Explotación de dispositivos embebidos y más

Josué Rojas

Contenido

1. Resumen

Embedded devices often have a too little or none security, which could be an attack vector to malware writers. This research aims to demonstrate statistics in Perú about vulnerable routers exposed to internet and a cross-compilated malware analysis to ARM and MIPS affecting embedded devices being detected by VirusTotal at the current date.

2. Introducción

El uso de dispositivos electrónicos ha simplificado tareas cotidianas y/o repetitivas, con lo cual la sociedad los ha adoptado de manera masiva. Las empresas en la continua vorágine tecnológica, decidieron dotarlos de un microprocesador y un sistema operativo, creando con ello lo que conocemos como "dispositivos embebidos".

Siempre que hay un sistema operativo, existe el software malicioso que le afecta como si fuera inherente del mismo aunque no lo sea. Se puede observar un gran ejemplo en los sistemas operativos de escritorio, obteniendo grandes cantidades de muestras únicas de malware diarias. De esto no se escapan los dispositivos embebidos que tienen un sistema operativo. Debido a la ínfima seguridad implementa – ya que muchos casos desde la concepción no fue pensando desde la perspectiva de la seguridad –, la fácil explotación de vulnerabilidades, al descuido por parte de los usuarios dejando las credenciales por defecto, servicios instalados por el fabricante que permiten acceso al dispositivo con configuraciones por defecto, a la mala práctica de las empresas dejando un backdoor en estos dispositivos, o cualquiera que fuere, con el tiempo los malware writers se enfocaron en estos nuevos vectores de ataque para obtener el mayor partido posible. Ya se ha hecho público por empresas de seguridad y antivirus los casos descritos (1). Y en el futuro seguirán publicándose debido a todo lo expuesto si no se realiza un cambio. Este es el objetivo de este escrito, la concientización; demostrar que está sucediendo hoy en día, que sucederá en el futuro y como nosotros podemos intervenir el cambio.

En esta investigación se mencionará diversos routers vulnerables de arquitecturas MIPS y ARM. En un escaneo a todo el rango IPv4 se encontró expuestos a internet un aproximado de 15 mil routers vulnerables. Y finalizará en el análisis de un malware hallado, con las variantes en ARM y MIPS al cual he denominado "elan2", por el nombre de la primera muestra obtenida.

3. Investigaciones previas

He realizado investigaciones previas publicadas bajo el seudónimo de Nox y he podido colaborar en otra (2). Las nombradas a continuación han sido usadas en el desarrollo de este escrito. La ponencia "Explotación sobre ARM [1]", "Hacking de routers para profesionales de seguridad[2]", "Otra charla sobre explotación de routers[3]" por César Neira donde pude colaborar, "Hackeando routers Olo" y "¿Backdoors en routers Olo?

[1] Expuesto en LimaHack 2013.
[2] Expuesto en PeruHack 2014.
[3] Expuesto en LimaHack 2013.

4. Vulnerabilidades en routers

Este apartado tiene como finalidad resumir los trabajos mencionados en el apartado "Investigaciones previas". Demostrando como un número considerable de routers en Perú son vulnerables.

Se pueden observar claros ejemplos de reversing de dispositivos embebidos en la página de devttys0 (10). Otra página que almacena y expone vulnerabilidades de routers es routerpwn (11). Sin embargo, se hizo un análisis sin tomar en cuenta estas páginas.

Vulnerabilidades consideradas:

- Inyección de comandos.
- Elevación de privilegios
- Evasión de autenticación.
- Inclusión local de ficheros (LFI).
- Puertas traceras (backdoors).
- Credenciales por defecto.
- Otros.

Vulnerabilidades no consideradas:

- Cross Site Scripting (XSS).
- Cross Site Request Forgery (CSRF).
- Fijación de sesiones.
- Otros ataques que requieran interacción con el usuario.

Los routers analizados son brindados por los ISP en Perú y las vulnerabilidades encontradas son las siguientes:

- ZTE ZXHN H108N.
    - Fuga de credenciales en el código fuente.
    - Elevación de privilegios.
    - Inclusión Local de Ficheros (LFI).
    - Credenciales por defecto en conexión SSH.
    - Login bypass.
- ZTE ZXV10 W300 v1.0 y v2.1.
    - Elevación de privilegios.
    - Backdoor.
- Asus AR7WRD.
    - Inclusión Local de Ficheros (LFI).
    - Backdoor.
    - Fuga de credenciales del administrador.
- Nucom R5000UNv2.
    - Credenciales por defecto en conexión telnet generados a partir de la MAC.
    - Elevación de privilegios.
    - Inyección de comandos a través del usuario support y elevación a root.
    - Backdoor.
- Seowon Intech WiMAX SWC – 9100.

