universidad de león

Escuela de Ingenierías

Industrial, Informática y Aeroespacial

MÁSTER EN INGENIERÍA INFORMÁTICA

Trabajo de Fin de Máster

Análisis de Tráfico de Red de dispositivos IoT

IoT Devices Network Traffic Analysis

Autor: Rubén Rodríguez Campelo
Tutor: Isaías García Rodríguez

(Febrero, 2022)

UNIVERSIDAD DE LEÓN
Escuela de Ingenierías Industrial, Informática y Aeroespacial

MÁSTER EN INGENIERÍA INFORMÁTICA
Trabajo de Fin de Máster

ALUMNO: Rubén Rodríguez Campelo

TUTOR: Isaías García Rodríguez

TÍTULO: Análisis de Tráfico de Red de dispositivos IoT

TITLE: IoT Devices Network Traffic Analysis

CONVOCATORIA: Febrero, 2022

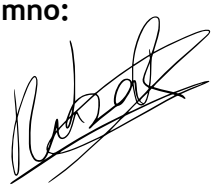
RESUMEN:

En la actualidad multitud de dispositivos se conectan entre sí para intercambiar información. La importancia de dicha interconexión y la gran variedad de servicios ofrecidos a través de la red de redes (Internet) ha alcanzado lo más profundo de los hogares de las personas. Hoy en día es común encontrar televisores con acceso a Internet o termostatos que se pueden controlar desde miles de kilómetros de distancia. A este fenómeno de conectar a Internet dispositivos de uso cotidiano que antes no disponían de dicho acceso a la red, se le conoce como Internet de las cosas. Al margen de este fenómeno y todas las ventajas que trae consigo, este tipo de dispositivos pueden presentar problemas de seguridad que los hace objetivo de ataques. Además, y dada la diversidad de fabricantes, la homogeneidad en cuanto a los métodos de conexión utilizados es inexistente. Este trabajo utiliza el tráfico generado por el comportamiento de estos dispositivos en las redes domésticas de los usuarios capturado mediante un entorno de pruebas desarrollado, y aplica diferentes técnicas de análisis como la inspección profunda de paquetes o el análisis basado en flujos con el fin de estudiar y profundizar en el uso de dichas técnicas, así como explorar formas avanzadas de enriquecimiento y visualización de los datos.

ABSTRACT:

Nowadays, many devices interconnect each other to exchange information. The importance of that interconnection and the wide range of available services on the Internet has reach people homes. Today, it is usual to find common appliances like TVs with Internet access and thermostats that can be controlled remotely from miles away of its location. This phenomenon is known as Internet of Things (IoT). This kind of devices have many advantages, but they also pose a challenge regarding security flaws that could risk user's privacy. Furthermore, and considering the number of different manufacturers, the homogeneity across these devices is nearly non-existing. The present work uses network traffic generated from the behavior of common IoT devices found in home networks and captured leveraging a testbed developed as part of it, applying different analysis techniques like deep packet inspection or flow-based analysis to better understand the use of these techniques as well as explore new ways of advanced network data enrichment and visualization.

Palabras clave: IoT, Networks, Analysis, Security

Firma del alumno:	**VºBº Tutor/es:**
	GARCIA RODRIGUEZ ISAIAS - 09778112F Firmado digitalmente por GARCIA RODRIGUEZ ISAIAS - 09778112F Fecha: 2022.02.21 10:35:29 +01'00'

Resumen

En la actualidad multitud de dispositivos se conectan entre sí para intercambiar información. La importancia de dicha interconexión y la gran variedad de servicios ofrecidos a través de la red de redes (Internet) ha alcanzado lo más profundo de los hogares de las personas. Hoy en día es común encontrar televisores con acceso a Internet o termostatos que se pueden controlar desde miles de kilómetros de distancia. A este fenómeno de conectar a Internet dispositivos de uso cotidiano que antes no disponían de dicho acceso a la red, se le conoce como Internet de las cosas. Al margen de este fenómeno y todas las ventajas que trae consigo, este tipo de dispositivos pueden presentar problemas de seguridad que los hace objetivo de ataques. Además, y dada la diversidad de fabricantes, la homogeneidad en cuanto a los métodos de conexión utilizados es inexistente. Este trabajo utiliza el tráfico generado por el comportamiento de estos dispositivos en las redes domésticas de los usuarios capturado mediante un entorno de pruebas desarrollado, y aplica diferentes técnicas de análisis como la inspección profunda de paquetes o el análisis basado en flujos con el fin de estudiar y profundizar en el uso de dichas técnicas, así como explorar formas avanzadas de enriquecimiento y visualización de los datos.

Abstract

Nowadays, many devices interconnect each other to exchange information. The importance of that interconnection and the wide range of available services on the Internet has reach people homes. Today, it is usual to find common appliances like TVs with Internet access and thermostats that can be controlled remotely from miles away of its location. This phenomenon is known as Internet of Things (IoT). This kind of devices have many advantages, but they also pose a challenge regarding security flaws that could risk user's privacy. Furthermore, and considering the number of different manufacturers, the homogeneity across these devices is nearly non-existing. The present work uses network traffic generated from the behavior of common IoT devices found in home networks and captured leveraging a testbed developed as part of it, applying different analysis techniques like deep packet inspection or flow-based analysis to better understand the use of these techniques as well as explore new ways of advanced network data enrichment and visualization.

Índice de Contenidos

Índice de Figuras

Índice de Tablas

Lista de Acrónimos y Siglas

Sigla	Significado
BLE	Bluetooth Low Energy
CPU	Central Processing Unit
DDS	Data Distribution Service
DNS	Domain Name System
DPI	Deep Packet Inspection
EC2	Elastic Compute Cloud
ELK	Elasticsearch Logstash Kibana
GPU	Graphics Processing Unit
HTTPS	HyperText Transfer Protocol Secure
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
IGMP	Internet Group Management Protocol
IP	Internet Protocol
IPFIX	Internet Protocol Flow Information Export
ISP	Internet Service Provider
ISP	Internet Service Provider
IoT	Internet of Things
JSON	JavaScript Object Notation
MAC	Media Access Control
mDNS	Multicast DNS
MQTT	Message Queing Telemetry Transport
MUD	Manufacturer Usage Description
PDU	Protocol Data Unit
QoS	Quality of Service
RAM	Random Access Memory
RTSP	Real-Time Streaming Protocol

SBC	Simple Board Computer
SQL	Structured Query Language
SSDP	Simple Service Discovery Protocol
SoC	System on Chip
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
UPnP	Universal Plug and Play

1. Introducción

Los dispositivos *IoT (Internet of Things)* se pueden definir como aparatos cuyo propósito es muy específico [1] y que tienen la habilidad de transmitir y recibir datos utilizando diferentes medios de comunicación. Entre estos medios de comunicación se encuentran multitud de diferentes protocolos como *MQTT (Message Queing Telemetry Transport)* o *DDS (Data Distribution Service)*; y tecnologías de comunicación como *Wi-Fi, Ethernet, Zigbee* o *BLE (Bluetooth Low Energy)*. Teniendo en cuenta la gran diversidad de dispositivos y heterogeneidad de sistemas y protocolos de comunicación, para la realización de este trabajo se ha seleccionado un conjunto de dispositivos que utilizan *Ethernet*, *Wi-Fi* o *BLE* y utilizan conexiones directas o a través de controladores que se comunican con el dispositivo final por medio de otros protocolos (*Zigbee*). En el siguiente diagrama se muestra, de manera sintetizada, los diferentes tipos de esquema de conexión que utilizan dichos dispositivos:

