto, un número, un punto, un número, un punto y un número**), y por último 3 puntos seguidos”, pongo la IP a propósito entre paréntesis () porque es el dato de importancia que necesitamos capturar. Una expresión regular lo que nos está indicando es que forma tiene un texto que estemos buscando. Acabamos de definir nuestra expresión en palabras simples para poder entenderlo, pero ya existen definiciones de este tipo de cosas a saber dejo algunas de las más importantes:

**Carácter** **Significado**

(x) Encuentra x y recuerda la búsqueda. Se llaman paréntesis de captura. Es el dato de importancia a capturar.

Por ejemplo, /(foo)/ encuentra y recuerda 'foo' en "foo bar."

\d Todo aquello que sea un **número**. Coincide con un digito básico de 0-9. Si queremos que coincida con un número más grande como por ejemplo 25 o 139 debemos agregarle el “+”

Es decir “\d+”

\s Coincide con un **espacio** “ “

\w Coincide con cualquier carácter alfanumérico. Equivalente a [A-Za-z0-9\_]. “\w” es para un solo carácter alfanumérico, si queremos que coincida con una **palabra** entonces es “\w+” como lo vimos con los números

\. Para algunos caracteres especiales hay que marcarlos bien definidos, en este caso \. Coincide con un **punto** “.”, y si quisiéramos que coincida con un “+” sería \+

Existen muchas más definiciones de expresiones regulares, les dejo una página web a continuación que es la que yo uso para trabajar con esto:

<https://regexr.com/>

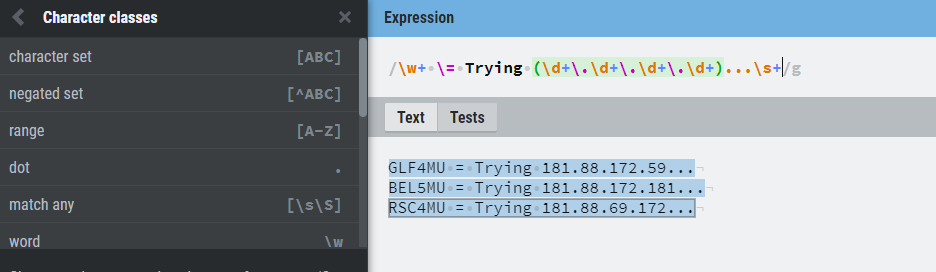


Figura 1

Funciona de la siguiente manera en la parte de “Text” ponemos ejemplos de texto a probar, y arriba en “Expression” empezamos a armar la expresión regular de la manera que corresponde para el código. En el apartado de la izquierda también podemos encontrar cual es la forma correcta para cada caso, como “\w” que lo explicamos arriba, entre otros.

Volviendo al principio de nuestra regular expression:

**var** **re** = **/Trying\s(\d+\.\d+\.\d+\.\d+)...\s+/g**;

Lo que estoy haciendo es crear una variable llamada “re” que es una regular expresión (las regular tienen esta forma “/ lo que sea/g”, g de global flag) y lo que estoy diciendo es:

/ empiezo la regular expression

Trying matchea con la palabra “Trying”

\s matchea con “un espacio en blanco”

() entre paréntesis lo que me interesa

\d+ matchea con un número

\. matchea con un punto

… matchea con 3 puntos seguidos

\s+ matchea con todos los espacios en blancos que encuentre al final

/g cierro la expresión regular y le pongo el flag global

Fue un poco largo, pero al final podemos responder la pregunta anterior, ¿Cómo hacemos para entender que es una IP? Para la máquina esto es una IP: **\d+\.\d+\.\d+\.\d+**

Ahora si continuando con el resto del código tenemos:

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

Por un lado, tenemos en **readline** capturado lo que leímos desde el gestor esa fila actual desde la columna 1 a la 200, ¿lo recuerdan? y por otro lado en **re** tenemos nuestra regular expression que tiene entre () el dato de importancia que queremos capturar, como es un único () se identifica con “$1” si habría más de uno el segundo sería “$2” y así sucesivamente. Lo que hace esta línea de código es comparar a nuestra línea leída **readline** con nuestra expresión regular **re**, primero que nada, debe haber una coincidencia, luego si coinciden reemplaza lo que tenemos entre paréntesis en este caso la IP (o “$1”) y lo guarda en la variable **DATO**.

crt.Dialog.MessageBox(**DATO**); // muestro el DATO en pantalla, o sea la IP

Finalmente mostramos ese dato en pantalla, el resto del código es igual al del ejemplo anterior.

**FIN**