    - Inyección de comandos sin autenticación como root.
    - Reinicio del router sin autenticación.
    - Reinicio del router a estado de fábrica sin autenticación.
    - Backdoor.
- Seowon Intech WiMAX SWU – 9100.
    - Reinicio del router sin autenticación.
    - Reinicio del router a estado de fábrica sin autenticación.
    - Backdoor.

Todos los detalles técnicos se encuentran en la google drive (2) y en mi web, noxsoft.net.

5. Escaneo de routers

Debido a las vulnerabilidades expuestas en el apartado "Investigaciones previas". Conocer un número aproximado de routers, que son vulnerables en Perú es un dato que podría ayudarnos a concientizarnos sobre la seguridad de estos dispositivos, así como el reporte de estos ayudó a los fabricantes a conocer sus fallos. Nosotros podemos poner nuestros dos centavos, los fabricantes el resto, y así disminuir las vulnerabilidades que ponen en peligro a nosotros, los usuarios finales.

El alcance debe de darse al todo el rango IPv4 de Perú y según LACNIC (3), Perú tiene 2816000 IPs asignadas.

Problemas:

- Escanear 2816000 puede tomar mucho tiempo.
- Los modem/routers domésticos suelen tener IP dinámica.
- Se debe realizar el escaneo en el menor tiempo posible.
- Falsos positivos.
- Crear una firma (fingerprint) para la detección.

Para la identificación de routers podríamos usar diferentes técnicas de fingerprinting, sin embargo, usaremos dos en conjunto que dieron resultados[4]. El primero es usando el campo de la cabecera TCP, llamado "Window Size".

[4] Fyodor el creador de nmap en su web (4), explica diversas técnicas de *fingerprinting*.

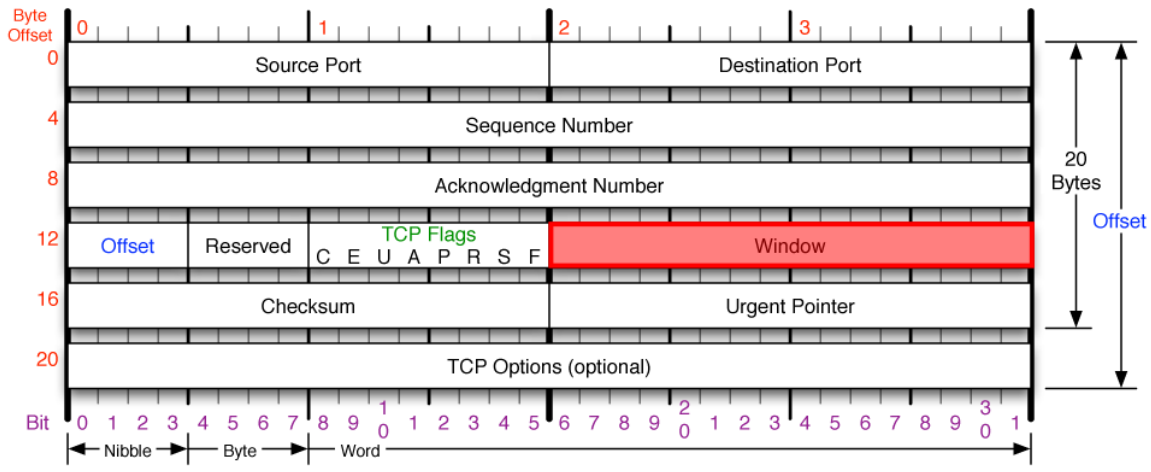

Ilustración 1 - TCP Window Size

El campo Window Size, se usa para indicar la cantidad de datos que un equipo está dispuesto a aceptar en el siguiente mensaje. Especifica la cantidad de datos que puede almacenar en el buffer reservado para la conexión durante el three-way handshake del TCP.

Ejemplo, si Windows Size indica que solo 100 bytes es la cantidad disponible que puede almacenar, y yo le envío 40 bytes, Windows Size indicará que en el siguiente mensaje aceptará 60 bytes.