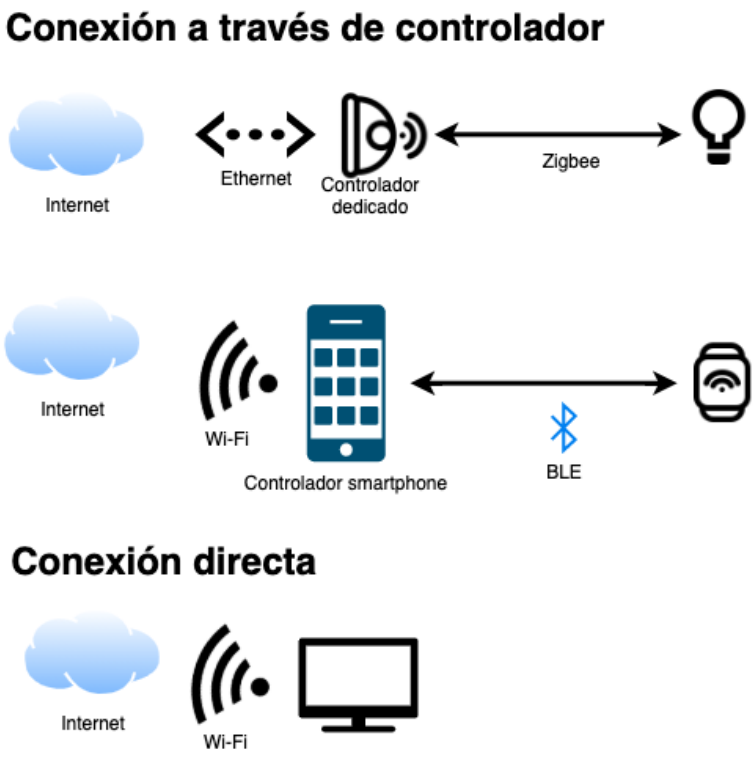


Figura 1.1 Tipos de conexiones utilizadas por dispositivos *IoT*

Como se puede observar en el diagrama, ciertos dispositivos utilizan un conector propietario o *hub* para gestionar la conexión entre el dispositivo e Internet, considerando además que la conexión

entre dispositivo y controlador se realiza utilizando diversas tecnologías de comunicación, como *ZigBee*, con el fin de minimizar el impacto energético de establecer dichas comunicaciones por los dispositivos finales. El proyecto desarrollado consiste en la creación de un entorno de pruebas para canalizar el tráfico producido por estos dispositivos y el análisis posterior de dicho tráfico utilizando para ello diferentes técnicas; herramientas clásicas y muy conocidas de inspección profunda de paquetes, y otras más recientes y menos extendidas basadas en la extracción de características del tráfico utilizando métodos basados en flujos, utilizando como muestra dispositivos *IoT* que utilizan ambos esquemas de conexión basados en tecnología alámbrica *Ethernet* e inalámbricas *Wi-Fi* y *BLE*.

La motivación de este trabajo reside en el deseo de aplicar las metodologías y técnicas más relevantes y actuales en cuanto al análisis de tráfico de red, enfocando dicho estudio en el tráfico generado por los dispositivos del Internet de las Cosas, profundizar en el análisis general de tráfico de red y construir un entorno de pruebas para llevar a cabo dicho análisis para otros dispositivos en el futuro, así como explorar técnicas avanzadas de enriquecimiento y visualización de los datos para poder extraer más información del comportamiento de este tipo de dispositivos y los riesgos de ciberseguridad implícitos de dicho comportamiento.

En los primeros capítulos de esta memoria se exponen el ámbito de trabajo y, en particular, los objetivos del trabajo y el estado de la cuestión. A continuación, se incluye un capítulo de planificación en el que se recogen los aspectos organizativos y tareas que componen el trabajo. En cuanto al desarrollo central del trabajo se ha dividido en los siguientes capítulos:

- Creación entorno de pruebas sobre *Raspberry Pi 4*, en el que se tratan los pasos seguidos para la creación de un entorno de pruebas para la captura de tráfico de dispositivos, utilizando la *Raspberry Pi 4* como switch de red e interfaz de captura de tráfico *BLE*.
- Captura de tráfico de red, en el que se recogen las diferentes configuraciones utilizadas para la captura de diferentes conjuntos de tráfico de red de distintos dispositivos y utilizando diferentes técnicas.
- Análisis de tráfico de red, en el que se recogen las diferentes técnicas de análisis utilizadas tomando como punto de partida los diferentes conjuntos de datos capturados en el capítulo anterior.

Por último se incluye un capítulo de conclusiones en el que se recogen los resultados y pareceres obtenidos tras la realización de este trabajo.

2. Objetivos

El objetivo principal de este trabajo es el estudio y análisis de diferentes técnicas de análisis de tráfico utilizando los paquetes de información o tráfico de red producido por dispositivos que se engloban en el fenómeno del Internet de las cosas, y particularmente en aquellos dispositivos que se puedan encontrar en el ámbito doméstico como *wereables* o dispositivos *Smart Home* con el fin de aprender sobre su comportamiento en cuanto a la generación de tráfico de red y los riesgos de ciberseguridad implícitos al mismo. Para ello, se han llevado a cabo las siguientes tareas:

- Por una parte, se ha construido un *testlab* o entorno de pruebas utilizando una *Raspberry Pi* y configurándola como un *switch* de red que ha servido para la captura del tráfico de red producido por diferentes dispositivos *IoT*.
- Además, se han utilizado distintos dispositivos *IoT* que, conectándolos de forma inalámbrica (*Wi-Fi, BLE*) y alámbrica (*Ethernet*) a través del entorno de pruebas desarrollado, ha sido posible capturar el tráfico de red generado por los mismos, teniendo en cuenta la diversidad en los tipos de conexión de estos dispositivos.
- Por último, se han utilizado diferentes técnicas de análisis como la inspección profunda de paquetes y el análisis basado en flujos con el fin de conocer dichas técnicas en profundidad, conocer los riesgos de seguridad que exponen algunos de los dispositivos utilizados, así como explorar técnicas de enriquecimiento y visualización de los datos recolectados.

