Uno de los trabajos más complicados que he tenido que realizar ha sido el «hackeo» de un módulo stick Bastone GPON SFP ONU, también conocido como GPON-ONU-34-20BI o SourcePhotonics SPS-34-24T-HP-TDFO, que fue un encargo de una empresa del sur de España (no puedo decir más). Estaban usando el sistema en producción y querían ver lo seguro que podía ser, específicamente en la parte de (una vez hackeado) la instalación de nuevos paquetes sobre el firmware que ya estaba corriendo.

Para realizar la tarea, emulé en mi laboratorio de pruebas un sistema con un procesador MIPS32 virtual del exacto mismo modelo con el que cuenta el stick y con el mismo sistema de archivos de OpenWrt que forma parte del firmware que el stick contiene. Para obtener la imagen de donde sacar el kernel y el sistema de archivos podemos utilizar dos métodos:

- Extraer la imagen del propio stick
- Descargar la imagen de alguien que ya haya hecho este trabajo

En este hack explicaré ambos métodos, pero dejaré la explicación de como hacer directamente desde el stick, para más adelante.

EXTRAER EL FIRMWARE DIRECTAMENTE DEL STICK

Para más adelante...

EXTRAER EL FIRMWARE DESDE LA IMAGEN DISPONIBLE EN hack-gpon.org

Si vamos a la web de hack-gpon.org, específicamente a la sección correspondiente al módulo en cuestión, podemos observar que existe disponible para su descarga el firmware con versión 6BA1896SPLQA42, correspondiente al 18 de septiembre de 2021. A la fecha de escritura de este hack los enlaces de descarga son, este para el U-Boot y este para la imagen que contiene el kernel y el sistema de archivos.

Si ejecutamos binwalk sobre la imagen mt2 (que es la del kernel y el sistema de archivos) obtendremos la siguiente salida:

```
DECIMAL HEXADECIMAL DESCRIPTION
--------------------------------
      0 0x0         uImage header, header size: 64 bytes, ..., OS: Linux, CPU: MIPS, image type: OS Kernel Image,
compression type: lzma, image name: "SFP_7.5.3"
     64 0x40        LZMA compressed data, properties: 0x5D, dictionary size: 8388608 bytes, uncompressed size:
3510436 bytes
1207910 0x126E66    Squashfs filesystem, little endian, version 4.0, compression:xz, size: 2205498 bytes, 911
inodes, blocksize: 262144 bytes, created: 2021-09-18 02:35:56
3473408 0x350000    JFFS2 filesystem, big endian
```

La primera imagen sólo tiene 64 bytes, porque va desde 0 a 64, en decimal. por lo que el kernel no puede estar ahí Estará seguramente en la siguiente imagen (LZMA comporessed), que tiene 1207846 bytes (1207910 – 64). Y, aunque veamos que tiene ese espacio, al final de la línea podemos ver su espacio descomprimido, 3510436 bytes, que ya nos encaja mejor como una imagen de kernel pequeña. Para extraerla, entonces, ejecutamos:

```
dd if=/home/usuariox/mt2.img of=/home/usuariox/kernel.lzma bs=1 skip=64 count=1207846
```

Para descomprimirla, ejecutamos:

```
unlzma /home/usuariox/kernel.lzma
```

El archivo comprimido se borrará y obtendremos el archivo del kernel descomprimido.

Si hacemos binwalk sobre ese archivo descomprimido obtendremos el siguiente resultado:

```
   1040    0x410 Flattened device tree, size: 8160 bytes, version: 17
2845464 0x2B6B18 DES SP1, big endian
2845720 0x2B6C18 DES SP2, big endian
2896172 0x2C312C Linux kernel version 3.10.4
2929746 0x2CB452 Unix path: /var/run/rpcbind.sock
2939106 0x2CD8E2 Copyright string: "Copyright 2013, Lantiq Deutschland GmbH"
```

```
3012984 0x2DF978 xz compressed data
3027464 0x2E3208 Unix path: /lib/firmware/updates/3.10.49
3064026 0x2EC0DA Neighborly text, "neighbor %.2x%.2x.%pM lost rename link %s to %s"
3198464 0x30CE00 CRC32 polynomial table, big endian
3509920 0x358EA0 ASCII cpio archive (SVR4 with no CRC),file name:"dev",file name length:"0x00000004",file size:
3510036 0x358F14 ASCII cpio archive (SVR4 with no CRC),file name:"dev/console",file name length:"0x0000000C",file
size:
3510160 0x358F90 ASCII cpio archive (SVR4 with no CRC),file name:"root",file name length:"0x00000005",file size:
3510276 0x359004 ASCII cpio archive (SVR4 with no CRC),file name:"TRAILER!!!",file name length:"0x0000000B",file
size:
```

...vemos que a partir del byte 2896172 empieza la imagen de kernel v3.10.4. También vemos que en el byte 2929746 comienza una nueva imagen. Entonces, para extraer la imagen del kernel sabemos que debemos empezar en 2896172 y debemos acabar en 2929746. Entonces 2929746 – 2896172 = 33574 (lo que serán los bytes a contar a partir de 2896172) simplemente ejecutamos:

```
dd if=/home/usuariox/kernel of=/home/usuariox/vmlinux bs=1 skip=2896172 count=33574
```

Ahora si, si ejecutamos:

```
strings /home/usuariox/vmlinux | grep inux
```

...obtenemos:

```
Linux version 3.10.49 (sean@Lantiq-DEV) (gcc version 4.8.3 (OpenWrt/Linaro GCC 4.8-2014.04 14.07_ltq) ) #1 Sat Sep
18 10:35:47 CST 2021
```

Listo, ya tenemos la imagen del kernel descomprimida!

Ahora, para extraer la imagen del sistema de archivos lo haremos de la siguiente manera:

Creamos un archivo de imagen vacío con dd del tamaño máximo de memoria que tiene el stick (64MB)

```
dd if=/dev/zero of=/home/usuariox/rootfs.img bs=1M count=64
```

A continuación, utilizamos el comando mkfs (sólo disponible como root o con sudo) para formatear el archivo de imagen como un sistema de archivos ext4:

```
mkfs.ext4 /home/usuariox/rootfs.img
```

Una vez formateado, montamos el archivo, ejecutando como root:

```
mkdir /mnt/rootfs
mount -o loop /home/usuariox/rootfs.img /mnt/rootfs
```

Copiamos el contenido del sistema de archivos en la imagen montada, ejecutando como root:

```
cp -r /home/usuariox/rootfs/* /mnt/rootfs
```

...y, finalmente desmontamos la imagen, ejecutando como root:

```
umount /mnt/rootfs
```

Listo, ya tenemos la imagen con el sistema de archivos!

NOTA: Binwalk ofrece una forma fácil de extracción recursiva de una imagen (y todas las imagenes que se encuentran dentro de ellas), pero no siempre funciona. En este caso, la orden sería:

```
binwalk -e mt2.img
```

...lo que nos habría descomprimido estos tres archivos:

```
126E66.squashfs
350000.jffs2
40.7z
```

...que a su vez habrían sido descomprimidos en estos otros:

```
40
jffs2-root
squashfs-root
squashfs-root-0
```

Pero, como vemos, el archivo 40, que ha sido nombrado así por el offset en el que se encontró, sería la imagen equivalente a /home/usuariox/kernel.lzma, pero binwalk no lo ha descomprimido a su vez en todas las imágenes que hay dentro de él, por lo que la extracción recursiva automática de binwalk no siempre funciona y deberemos hacerlo manualmente.