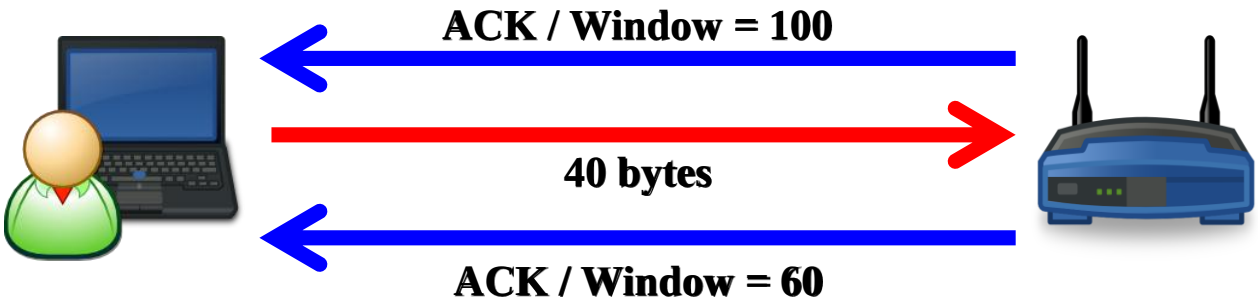

Ilustración 2 - Procedimiento de TCP Window Size

El tamaño inicial del buffer depende de cada sistema operativo, por tanto conocer el Window Size inicial, podría dar una idea del tipo de sistema operativo. Cabe resaltar que en algunos routers varía el Windows Size inicial según el puerto.

Algunos ejemplos son:

Equipo	Puertos	Window Size
ZTE W300 v2.1	80, 23	8192
ZTE W300 v2.1	7547	2048
ZTE W300 v1.0	21, 23, 80, 9999	5808
ZTE ZXHN H108N	22, 80, 443, 7547	5808

Tabla 1 - Ejemplos de Windows Size

El segundo es, identificando el banner de servicio. Diversos dispositivos nos dicen quienes son sin necesidad de análisis complejos.

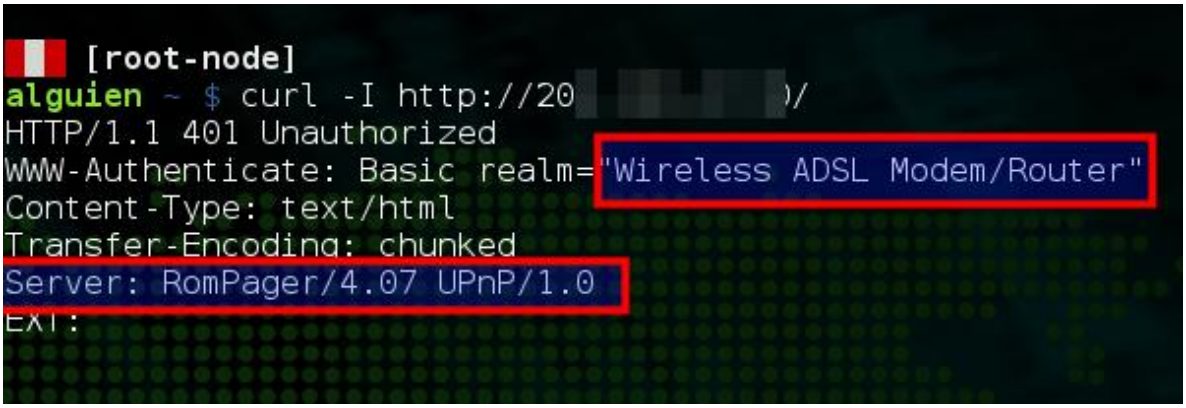

Ilustración 3 - Banner de servicio

Metodología empleada:

- Escaneo inicial de reconocimiento (todos los puertos/pocos equipos).
- Identificación del valor TCP Window Size para cada puerto.
- Identificación del banner de servicio para cada puerto.
- Escaneo SYN masivo con zmap (5), recolectando el Window Size devuelto.
- Reducción de IPs a escanear por Windows Size/Puerto.
- Escaneo de verificación por banner de servicio con nmap (6).

Los resultados se expresan en el siguiente cuadro, todos ellos expuestos a internet y vulnerables como se puede observar en el apartado “Vulnerabilidades en routers”.