3. Estado del arte

El avance y progreso en el número y diversidad de estudios que tienen por objeto los dispositivos *IoT* se debe principalmente a su crecimiento y adopción en los últimos años, tanto en el ámbito doméstico como en el industrial, siendo 2020 [2] el año en el que el número de conexiones de dispositivos *IoT* ha superado al de conexiones del resto de dispositivos:

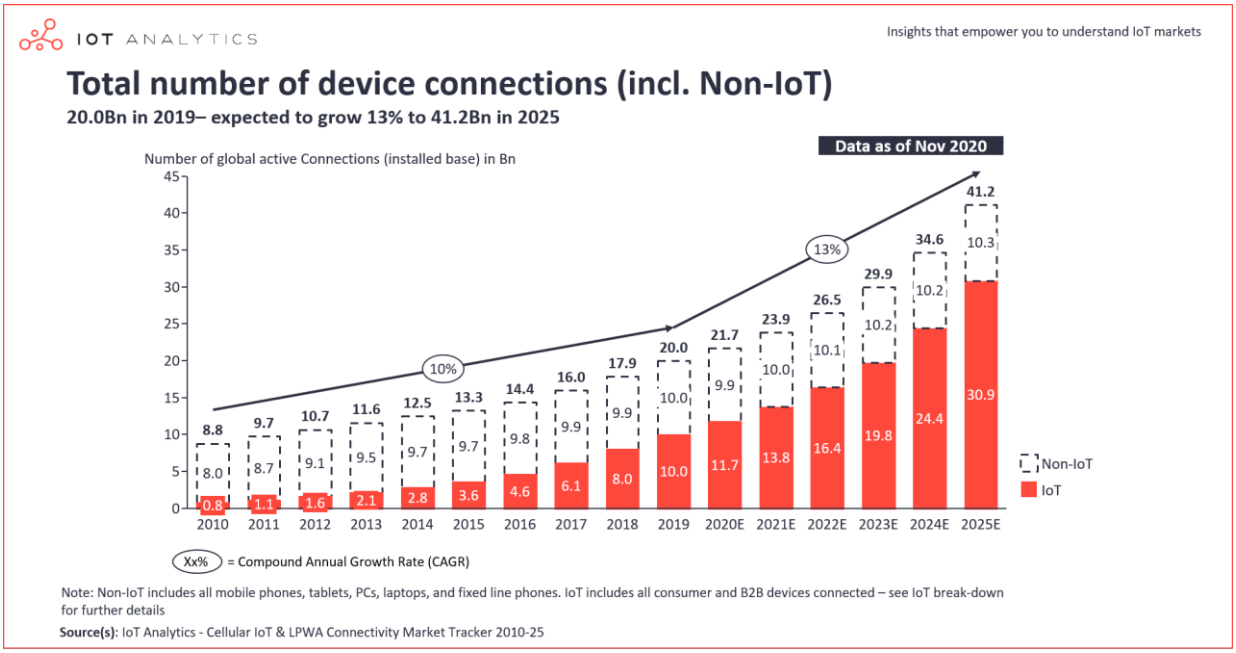


Figura 3.1 Número total de conexiones de dispositivos a Internet (Fuente: https://iot-analytics.com/state-of-the-iot-2020-12-billion-iot-connections-surpassing-non-iot-for-the-first-time/)

Además, el hecho de que este tipo de dispositivos no estén diseñados con la seguridad como principal objetivo también ha influido en el interés de los investigadores y expertos en ciberseguridad, pues estos dispositivos han sido objeto de diversos ataques y brechas de seguridad [3] que han comprometido, no sólo la privacidad de sus usuarios, sino que también han sido objeto de infecciones de malware, convirtiéndolos en parte de una red automatizada para la infección de más dispositivos (*IoT Botnet*), posteriormente utilizadas para ataques a mayor escala. En este sentido y con el fin de ayudar a la investigación y desarrollo de soluciones para mitigar este tipo de amenazas de ciberseguridad, se han publicado conjuntos de datos [4] que contienen tanto tráfico maligno como benigno de este tipo de dispositivos.

Dada la heterogeneidad y el reto que supone la gestión de redes con este tipo de dispositivos, también han surgido trabajos [5, 6] relacionados con el análisis general del tráfico *IoT* en redes domésticas generado por dispositivos específicos y que son comunes en este ámbito, aportando características propias que pueden servir para identificarlos, así como los riesgos de privacidad y seguridad que dichos dispositivos manifiestan. En esta misma línea de investigación, algunos trabajos también se centran en la optimización de los recursos de red mediante la integración de sistemas de calidad del servicio (*QoS o Quality of Service*) [7] mientras que otros trabajos existentes se centran en aplicar diferentes técnicas de monitorización basada en flujos y apoyadas en la extracción de características de los mismos para el reconocimiento del tráfico generado por estos dispositivos [1] y la creación de modelos [8], siendo el objetivo principal la automatización y creación de algoritmos [9] capaces de identificar dichos dispositivos, aplicando diferentes algoritmos de aprendizaje automático o *Machine Learning* para ello [10].

Cabe destacar que el grupo trabajo de la universidad de Nueva Gales del sur (*UNSW Sydney*) ha publicado diferentes conjuntos de datos recopilados en un entorno de pruebas similar al propuesto en este trabajo [11], desarrollando además un generador de perfiles *MUD* (*Manufacturer Usage Description*) [12] a partir del tráfico de red capturado para un dispositivo específico. Estos perfiles forman parte de una iniciativa de la *IETF* (*Internet Engineering Task Force*) cuyo fin es que los fabricantes de dispositivos *IoT* desarrollen y publiquen especificaciones formales de las intenciones o uso normal de dichos dispositivos, de tal forma que el comportamiento en cuanto al tráfico de red generado por los dispositivos pueda ser verificado de forma rigurosa. Dicho perfil *MUD* se puede representar a partir de un diagrama *sankey,* que muestra los flujos y la relación proporcional entre ellos (cuanto mayor es el ancho de un flujo, mayor es la cantidad de este) y puede dividirse mediante colores y etapas para mostrar las transiciones existentes de un proceso a otro del sistema. Estos diagramas se utilizan comúnmente para representar transferencias de energía o dinero.

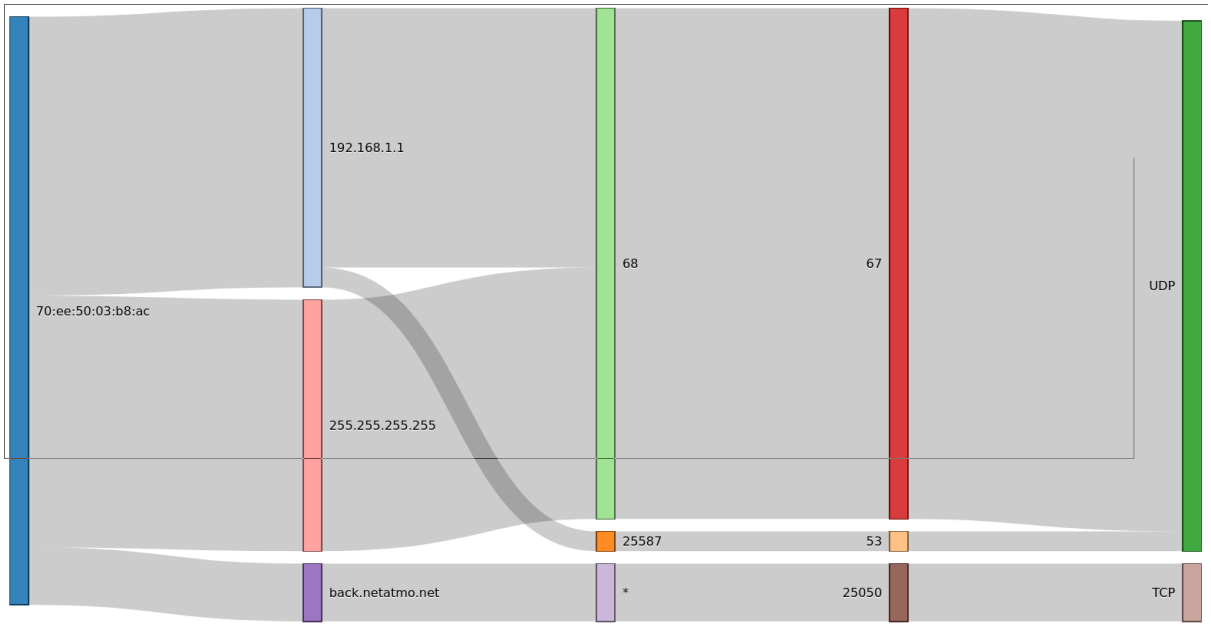


Figura 3.2 Representación perfil MUD con diagrama *sankey* (Fuente: https://iotanalytics.unsw.edu.au/profiles)

