Revisión	Fecha	Comentario	Autor
0	19/01/2010		Ulises Bigliati

ÍNDICE

Introducción

Este breve trabajo es en realidad una simplificación de notas anteriores, particularmente la CoAN-007, y está motivado por solicitudes de usuarios que buscaban programas de demostración en los cuales se ejemplificara alguna forma de modificar el password de acceso a la interfaz web basadas en RabbitWeb.

Objetivo

- Demostrar una forma para permitirle al usuario la modificación de su password de acceso a la interfaz web, mediante esa misma interfaz.

El hardware

En este caso, no hay hardware adicional que construir, simplemente se requiere el módulo Rabbit que utilicemos (obviamente debe tener interfaz de red, ya sea Ethernet o WiFi) y la placa de prototipos o lo que fuera que utilicemos para, por lo menos, alimentar el módulo.

El software

El software está constituido en un proyecto de Dynamic C 10.50, que puede compilarse perfectamente en otras versiones de Dynamic C 10.xx.
Su estructura es la siguiente:

```
myProjects___CoAN-016
             |___CoAN-016.c          ---> Archivo fuente principal
             |___CoAN-016.dcp        ---> Archivo de proyecto
             |___lib.dir             ---> Archivo recursos
             |___myLibs
             |        |___iweb.lib   ---> Módulo de recursos web
             |        |___config.lib ---> Módulo de configuración y persistencia
             |        |___netcfg.lib ---> Módulo de configuración de red
             |
             |___iWeb
                      |___index.zhtml ---> Página principal
                      |___setup.zhtml ---> Página de configuración
                      |___style.css   ---> Hoja de estilos
```

La estructura precedente es típica de nuestros proyectos, por lo tanto solo comentaremos los aspectos relevantes para esta nota, a saber:
En el archivo ***iWeb.lib*** se concentran todos los elementos relacionados con RabbitWeb: importación de páginas, definición de tipos MIME, protección de páginas, creación de usuarios, exposición de variables, manejo de variables desde y hacia la interfaz de usuario web, etc. Aquí se definen los tres tipos de usuario fijos de la aplicación: *admin, operador, invitado*.
En el archivo ***config.lib*** tenemos por un lado, a las configuraciones relacionadas con la conectividad de red, y por otro, todas las rutinas relativas al manejo de la persistencia de datos configurables vía web en el "user block".
El módulo ***netcfg.lib***, concentra definiciones condicionales según el tipo de módulo para establecer la conexión de red y contiene la rutina de inicialización.
Finalmente, dentro del directorio ***iWeb***, tenemos a la interfaz web la cual se divide en un objeto principal (***index.zhtml***) que puede ser accedido por los tres usuarios definidos, y una segunda página (***setup.zhtml***) que solo puede ser accedida en su totalidad por el usuario *admin*, mientras que el usuario *operador* podrá accederla parcialmente y el usuario *invitado* directamente no tendrá acceso.

Ingresando entonces a la página ***setup.zhtml*** con el usuario *admin* se tiene el privilegio de poder cambiar las contraseñas de acceso a las citadas páginas web. Tanto las del propio *admin* como las del *operador (* el *invitado* accede sin contraseña).

El mecanismo de actualización

Se definieron tres grupos para el acceso web:

```
#web_groups ADMIN_GROUP
#web_groups OPER_GROUP
#web_groups ALL_GROUP
```

Y en el inicio del programa se crean los usuarios y se asignan a esos grupos:

```
void iweb_init()
{
   int uid;

   if((uid = sauth_adduser("admin", PASSWORD(0), SERVER_HTTP ))>=0)
             sauth_setusermask(uid, ADMIN_GROUP, NULL);
[...]
```

Nótese que el password se extrae según la macro `PASSWORD(0)`, que está definida en el módulo ***config.lib*** de la siguiente manera:

```
#define PASSWORD(index)            config.password[index]
```

Es decir, que refiere a una estructura de configuración que es la siguiente:

```
struct config_st{
   char version[NAME_MAXLEN+1];
   char name[NAME_MAXLEN+1]; //brief system description
   char password[PASS_MAXNUM][PASS_MAXLEN+1];
};
static struct config_st config;
```