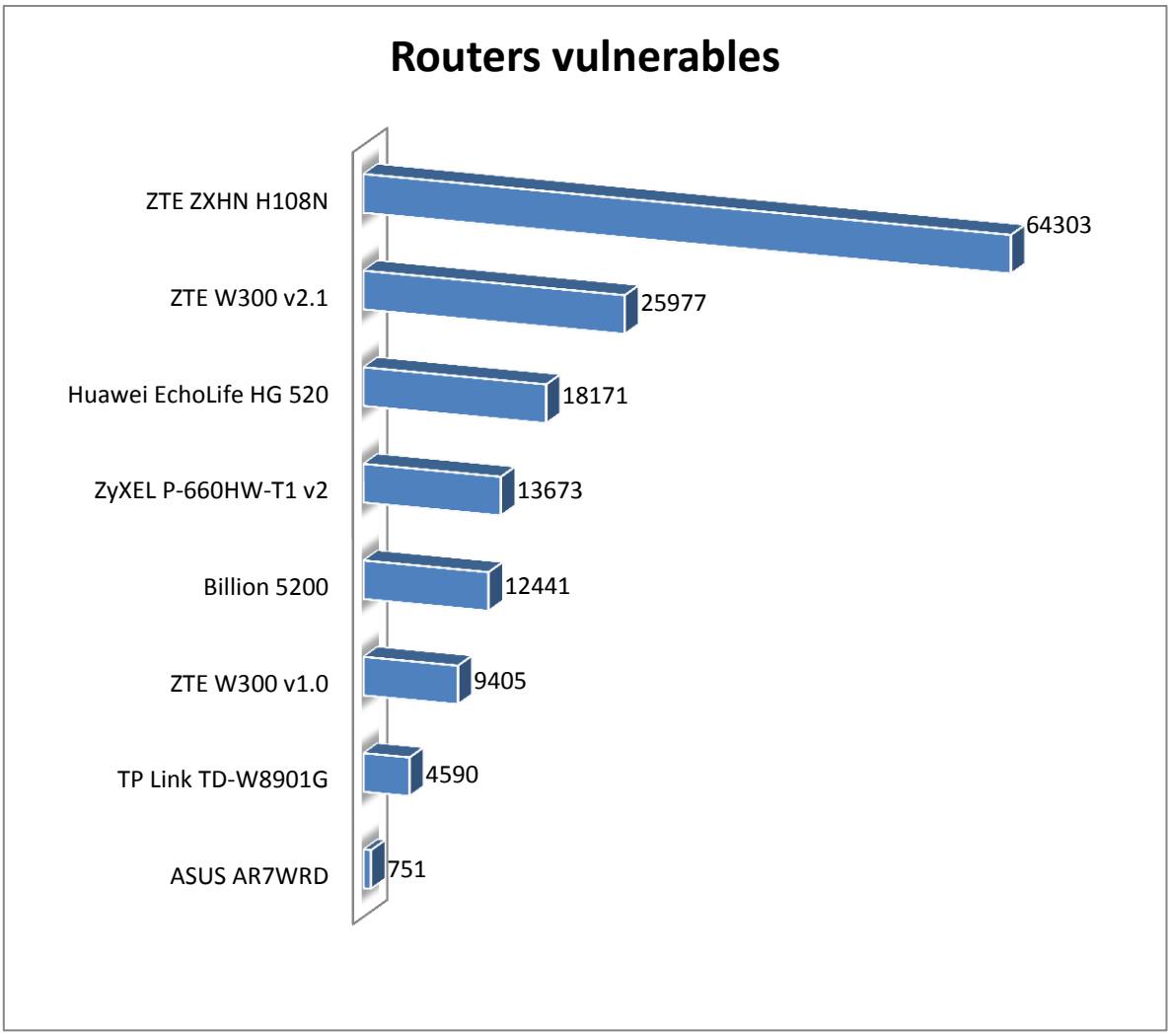

Ilustración 4 - Gráfico de routers vulnerables

6. Análisis del malware elan2

6.1 Identificación de las muestras

Antes de realizar ingeniería inversa a un malware, debemos saber el lenguaje en que ha sido programado, compilador, arquitectura, modo de compilación, packer usado, endian, etc.

```
$ file elan2-mips
elan2-mips: ELF 32-bit MSB executable, MIPS, MIPS32 version 1 (SYSV),
statically linked, stripped
$ file elan2-arm
elan2-arm: ELF 32-bit LSB executable, ARM, version 1, statically linked,
stripped
```

Dos binarios para diferente arquitectura. El primero, "elan2-mips" para arquitectura MIPS Big Endian, y el segundo "elan2-arm", para ARM Little Endian. Los dos han sido compilados estáticamente.

Una búsqueda rápida del hash sha-1 de dichos binarios en VirusTotal, da una búsqueda infructuosa[5]. Algo raro ya que tengo el malware en mi poder hace un mes.

Al analizar las strings, de cada malware, el binario "elan2-arm" nos dice la posible versión de la uClibc que se ha utilizado para compilar en estático. Esto es una información de suma importancia, y aunque recién haciendo este escrito, me puse a mirar los strings; nos sirve, ya que podremos crear signatures en IDA Pro. De esa manera evitamos estar analizando funciones propias de la libc y solo nos concentramos en el código programado por el malware writer.

En resumen, se ha obtenido la siguiente información de los binarios.

Nombre	elan2-mips.
MD5	df48ca35ca4a13a3121a24ca897c5eec
SHA-1	2903d9b47bfc2f3e1913ba647aa46d9ea62b0696
Arquitectura	MIPS Big Endian.
Compilador	GCC (compilación estática).
Lenguaje	C.
LibC	uclibc 0.9.33.
Packer	No.
Nombre	elan2-arm.
MD5	c89dab2ffcfccac9d37f837be5b2c4e1
SHA-1	bbb704ddb3d4613ebb02fc8eb6f61e759e75e0ac
Arquitectura	ARM Little Endian.
Compilador	GCC (compilación estática).
Lenguaje	C.
LibC	uclibc 0.9.30.
Packer	No.