Por otro lado, ciertos dispositivos cifran sus conexiones haciendo uso de protocolos criptográficos, siendo el más común *TLS (Transport Layer Security)*, dificultando así su identificación y actividades en la red, particularmente desde el punto de vista de los proveedores de servicios de internet (*ISP o Internet Service Provider*) [13]. A este respecto, los esfuerzos se centran principalmente en conseguir identificar el tráfico de dispositivos *IoT* aun estando cifrado mediante técnicas de *fingerprinting* [14] que en el caso del protocolo *TLS*, protocolo más utilizado en Internet para la encriptación de las comunicaciones, consiste en la recopilación de información de la comunicación del protocolo que no está encriptada, en concreto el paquete *Client Hello* que además de iniciar la comunicación incluye diversos parámetros y que, a partir de las combinaciones de estos, es posible crear una base de datos [15, 16] de *fingerprints* conocidos.

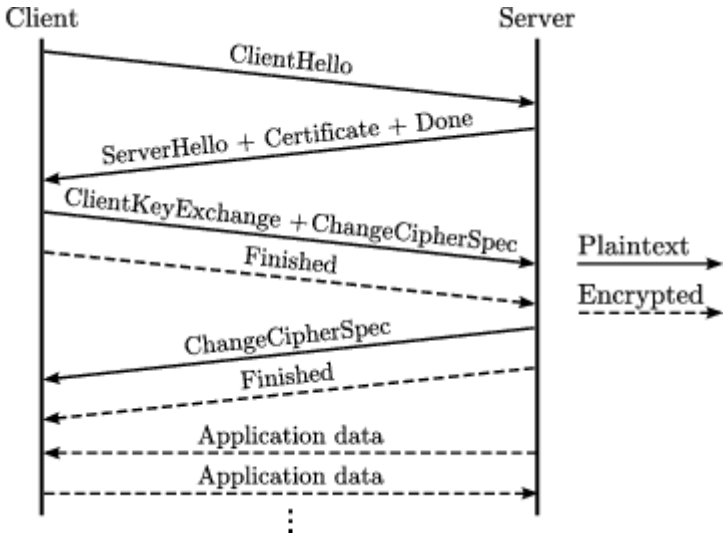


Figura 3.3 Esquema de negociación del protocolo *TLS* (Fuente: [16])

Para la generación y uso de dichos *fingerprints* existen diferentes utilidades y formatos de estos como *cisco/joy* o *JA3* [17, 18], aunque, en definitiva, un *fingerprint* es una cadena de caracteres con formato hexadecimal que condensa los diferentes campos característicos que se encuentran en el paquete *Client Hello*.

Por último, y siguiendo líneas de investigación más generales en el área del análisis forense, otros trabajos existentes [19] repasan las diferentes técnicas y procedimientos del análisis forense, y en particular aquel cuyo objeto de estudio principal es el tráfico de red, como *packet sniffing*, técnica de recolección y análisis de paquetes que fluyen por una red, o *IP (Internet Protocol) traceback,* que aglutina los métodos existentes que existen para determinar de manera rigurosa el origen de un paquete.

4. Planificación

En esta sección se recogen los aspectos organizativos y de planificación del trabajo y las diferentes partes que lo componen.

4.1. CREACIÓN DEL ENTORNO DE PRUEBAS

Para la creación del entorno de pruebas se han llevado a cabo las tareas que se muestran en la tabla siguiente:

Tabla 4.1 Relación de tareas: creación entorno de pruebas.

Tarea		Dependencias	Inicio	Fin
A	Documentación e investigación.	-	23/02/2021	25/03/2021
B	Recopilación de los materiales necesarios.	A	26/03/2021	15/04/2021
C	Configuración inicial de *Raspberry Pi*.	B	16/04/2021	06/05/2021
D	Creación del entorno de pruebas.	C	07/05/2021	10/06/2021

Para situar estas tareas en el calendario, se incluye el diagrama de Gantt correspondiente:

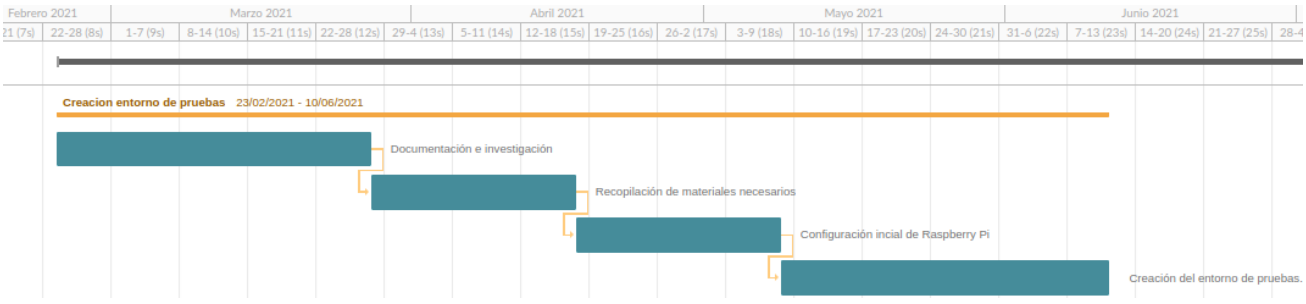


Figura 4.1 Diagrama de Gantt: planificación de la creación del entorno de pruebas

4.2. CAPTURA DE TRÁFICO DE RED

Para la captura de tráfico de red de los diferentes dispositivos, se han llevado a cabo las siguientes tareas:

Tabla 4.2 Relación de tareas: captura de tráfico de red.

Tarea		Dependencias	Inicio	Fin
A	Documentación e investigación.	-	10/06/2021	12/07/2021
B	Recopilación de dispositivos.	-	10/06/2021	11/08/2021
C	Captura de tráfico de báscula Bluetooth.	B	11/08/2021	25/08/2021
D	Captura de tráfico de cámara IP.	B	11/08/2021	01/09/2021
E	Captura de tráfico de varios dispositivos.	B	11/08/2021	06/09/2021

Para situar estas tareas en el calendario, se incluye el diagrama de Gantt correspondiente:

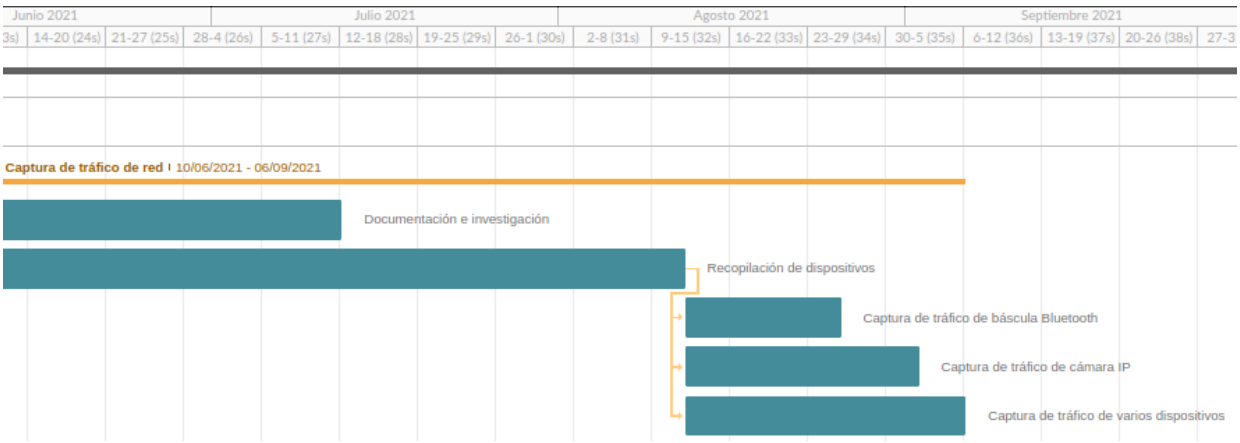


Figura 4.2 Diagrama de Gantt: planificación de la captura de tráfico de red.

