¿Backdoors en routers OLO

SW(C|U)-9100?

por Josué Rojas a.k.a Nox
http://www.noxsoft.net

Introducción

Este escrito detalla el análisis y proceso de ingeniería inversa en el *router* Seowon Intech SWC-9100 y SWU-9100 usados por OLO, donde se halló un usuario no documentado en el manual de usuario que viene adjunto al adquirir el *router*, ni en la página web, ni se hace referencia en algún lado de la administración web respecto a la única cuenta que menciona el manual: admin/admin, usuario y contraseña respectivamente.

Los *routers* afectados son los siguientes: OLO fijo con tres antenas tiene las siguientes características:

- **MODELO:** Wimax SWC - 9100.
- **PROCESADOR:** ARM926EJ-S rev 5 (v5l) - little endian.
- **VERSIÓN DE LINUX:** 2.6.26.8-rt16.
- **VERSIÓN DEL GCC USADO:** 3.4.4.
- **XN NO SOPORTADO.**
- **SERVIDOR HTTP:** micro_httpd.

OLO fijo

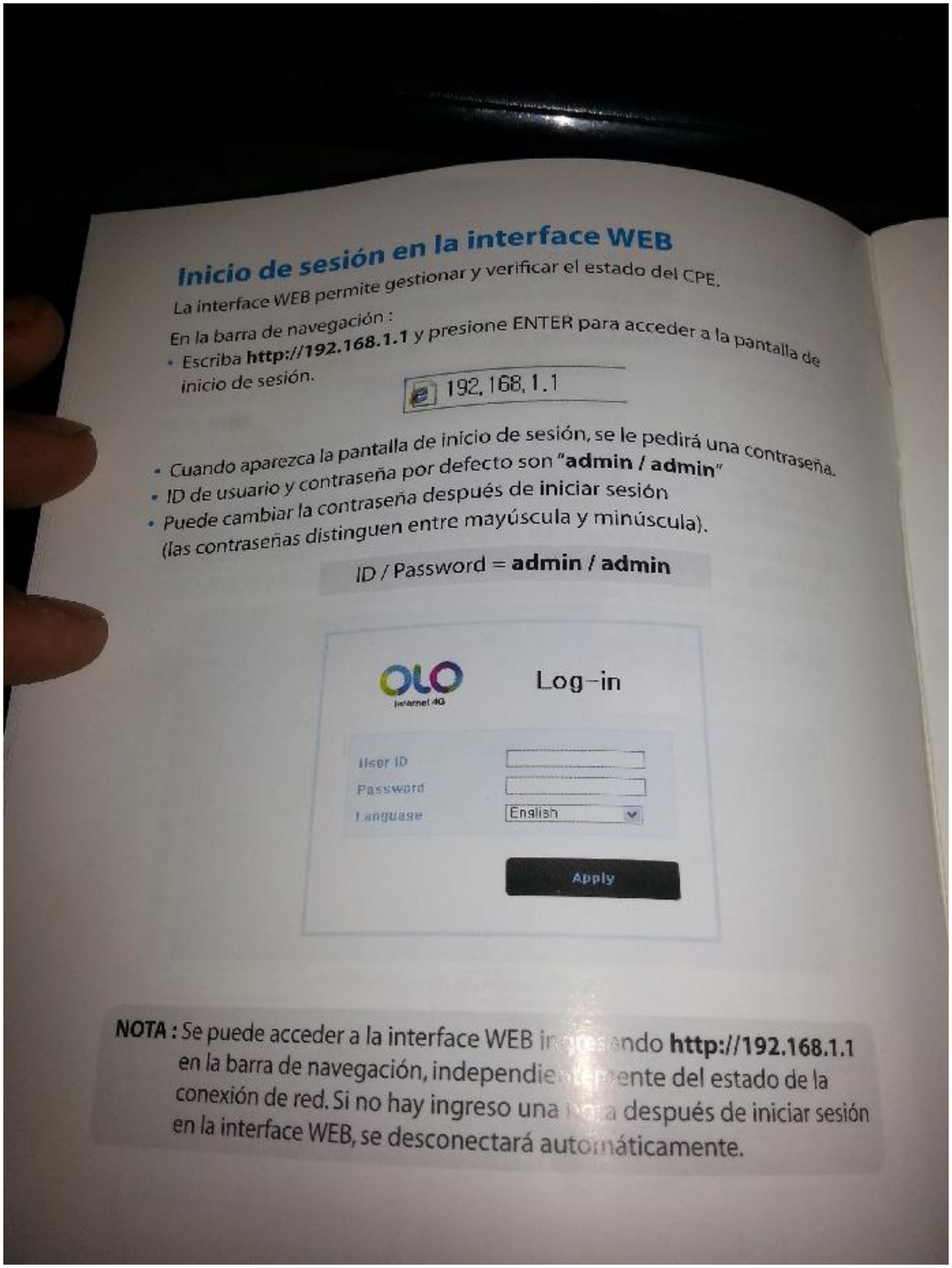
Nota: Toda la investigación fue basada en este *router* y, posteriormente probada los mismos fallos de seguridad encontrados, en el *router* móvil de OLO.