Tabla 2 - Información de las muestras

6.2 Creando firmas para IDA

La creación de firmas o signatures[6] en IDA Pro, es esencial si queremos reducir el tiempo de análisis de las funciones. El binario "elan2-mips" tiene 583 funciones, mientras que "elan2-arm" tiene 567, analizar cada una función sería un trabajo de chinos. Y en este caso no lo necesitamos porque sabemos que el binario ha sido compilado en estático y la gran mayoría de las funciones son parte de la libc.

[5] VirustTotal permite buscar muestras especificando el hash (7).

[6] Firmas o Signatures, se crean para la identificación de funciones. Se crea una "firma" o conjunto de bytes, que será usado como signature, para recorrer todo un binario. Al encontrar coincidencias, la función será nombrada como indique la signature.

Para crear una signature para IDA Pro, necesitas compilar y obtener una librería estática. De esta manera, usando FLAIR, se podrá crear una sinature, reconocible por el FLIRT engine (8) integrado en IDA Pro.

Intenté crear signatures, para la variante en MIPS – porque fue la primera que tuve en mis manos –, sin embargo, cada vez que lo intentaba con varias versiones de la libc, siempre obtenía un error de relocación. Como no quería darme por vencido, programé un script en Bash, que hiciera el trabajo por mí, e identificara las librerías estáticas que no daban ese error – en Unix las librerías estáticas tienen la firma "AR" en el offset 0, y la extensión ".a" (9) –, para luego crear la signature. Al final solo obtuve 38 funciones reconocidas.

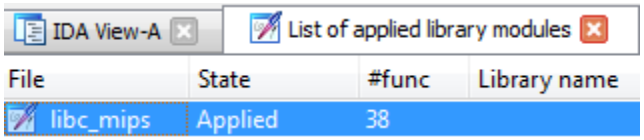

Ilustración 5 – Funciones de la libc reconocidas en elan2-mips

La uClibc, tiene código pre-compilado a disposición de cualquiera. Al yo haber desensamblado los binarios, y mirar el código por encima se puede deducir que el "malware writer", no tenía gran conocimiento, o tenía pocas ganas de programar "bien".

Específicamente, descargué la versión 0.9.30.1 del cross-compiler, ARM y MIPS.

- http://uclibc.org/downloads/binaries/0.9.30.1/cross-compiler-armv5l.tar.bz2.
- http://uclibc.org/downloads/binaries/0.9.30.1/cross-compiler-mips.tar.bz2.

Con la versión para MIPS tuve el mismo problema. Por el contrario, la versión para ARM no tuve casi ningún inconveniente.

```
$ ../../../flair/bin/linux/pelf libc.a libc.pat
libc.a: skipped 1, total 874
$ ../../../flair/bin/linux/sigmake libc.pat libc.sig
libc.sig: modules/leaves: 783/873, COLLISIONS: 5
See the documentation to learn how to resolve collisions.
$ ../../../flair/bin/linux/sigmake libc.pat libc.sig
$ ../../../flair/bin/linux/pelf libpthread.a libpthread.pat
libpthread.a: skipped 0, total 20
$ ../../../flair/bin/linux/sigmake libpthread.pat libpthread.sig
```

El binario "sigmake" dice que hay 5 colisiones al crear las signature de la libc, y debemos resolverlo. Es bastante sencillo la verdad, solo hay que leer la documentación. Una vez realizado, se ejecuta otra vez "sigmake" y este creará las signatures. En este caso he creado dos: "libc.sig" y "libpthread.sig". Finalmente cargamos las signatures.

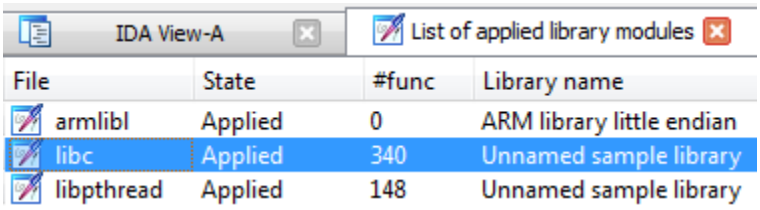

Ilustración 6 – Funciones de la libc reconocidas en elan2-arm

¡448 funciones reconocidas! Gran parte del trabajo ya está hecho. El único inconveniente es que no ha reconocido todas las funciones de la libc. Y las que sobran toca hacerlas a mano. Pero ya no son muchas y además es obvio en este malware, cuando una función le pertenece a la libc.