4.3. ANÁLISIS DE TRÁFICO DE RED

Para el análisis de tráfico de las distintas capturas de datos de los diferentes dispositivos, se han llevado a cabo las siguientes tareas:

Tabla 4.3 Relación de tareas: análisis de tráfico de red.

Tarea		Dependencias	Inicio	Fin
A	Documentación e investigación.	-	06/09/2021	27/09/2021
B	Ingeniería inversa del *payload* de tráfico BLE	A	27/09/2021	18/10/2021
C	Reconstrucción de *streaming* RTP	A	18/10/2021	08/11/2021
D	Uso de flujos para el filtrado de paquetes	A	08/11/2021	13/12/2021
E	Enriquecimiento y visualización de flujos utilizando ELK.	A	08/11/2021	28/12/2021

Para situar estas tareas en el calendario, se incluye el diagrama de Gantt correspondiente:

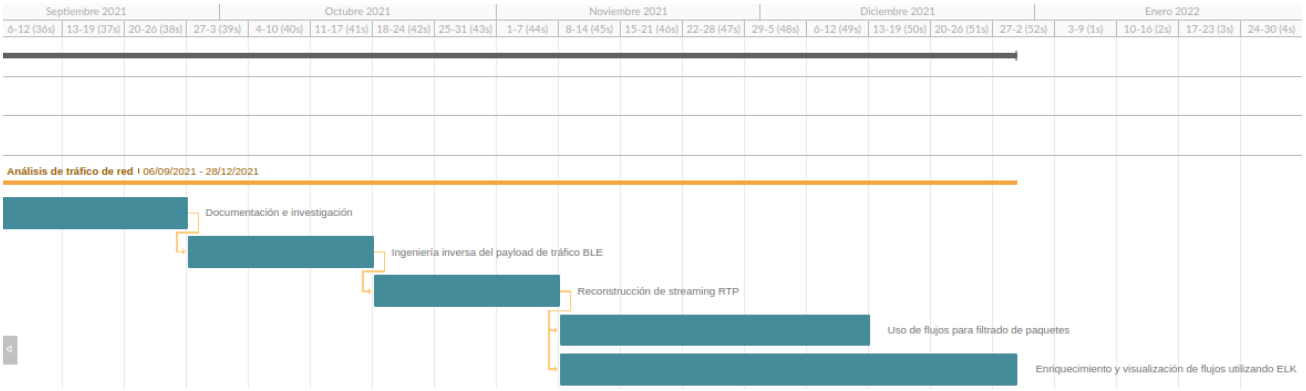


Figura 4.3 Diagrama de Gantt: planificación del análisis de tráfico de red.

4.4. VISTA GENERAL DE LA PLANIFICACION

Aglutinando todas las tareas de las diferentes partes del proyecto, se puede calcular la ruta crítica que identifica el conjunto de tareas mínimo e interdependientes entre sí que, en caso de sufrir algún retraso, impactaría directamente sobre la fecha de finalización del proyecto:

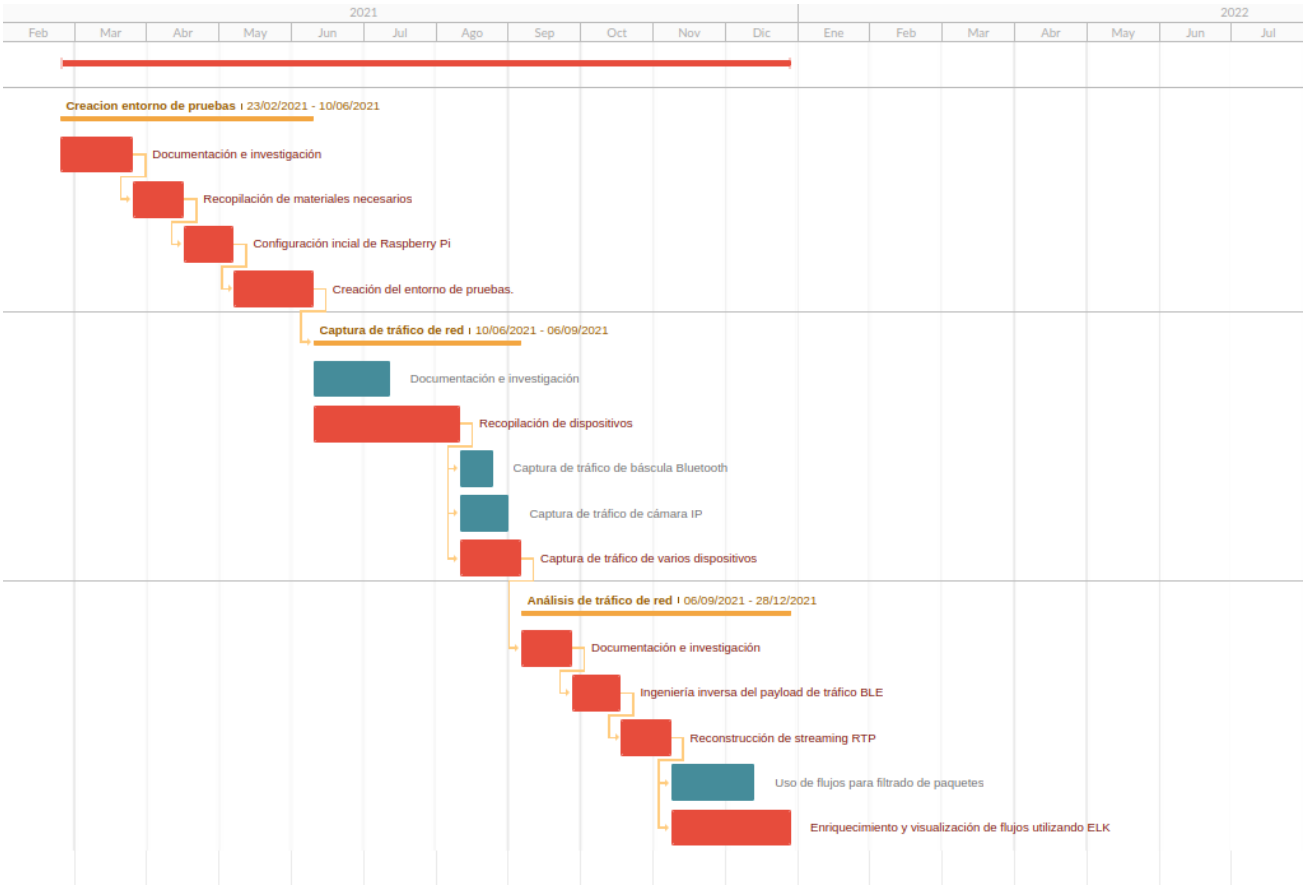


Figura 4.4 Diagrama de Gantt: planificación y ruta crítica del proyecto.