El *router* movil:

- **MODELO:** Wimax SWU - 9100.

OLO Movil

Al revisar el manual de usuario de OLO, que viene adjunto al adquirir el *router*, la página 10 se titula “Inicio de sesión en la interface WEB”. Es la única página donde hace referencia a las credenciales para la interfaz web, admin/admin, usuario y contraseña respectivamente.



Credenciales por defecto

Busqué más información en la página web del ISP, así como en la misma interfaz web y no encontré nada al respecto de otro usuario.

Analizando el binario "login.cgi"

El siguiente *basicblock* muestra de dónde lee las credenciales para comparar con los datos que se ingresa en el inicio de sesión de la interfaz web.

```
.text:0000E5E8
.text:0000E5E8 loc_E5E8                ; "/etc/webpasswd"
.text:0000E5E8 LDR     R0, =aEtcWebpasswd
.text:0000E5EC BL      get_contents
.text:0000E5F0 MOV     R3, R0
.text:0000E5F4 STR     R3, [R11,#cont_webpasswd]
.text:0000E5F8 LDR     R0, [R11,#cont_webpasswd]
.text:0000E5FC LDR     R1, =aAdmin_id  ; "ADMIN_ID="
.text:0000E600 BL      get_value_from_data
.text:0000E604 MOV     R3, R0
.text:0000E608 STR     R3, [R11,#ADMIN_ID]
.text:0000E60C LDR     R0, [R11,#cont_webpasswd]
.text:0000E610 LDR     R1, =aEncryption ; "ENCRYPTION="
.text:0000E614 BL      get_value_from_data
.text:0000E618 MOV     R3, R0
.text:0000E61C STR     R3, [R11,#ENCRIPTION]
.text:0000E620 LDR     R0, [R11,#cont_webpasswd]
.text:0000E624 LDR     R1, =aSystem_id ; "SYSTEM_ID="
.text:0000E628 BL      get_value_from_data
.text:0000E62C MOV     R3, R0
.text:0000E630 STR     R3, [R11,#SYSTEM_ID]
.text:0000E634 LDR     R0, [R11,#login_pw]
.text:0000E638 BL      generate_hash_pw
.text:0000E63C MOV     R3, R0
.text:0000E640 STR     R3, [R11,#pw_hash]
.text:0000E644 LDR     R0, [R11,#ADMIN_ID] ; s1
.text:0000E648 LDR     R1, [R11,#login_id] ; s2
.text:0000E64C BL      strcmp
.text:0000E650 MOV     R3, R0
.text:0000E654 CMP     R3, #0
.text:0000E658 BNE     other_auth
```

Se puede observar que en el ruta "/etc/webpasswd" existe un archivo, que cuyo contenido es comparado con los datos que se ingresan el iniciar la sesión.