6.3 Análisis de código muerto

6.3.1 Reconociendo el Endian

Este tipo de malware para estar diseñados a diferentes arquitecturas, también debe tener en cuenta el formato en que se guarda los datos el procesador (12). Este malware tiene una función que reconoce si es big endian o little endian.

```
.text:000083AC                 MOV             R3, #1
.text:000083B0                 STR             R3, [R11,#var_10]
.text:000083B4                 SUB             R3, R11, #-var_10
.text:000083B8                 LDRB            R3, [R3]
.text:000083BC                 CMP             R3, #1
.text:000083C0                 BNE             loc_83D0
.text:000083C4                 MOV             R3, #1
.text:000083C8                 STR             R3, [R11,#var_14]
.text:000083CC                 B               loc_83D8
.text:000083D0 ; ---------------------------------------------------------------------------
.text:000083D0
.text:000083D0 loc_83D0                                ; CODE XREF: return_val_endian+24j
.text:000083D0                 MOV             R3, #0
.text:000083D4                 STR             R3, [R11,#var_14]
```

Tabla 3 – Identificación del endian

Un Endian es el formato en que se almacenan los datos. Little endian, almacena el byte menos significativo en la parte alta de una dirección de memoria, y el byte más significativo en la parte baja. Big endian, lo hace de manera viceversa.

Si guardamos el valor 1, del tamaño de un DWORD sería 00000001h. Teniendo en cuenta los endianness, al primer byte sería 01h si es little endian, y 00h si es big endian. Esta es la manera en como reconoce el malware el endian.

6.3.2 Creando un bind socket

Crea un socket de tipo bind en el puerto 10073, usando el protocolo TCP. Guarda el descriptor del socket en una variable global llamada "socket_bind" para finalmente crear un nuevo hilo de la función "wait_connection" quien inicializará su escucha.

```
    sockaddr_in.sin_family = AF_INET;
```

```
v0 = htons(0x2759);
sockaddr_in.sin_port = v0;
sockaddr_in.sin_addr = htonl(0);
if ( _GI_bind(socket_fd, (const struct sockaddr *)&sockaddr_in, sizeof(sockaddr_in)) >= 0 )
{
  if ( _GI_listen(socket_fd, 128) == -1 )
  {
    v2 = 1;
  }
  else
  {
    socket_bind = socket_fd;
    pthread_create(&thread, 0, (void **)wait_connection, 0);
    v2 = 0;
  }
}
```

Tabla 4 - Creación de un bind socket

Encontrar este puerto escuchando es usado por el malware como marca identificadora de infección.

6.3.3 Eliminando a la competencia

El pseudo-filesystem "/proc", entre otras cosas, se puede encontrar el PID[7] de los procesos en ejecución. El malware realiza esta acción para obtener la información que retorna al leer "proc/<PID>/cmdline" (13). Esta información devuelve la línea de comandos usada en la ejecución del proceso. Ejemplo:

```
$ qemu-mips -strace -g 4444 elan2-mips
$ cat /proc/2922/cmdline
qemu-mips-strace-g4444elan2-mips
```

Tabla 5 - Obteniendo línea de comandos

De la misma manera el malware busca cadenas como "/Challenge", "--scrypt", "stratum+tcp://" y "cmd.so" (14), en la línea de comandos especificados en la creación de algún proceso. Una búsqueda rápida sobre estas cadenas, podemos hallar que se encuentran en el uso de mining (minería) de crypto-monedas (15). Una vez hallada dichas cadenas, termina el proceso. Haciendo una llamada a la función "system" con los parámetros "kill <PID>".

```
int __fastcall read_cmdline_from_pids()
{
  void *v0; // r1@1
  char buff_cmdline; // [sp+5h] [bp-11Bh]@4
  int v3; // [sp+104h] [bp-1Ch]@1
  int DIR_proc; // [sp+108h] [bp-18h]@1
  struct dirent *v5; // [sp+10Ch] [bp-14h]@3
  int pid; // [sp+110h] [bp-10h]@3

  v3 = 0;
  DIR_proc = _GI_opendir("/proc");
  if ( DIR_proc )
  {
    while ( 1 )
    {
      v5 = _GI_readdir((struct dirent *)DIR_proc);
      if ( !v5 )
        break;
```

[7] ProcessID o Identificador de Proceso.

```
      szPid = atol(v5->d_name);
      if (szPid > 0 )
      {
        _GI_snprintf(&buff_cmdline, 255, "/proc/%s/cmdline", szPid);
        if ( find_substring(
               "/Challenge",
               &buff_cmdline,
               "--scrypt",
               "stratum+tcp://",
               "cmd.so") )
        {
          _GI_snprintf(&buff_cmdline, 255, "kill %s", szPid);
          system(&buff_cmdline);
          ++v3;
        }
      }
    }
    _GI_closedir(DIR_proc);
  }
  return v3;
}
```