5. Creación del entorno de pruebas

En primer lugar, en este trabajo se ha construido un *testlab* o entorno de pruebas que ha servido para la captura del tráfico de red producido por diferentes dispositivos *IoT*. Para la creación de este entorno se ha utilizado como elemento principal una *Raspberry Pi 4*, y conectado a la misma unos adaptadores *USB* a *ethernet* y el adaptador *Adafruit LE Sniffer*, y se ha configurado como un *switch* de red.

5.1. ESQUEMA DE FUNCIONAMIENTO

El entorno de pruebas creado sirve como *switch* de red para habilitar la conectividad de los dispositivos o sus controladores con Internet, además de incluir una interfaz de captura *BLE* para capturar el tráfico de dispositivos cercanos con el fin de aplicar técnicas de análisis al tráfico generado por estos dispositivos.

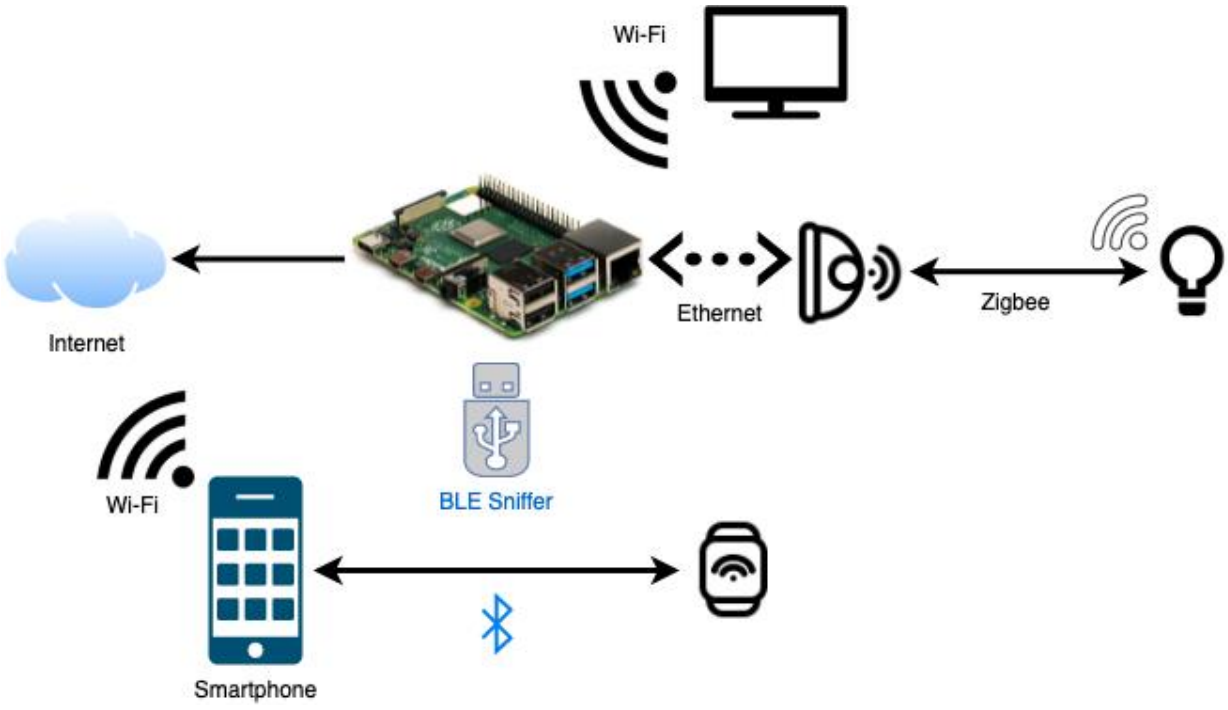


Figura 5.1 Esquema de funcionamiento del entorno de pruebas

Para su construcción, se ha utilizado una *Raspberry Pi* [20] computador de placa única o de placa simple (*SBC o Simple Board Computer*) de bajo coste y tamaño reducido. Creada por la fundación que recibe el mismo nombre y administrada por Eben Upton, la *Raspberry Pi* está inspirada en el computador *BBC Micro* y su objetivo es acercar la computación y la programación al ámbito educacional, aunque gracias a su reducido tamaño y su modesta potencia, ha sido adoptado como plataforma para proyectos electrónicos y *software* más avanzados.

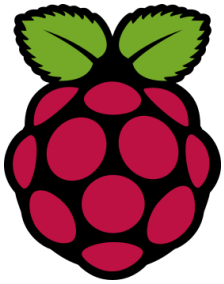


Figura 5.2 Logo de *Raspberry Pi* (Fuente: https://en.wikipedia.org/wiki/File:Raspberry_Pi_Logo.svg)

Desde su creación, han salido diferentes modelos de este computador al mercado, ampliando la capacidad de procesamiento y la memoria. Para la realización de este proyecto se ha escogido la *Raspberry Pi 4 model B*:

Figura 5.3 *Raspberry Pi 4 Model B*

En la tabla siguiente se indican las características técnicas más relevantes de este modelo:

Tabla 5.1 Características técnicas Raspberry Pi 4 Model B (Fuente: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/)

SOC (System On Chip)	*Broadcom BCM271*
CPU (Central Processing Unit)	*Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz*
GPU (Graphics Processing Unit)	*VídeoCore VI*
RAM (Random Access Memory)	*2GB LPDDR4-3200 SDRAM*

Además, con el fin de ampliar la cantidad de conexiones alámbricas que puede ofrecer el entorno de pruebas, se han conectado dos adaptadores *USB (Universal Serial Bus)* a *Ethernet*:

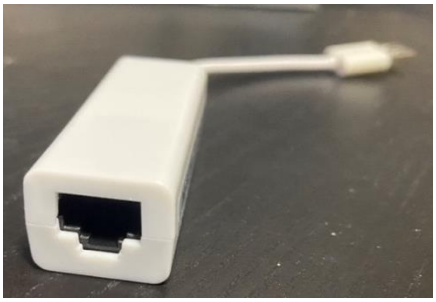


Figura 5.4 Adaptador *USB* a *Ethernet*

5.2. INSTALACIÓN Y CONFIGURACIÓN DE SWITCH DE RED

Para facilitar la recreación de este entorno de pruebas, se han desarrollado una serie de *scripts* y documentación publicada en formato de repositorio de código fuente de forma pública: *https://github.com/ruben-rodriguez/iot-network-analysis*.

Se ha configurado la *Raspberry Pi* como un *switch* de nivel 2 utilizando para ello un *bridge* o puente mediante el software *bridge-utils*. Un *bridge* [21] es un *software* que se utiliza para unir dos o más segmentos de red, comportándose como un *switch* de red virtual y que funciona de manera transparente (el resto de los dispositivos no conocen su existencia) permitiendo conectar al mismo cualquier dispositivo real (*eth0*) o virtual, siendo la diferencia principal con un *switch* dedicado el número de puertos disponibles, contando con un número más elevado este último. En el *ANEXO I: CREACIÓN DEL ENTORNO DE PRUEBAS PARA LA CAPTURA DE TRÁFICO*, se puede encontrar

información técnica detallada para conseguir recrear este entorno de pruebas utilizando una serie de *scripts* para automatizar el proceso.