Dicha estructura es escrita con la siguiente función de ***config.lib*** para modificar el password, cuando así se lo solicita desde la web según veremos más adelante:

```
void cfgSavePassword(char * newpass, unsigned char index)
{
     strcpy(PASSWORD(index), newpass);
     writeUserBlock( PASSOFFSET(index), PASSWORD(index), PASSSIZE(index));
}
```

Para recuperar los datos de configuración la siguiente función es invocada al inicio del programa:

```
void cfgReadAll()
{
  readUserBlock(&config, CONFIGOFFSET, CONFIGSIZE);
}
```

Volviendo al módulo ***iWeb.lib*** en la tabla de recursos se puede ver la relación entre la página ***setup.zhtml*** y los grupos que tienen acceso a ella, (anteriormente habíamos vinculado un usuario a cada grupo):

```
SSPEC_RESOURCETABLE_START
[...]
       SSPEC_RESOURCE_P_XMEMFILE("/setup.zhtml",setup_zhtm,CURRENT_VERSION,ADMIN_GROUP|OPER_GROUP,0,
                                 SERVER_HTTP,SERVER_AUTH_BASIC | SERVER_AUTH_DIGEST)
[...]
```

Así nuestro servidor RabbitWeb sabe que la autorización es requerida cuando se solicita aquella página, y luego sabe que un usuario puede ingresar si es que pertenece a alguno de esos grupos.

A continuación definimos mediante una directiva de compilador (`#web_update`) cual será la función encargada de actuar cuando se intente actualizar la variable que nos interesa (`strPass`) :

```
#web_update strPass                    params_change
```

`params_change` sería el puntero a la función del mismo nombre:

```
void params_change(void)
{
   switch (admincmd)
   {
      case 'o':
         cfgSavePassword(strPass, 1);
         sauth_setpassword( 1, strPass );
         break;
      case 'a':
         cfgSavePassword(strPass, 0);
         sauth_setpassword( 0, strPass );
         break;
      default:
                              break;
   }
   webreply=0;
}
```

Vemos, que desde esta función grabamos el password nuevo con la función `cfgSavePassword()` del módulo ***config.lib*** y a continuación activamos del password con la función de librería `sauth_setpassword()`. Cabe destacar que esta función podría mejorarse, si se enviara el UserId a la página web, y desde esta se lo remitiera como parámetro junto al password en lugar de utilizar una la variable `admincmd` como código para switchear según corresponda el password a un usuario o a otro.

La página setup.zhtml

En algún momento comentábamos que el usuario ***operador*** solo tiene acceso parcial a la página ***setup.zhml*** ese acceso parcial lo logramos gracias a la definición y protección de una simple variable, hacemos esto en el módulo ***iWeb.lib***:

```
unsigned char admincmd;
#web admincmd auth=basic, digest groups=ADMIN_GROUP(rw),OPER_GROUP(ro)
```

Y luego en la página propiamente dicha, preguntamos sobre el tipo de acceso que tiene el usuario actual sobre dicha variable, entonces, el servidor RabbitWeb al parsear la página, verá si el usuario que la solicita tiene los accesos requeridos, y en caso afirmativo, “dibujará” el formulario protegido, y en caso negativo toda la sección encerrada dentro del `if` se enviará en blanco hacia el explorador.

```
<?z if (auth($admincmd, "rw")) { ?>
      [ FORMULARIO PROTEGIDO]
<?z } ?>
```

Para finalizar, dentro del formulario protegido habrá un elemento input como este para el password:

```
< input style="text-align: left" type="password" name="strPass" id="pass1"
size="4" maxlength= "4" value="****">
```

Nótese que el nombre de la variable coincide necesariamente con el declarado en ***iWeb.lib.***

Y para el envío de los datos al servidor disponemos de un botón como este:

```
<input class="boton" type="button" onclick="javascript:send('a');" >
```

Que invoca a una función javascript que tras validar los datos simplemente los envía mediante una operación `submit()` sobre el formulario en cuestión: `document.frmAdmin.submit();`