"/etc/webpasswd"

```
ENCRYPTION=hi
ADMIN_ID=admin
ADMIN_PW=a720331bacf0ea52
SYSTEM_ID=system
SYSTEM_PW=ac5c044c25fc6fce
```

La estructura encontrada me hace recordar a los ficheros INI, que se usa para leer datos que el usuario pueda configurar. Donde se especifica como sigue:

```
[section]
key=string
```

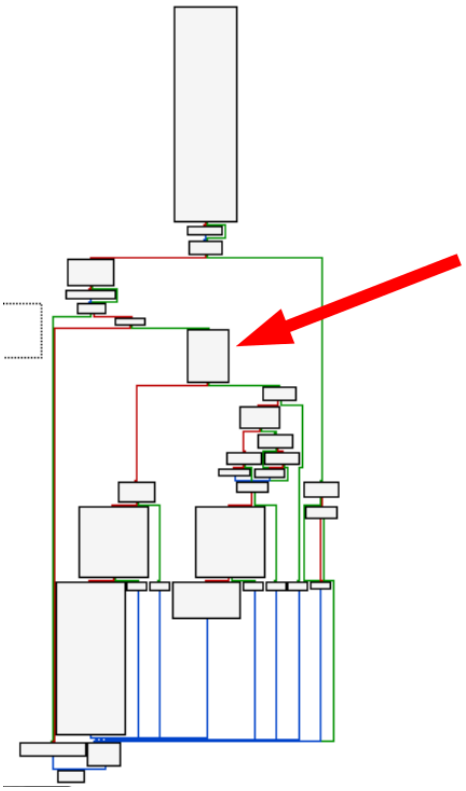
En este caso no existe la etiqueta sección que separa las “key” de otras, pero el formato es muy similar.

Aunque sea ensamblador, eh modificado las etiquetas de IDA para que sea muy entendible –y espero no haber fracasado–. Según el formato mostrado iremos desglosando el análisis.

1. Lee el contenido del archivo “webpasswd”.
2. Obtiene la `string` de la `key` “`ADMIN_ID`”.
3. Obtiene la `string` de la `key` “`ENCRYPTION`”.
4. Obtiene la `string` de la `key` “`SYSTEM_ID`”.
5. Se genera un hash de la contraseña ingresada.
6. Se compara la `string` de la `key` “`ADMIN_ID`” con el usuario ingresado en el inicio de sesión de la interfaz web.

A estas alturas ya te habrás dado cuenta de que existen dos usuarios, uno documentado y otro no. De nombre muy sugestivo, “system”, es el usuario que no se menciona en el manual de OLO.

El *basicblock* analizado se sitúa en la posición señalada del *callgraph* del binario “login.cgi”.



Callgraph del binario login.cgi

El *callgraph* es bastante claro en este punto, solo permite dos usuarios, no más, y este es un límite puesta en la programación del binario. Podemos ver que al terminar dicho *basiblock* hay un salto condicional, donde el flujo de ejecución toma dos caminos, saltar a la etiqueta “other_auth” o seguir de manera lineal.

Si se compara la `string` de la `key` “`ADMIN_ID`” con el usuario ingresado en el inicio de sesión de la interfaz web es igual, sigue de manera lineal, eso quiere decir que en la caja de texto, se ha ingresado el usuario “admin” y comienza la autenticación. Sin embargo sino es igual salta a la etiqueta “other_auth” y comienza la autenticación con el usuario “system”.

Autenticación con el usuario “system”.

Al comenzar la autenticación con el usuario “system” se ejecuta un procedimiento un tanto extraño, verifica si existe la key “`SYSTEM_PW2`” que debería tener el mismo hash que la key “`SYSTEM_PW`”, si esta no existe, la ignora por completo y solo compara el hash de la contraseña ingresada, con el hash obtenido de la key “`SYSTEM_PW`”. Como dije, “un tanto extraño”. Como esto no tiene sentido, seguimos.

Una vez que las dos cadenas han coincidido se levantan los servicios, configuraciones correspondientes y voilá, ya estás autenticado.