6.3.4 Sobreviviendo al reinicio

Esta función se encarga de buscar archivos que permiten una ejecución. Los archivos "/etc/init.d/rcS" y "/etc/rc.d/rc", son scripts en bash que se ejecutan al inicio del sistema. Un método usado para que el malware pueda sobrevivir al reinicio del sistema. La cadena "/home/hik/start.sh" también se ejecuta al inicio del sistema, sin embargo esta es característica de dispositivos DVRs hikvsion (16). Finalmente tenemos a cron (17), administrador de procesos en segundo plano (demonio) permite ejecutar una acción a determinado tiempo. El malware busca los archivos de configuración "/etc/crontab" y "etc/cron.hourly/x"; para sobrevivir al reinicio ejecutándose en un hora específica. Al encontrar uno de estos archivos devuelve un ID que podrá ser interpretado por el centro de comando para conocer el archivo a editar.

```
int __fastcall find_file_boot()
{
  […]

  v2 = open("/etc/init.d/rcS", 66);
  if ( v2 == -1 )
  {
    v3 = open("/home/hik/start.sh", O_RDWR);
    if ( v3 == -1 )
    {
      v4 = open("/etc/crontab", O_RDWR);
      if ( v4 == -1 )
      {
        v5 = open("/etc/cron.hourly/x", 66);
        if ( v5 == -1 )
        {
          v6 = open("/etc/rc.d/rc", O_RDWR);
          if ( v6 == -1 )
```

- Si el archivo "/etc/init.d/rcS" existe, retorna el valor 1.
- Si el archivo "/home/hik/start.sh" existe, retorna el valor 2.
- Si el archivo "/etc/crontab" existe, retorna el valor 3.
- Si el archivo "etc/cron.hourly/x" existe, retorna el valor 4.
- Si el archivo "/etc/rc.d/rc" existe, retorna el valor 5.

- Si ningún archivo de los anteriores existe, retorna el valor 0.

Se debe tener en cuenta, de que el malware se está ejecutando en un usuario que tiene los privilegios para leer los archivos anteriores. Usualmente, los dispositivos embebidos corren con privilegios de root o system.

6.3.5 Infección

Para verificar la infección de un dispositivo, el malware se conecta hacia IP que generó de manera aleatoria para poder propagarse. Si este dispositivo tiene el puerto 10073 abierto, no lanza la infección, sin embargo, si no tiene el puerto abierto, lanza la infección. Como se puede observar en el flowgraph siguiente.

Ilustración 7 - Flow graph de la infección

La infección consiste en conectarse por Telnet y probar una serie de credenciales que se descargaba del centro de comando y control, para realizar fuerza bruta.

Source	Destination	Protocol	Length	Info
192.168.	190.232.	TCP	74	23→41553 [SYN,
190.232.	192.168.	TCP	66	41553→23 [ACK]
192.168.	190.232.	TELNET	78	Telnet Data ...
190.232.	192.168.	TCP	66	41553→23 [ACK]
192.168.	190.232.	TELNET	100	Telnet Data ...
190.232.	192.168.	TCP	66	41553→23 [ACK]
190.232.	192.168.	TELNET	71	Telnet Data ...
192.168.	190.232.	TCP	66	23→41553 [ACK]
192.168.	190.232.	TELNET	71	Telnet Data ...
190.232.	192.168.	TELNET	68	Telnet Data ...
190.232.	192.168.	TCP	66	41553→23 [ACK]
192.168.	190.232.	TELNET	78	Telnet Data ...
190.232.	192.168.	TCP	66	41553→23 [ACK]
190.232.	192.168.	TELNET	71	Telnet Data ...
192.168.	190.232.	TCP	66	23→41553 [ACK]
190.232.	192.168.	TELNET	68	Telnet Data ...

Ilustración 8 - Fuerza bruta por Telnet

Usando credenciales como admin/admin, admin/12345, etc. Una vez ha conseguido autenticarse intenta obtener una Shell para luego descargar el malware dentro del dispositivo.