Un análisis superficial del algoritmo que genera el hash, nos da como conclusión de que no es *reverseable*. La función que he nombrado

“generate_hash_pw” en la dirección 0x0000DDF0, comienza generando un hash md5 sin *salt*, de la contraseña ingresada. Usar md5 ya nos indica que no se puede *reversear* para obtener la cadena original. Sabemos que el hash md5 tiene 32 caracteres, pero los observadores se habrán dado cuenta que en el archivo “webpasswd” solo hay 16, esto se debe a que se va convirtiendo a hexadecimal cada dos caracteres, pero al final solo se toma en cuenta una. Otro aspecto es que este algoritmo usa un *switch* con una lógica bastante inusual.

```
.text:0000DCC0
.text:0000DCC0 ; Attributes: bp-based frame
.text:0000DCC0
.text:0000DCC0 ; uint __cdecl switc_byte(uint caracter)
.text:0000DCC0 switc_byte
.text:0000DCC0
.text:0000DCC0 buf_char= -0x14
.text:0000DCC0 caracter= -0xD
.text:0000DCC0
.text:0000DCC0 MOV     R12, SP
.text:0000DCC4 STMFD   SP!, {R11,R12,LR,PC}
.text:0000DCC8 SUB     R11, R12, #4
.text:0000DCCC SUB     SP, SP, #8
.text:0000DCD0 MOV     R3, R0
.text:0000DCD4 STRB    R3, [R11,#caracter]
.text:0000DCD8 MOV     R3, #0
.text:0000DCDC STR     R3, [R11,#buf_char]
.text:0000DCE0 LDRB    R3, [R11,#caracter]
.text:0000DCE4 SUB     R3, R3, #0x41
.text:0000DCE8 CMP     R3, #0x25       ; switch 38 cases
.text:0000DCEC LDRLS   PC, [PC,R3,LSL#2] ; switch jump
```

El *byte* que se pasa como parámetro se le resta el valor 0x41, y luego es comparado con el valor 0x25, si es menor o igual sin considerar el signo, salta usando un *array* de direcciones que subsiguen al *basicblock*. Si no se cumple esa condición, hace un salto por defecto.

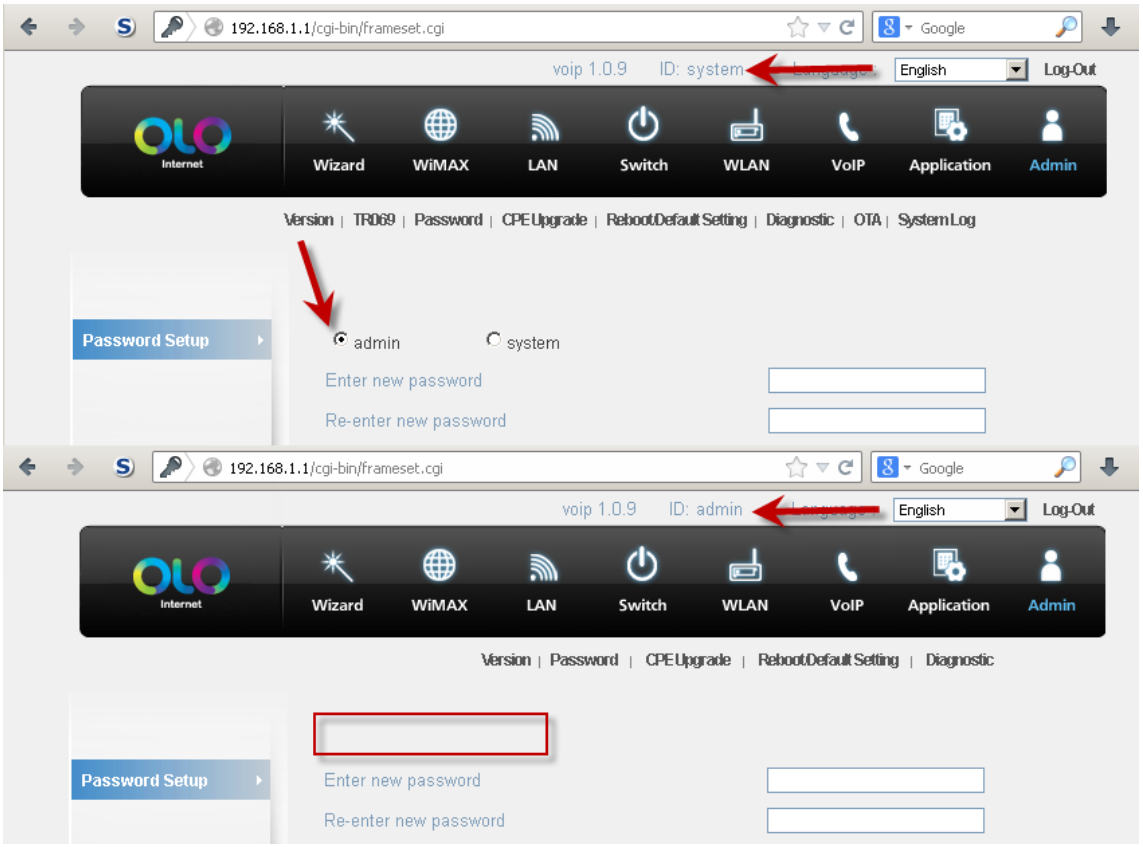
```
.text:0000DCF0                 B       loc_DDD4        ; jumptable 0000DCEC default case
```