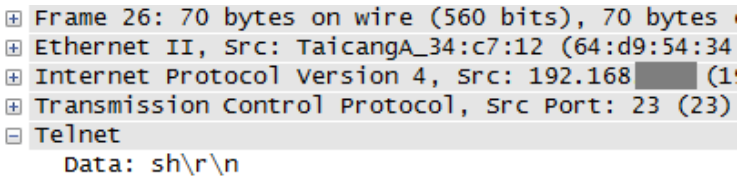

Ilustración 9 - Obtener shell

7. Impacto del malware en Perú

El malware analizado en el apartado "Análisis de malware elan2", contiene una función de infección como se ha mencionado en el apartado "Infección". Para ejecutar la infección verifica si el puerto 10073 está escuchando. Siguiendo estos mismos pasos, se realizó un escaneo en todo el rango IPv4 de Perú usando zmap(6) y se logró obtener aproximadamente 3000 IPs con dicho puerto escuchando. Luego se realizó una búsqueda usando el motor de google y no se encontró servicios asignados a este puerto (18). Aún se sigue trabajando para verificar si todos los dispositivos con dichas IPs han sido infectados.

8. Post-explotación

Aun en la actualidad, se siguen usando contraseñas por defecto como se ha mostrado. Al tener un acceso a un DVR hikvision (dispositivo que es considerado por el malware para infección), y tratar de conectarme por Telnet usando la credenciales admin/12345, me di con la sorpresa que se realizó una conexión exitosa.

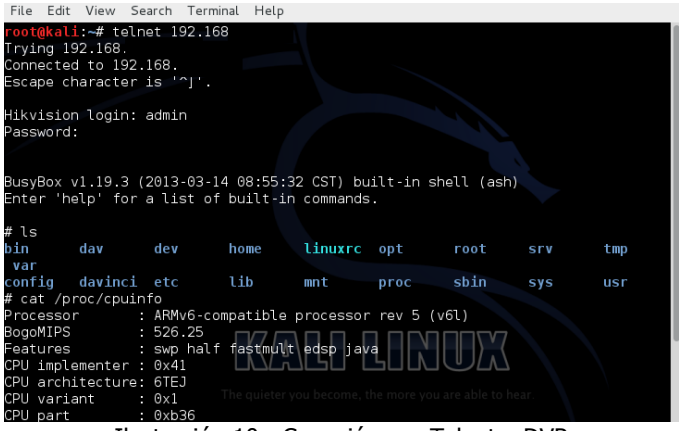

Ilustración 10 - Conexión por Telnet a DVR

La post-explotación tiene límites según la imaginación y conocimiento del malware writer. Pudiendo cambiar la DNS, de tal manera que al visitar una página (bancaria) el usuario puede visitar una página fraudulenta. Otro ejemplo sería crear un backdoor obteniendo una shell remota. La siguiente imagen muera una conexión de un router (192.168.1.1) a un computador con sietma operativo Ubuntu.

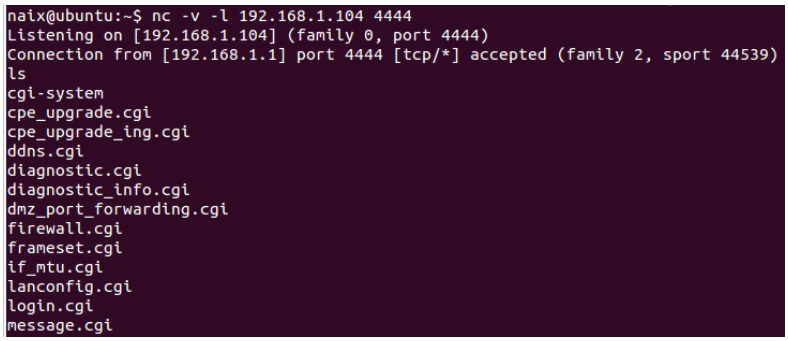

Ilustración 11 - Conexión de un router a un computador

9. Conclusiones

Como se ha podido comprobar en este escrito, la explotación de vulnerabilidades en dispositivos embebidos es sencilla. De tal manera que alguien sin muchos conocimientos puede causar grandes estragos explotando estos dispositivos.

Desde el inicio de este escrito, el fin fue demostrar que los dispositivos embebidos en su mayoría carecen de seguridad. Un trabajo en conjunto, los fabricantes con la ayuda en forma de concejos de parte de los usuarios para mejorar la seguridad de los dispositivos que consumimos.

10. Trabajo futuro

Continuar con el análisis del malware publicando un escrito técnico con análisis estático y dinámico avanzado. El análisis estático avanzado consiste en analizar el desensamblado en código muerto como se ha visto en el apartado "Análisis de malware elan2". El análisis dinámico avanzado, se extiendo a la ejecución del malware en un entorno controlado, usando la arquitectura específica del malware y monitorizando las acciones del malware para luego exponerlos en la publicación.

11. Agradecimiento

A César Neira, compañero de equipo y a todo los miembros de amn3s1a (grupo participante en CTFs), a Apo, todo CracksLatinoS, y a mi familia; quienes me soportan horas enfrente del computador viendo pantallas negras y código esotérico.