Esta operación se realiza dos veces, luego con los dos resultados se hace una operación lógica OR, y si no me olvido de algo, ese será un caracter que será guardado en el archivo “webpasswd”, luego continua de la misma manera con el resto de caracteres para generar su propio hash.

El análisis es bastante superficial y es digno de otro escrito aparte dedicado solamente al análisis del algoritmo que genera el hash, y como el objetivo de este escrito no es analizarlo a profundidad seguimos con la deducción de la contraseña del usuario “system”.

Con todo lo mostrado podemos ver que es complicado o imposible hasta donde se sabe, poder *reversear* todo el algoritmo y generar la clave original, es más hasta estoy casi seguro que tampoco se podría obtener ni siquiera el hash md5 con que se realiza las operaciones, sin embargo de lo que estoy seguro es que se puede reproducir dicho algoritmo para poder generar nuestros hashes, y usando un fallo de seguridad como una inyección de comandos podemos escribir en el archivo “webpasswd”, poniendo el hash que hemos generado.

Aquí podemos enfrentarlos con la desdicha de no saber la contraseña del usuario system, pero volviendo a mirar el formato que se muestra en el manual "admin/admin" y un ingenio más grande que Leonardo Da Vinci, podemos probar las credenciales "system/system".



Comparación del usuario "system y "admin".

Y se autenticó, dimos en el clavo :P. El usuario "system" supone más privilegios que el usuario "admin", ya que desde system se puede cambiar la contraseña al usuario "admin", pero no en viceversa.

Por último les dejo los hashes de las cadenas "admin" y "system" para que puedan escribir como mencioné antes, en el archivo "webpasswd".

```
admin=62aa652730d7ba64
system=ac5c044c25fc6fce
```

- Demo del modelo SWC-9100:
  http://www.youtube.com/watch?v=PsKx6zOemHc
- Demo del modelo SWU-9100:
  http://www.youtube.com/watch?v=EQJVq88ZXBc

¿Ustedes qué piensan? ¿Backdoor o no? Hay varios puntos que analizar, no está en el manual de usuario, no existe ninguna información al respecto de ese usuario, solo puedes saber de que existe el usuario "system", *reverseando* el binario "login.cgi" o "pw.cgi", puedes acceder a la interfaz con ese usuario,

tiene más privilegios que el usuario que está documentado, “admin”, y también puedes cambiar la contraseña del usuario “system”, sin embargo el no comunicarte de que existe un usuario de tal magnitud puede llevar a que tu información caiga en malas manos, y que no se pueda evitar esto.

Sin duda, este usuario está pensando desde que fue programado el binario, y no se considera una vulnerabilidad, pero sí, una mala práctica. Si es *backdoor* o no, usted juzgue.

Aquí deseo citar las palabras de mi amigo César: “Hackea tu *router*”, y es cierto, sino lo haces, jamás sabrás que podrá pasar con tu información que pasa a través del *router*, mientras que tu AV estará de vagaciones ;).

Saludos,
Nox.