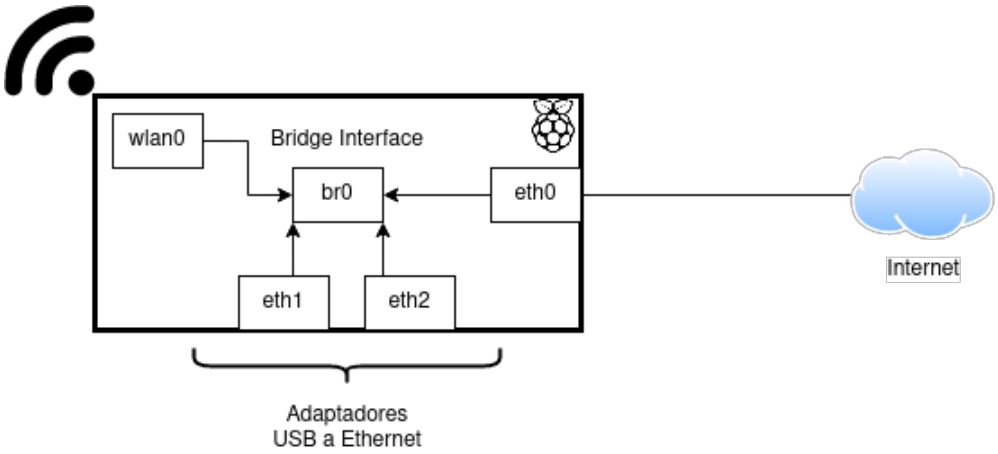


Figura 5.5 Diagrama funcionamiento interno entorno de pruebas

A continuación, se describen los pasos para la configuración de la *Raspberry Pi* como un *switch* de red utilizando *Raspbian* como sistema operativo:

1. Instalación de los paquetes *software* requeridos, en concreto *bridge-utils* que sirve para crear *bridges,* y *hostapd,* que sirve para la creación de puntos de acceso Wi-Fi:
   `$: apt-get install -y bridge-utils hostapd`
2. Configuración del punto de acceso Wi-Fi que formará parte como interfaz del *switch* en el directorio */etc/hostapd/hostapd.conf.*

```
# Bridge name hostapd will be part of
bridge=br0
# Channel to use
channel=4
# Wifi driver to use
driver=nl80211
# 2.4GHz band
hw_mode=g
# What interface to bound hostapd to
interface=wlan0
# SSID name
ssid=IOT_TEST_LAB
# Auth section
wpa=1
wpa_passphrase=iottestlab
wpa_key_mgmt=WPA-PSK
```

Figura 5.6 Configuración *hostapd* para crear punto de acceso Wi-Fi

3. Configuración de las diferentes interfaces en un *bridge:*

```
# interfaces(5) file used by ifup(8) and ifdown(8)

# Include files from /etc/network/interfaces.d:
source-directory /etc/network/interfaces.d

auto lo
iface lo inet loopback

# Using 3 ports (built-in + 2 usb dongles) and wlan0
allow-hotplug eth0
iface eth0 inet manual
   pre-up   ifconfig $IFACE up
   pre-down ifconfig $IFACE down

allow-hotplug eth1
iface eth1 inet manual
   pre-up   ifconfig $IFACE up
   pre-down ifconfig $IFACE down

allow-hotplug eth2
iface eth2 inet manual
   pre-up   ifconfig $IFACE up
   pre-down ifconfig $IFACE down

allow-hotplug wlan0
iface wlan0 inet manual
 up hostapd -B /etc/hostapd/hostapd.conf

auto br0
iface br0 inet static
  bridge_ports eth0 eth1 eth2 wlan0
  address 192.168.1.156
  broadcast 192.168.1.255
  netmask 255.255.255.0
```

Figura 5.7 Configuración interfaces de red para crear un *bridge*

4. Reinicio de los servicios de red y *hostapd* para que la configuración nueva sea aplicada:

```
$: systemctl stop networking
$: systemctl unmask hostapd
$: systemctl start hostapd
$: systemctl start networking
```

5. Modificación de las rutas del *kernel* y configuración de la interfaz principal de red (*eth0*) para desasignar su *IP*:

```
$: route add default gw 192.168.1.1 br0
$: route del default gw 192.168.1.1 eth0
$: ifconfig eth0 0.0.0.0
```

6. Verificación de la configuración utilizando los comandos de *bridge-utils:*

```
$: brctl show
$: brctl showstp br0
```

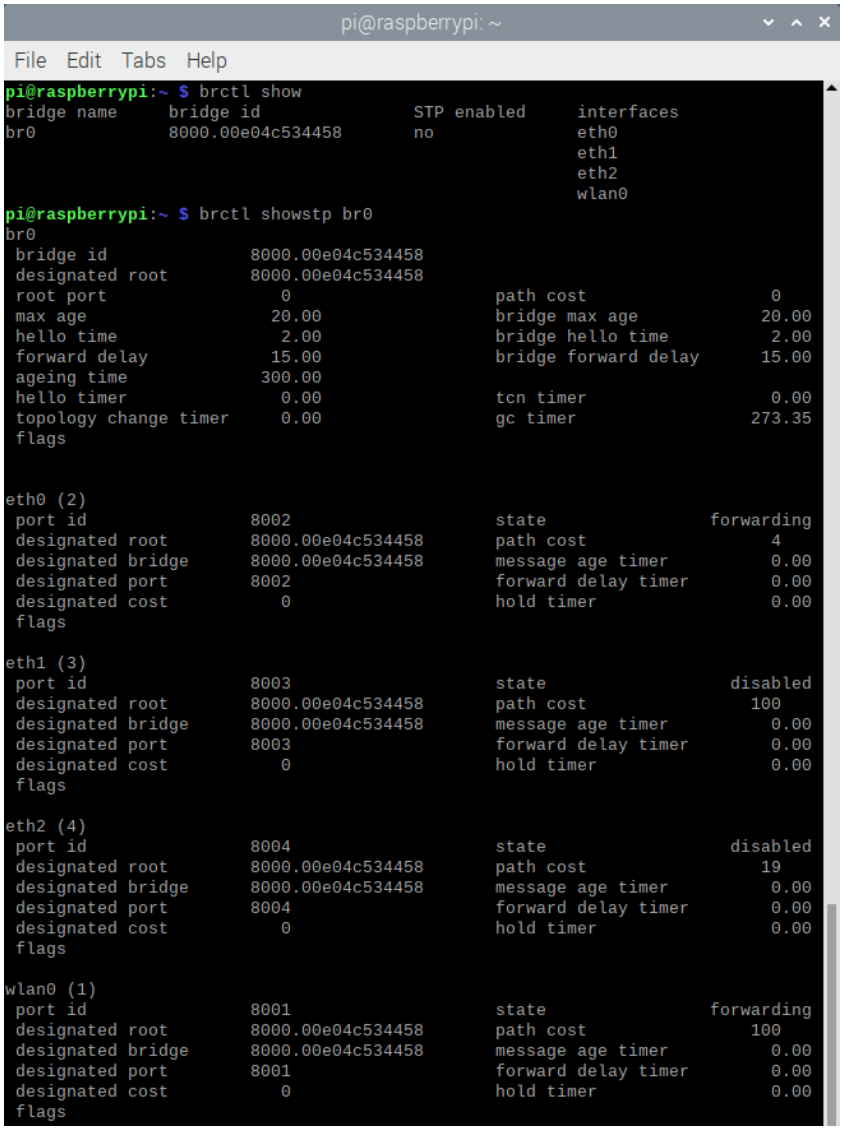


Figura 5.8 Comprobación del estado de bridge configurado en *Raspberry Pi*

5.3. INSTALACIÓN Y CONFIGURACIÓN DE CAPTURA DE INTERFAZ BLUETOOTH LOW ENERGY

Además, con el fin de poder interceptar las comunicaciones que hacen uso de *BLE*, se ha incorporado en el entorno de pruebas un adaptador *USB Adafruit LE Sniffer* [22], dispositivo que cuenta con un *chip nRF51822* y que junto con un *firmware* especial permite interceptar el tráfico *BLE* de dispositivos cercanos:

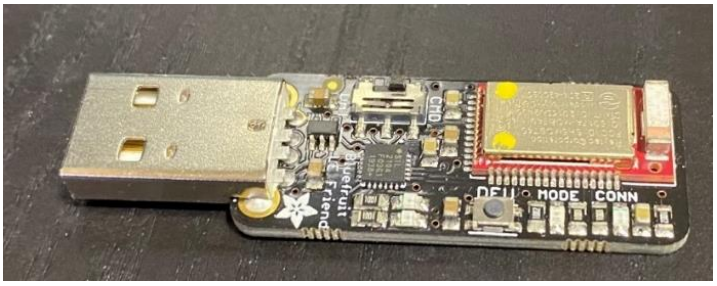


Figura 5.9 Adaptador *USB Adafruit LE Sniffer*

A continuación, se incluye un diagrama de funcionamiento de estos elementos en conjunto:

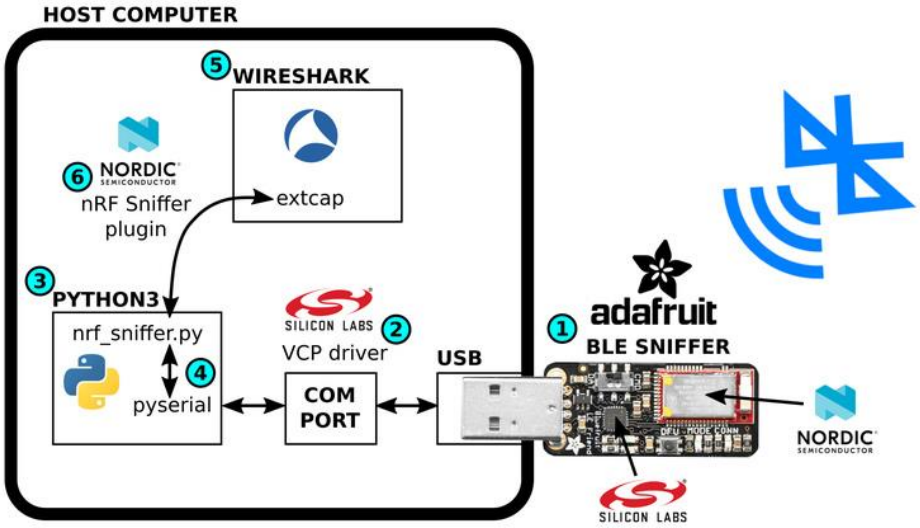


Figura 5.10 Diagrama funcionamiento *Adafruit Bluefruit LE Sniffer* (Fuente: https://learn.adafruit.com/introducing-the-adafruit-bluefruit-le-sniffer)

A continuación, se describen los pasos para la instalación:

1. Una vez conectado el adaptador *USB Adafruit LE Sniffer* a la *Raspberry Pi*, se han de activar los módulos correspondientes en el *kernel* de Linux:

   ```
   $: modprobe usbserial
   $: modprobe cp210x
   ```

2. Instalación de los requisitos (*Python*) del *plugin*:

   ```
   $: pip install psutil pyserial serial
   ```

3. Descarga del *zip* del *plugin*:

   ```
   $: wget https://www.nordicsemi.com/-/media/Software-and-other-downloads/Desktop-software/nRF-Sniffer/sw/nrf_sniffer_for_bluetooth_le_4.0.0.zip
   $: unzip nrf_sniffer_for_bluetooth_le_4.0.0.zip
   ```

4. Una vez descomprimido, se copian los contenidos de la carpeta *extcap* del *zip* en la carpeta de *Wireshark* que aparece en la configuración como carpeta global *extcap.*

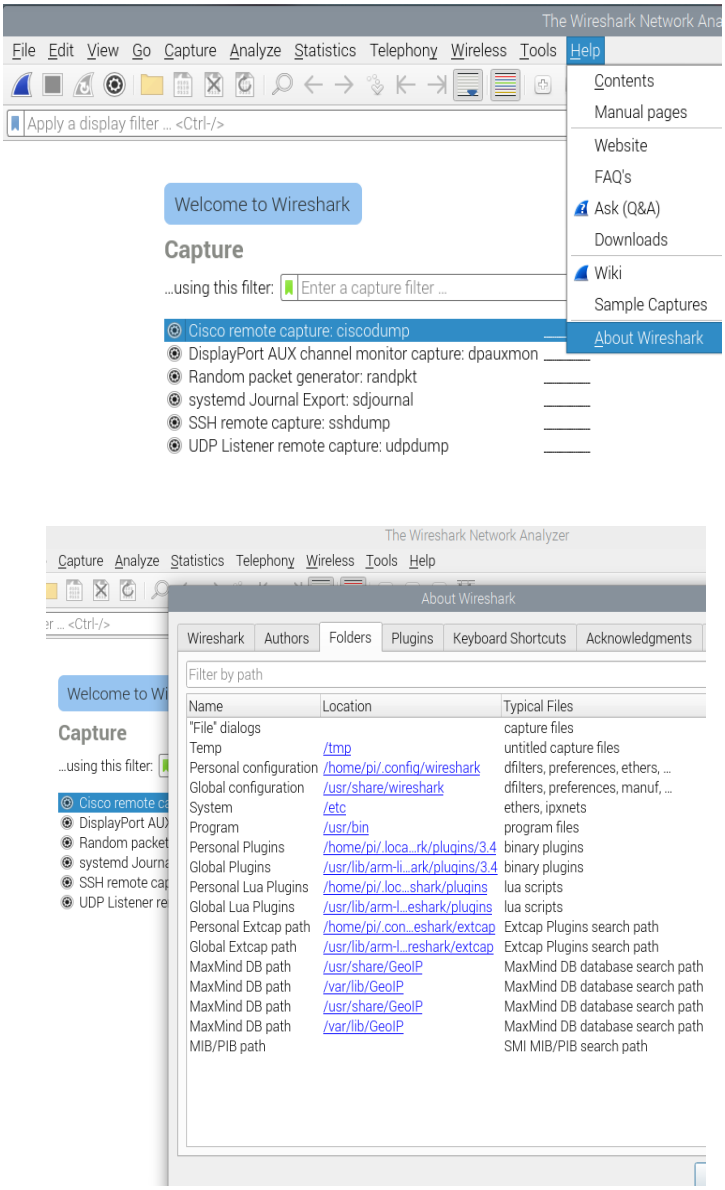


Figura 5.11 Configuración de carpetas *Wireshark*

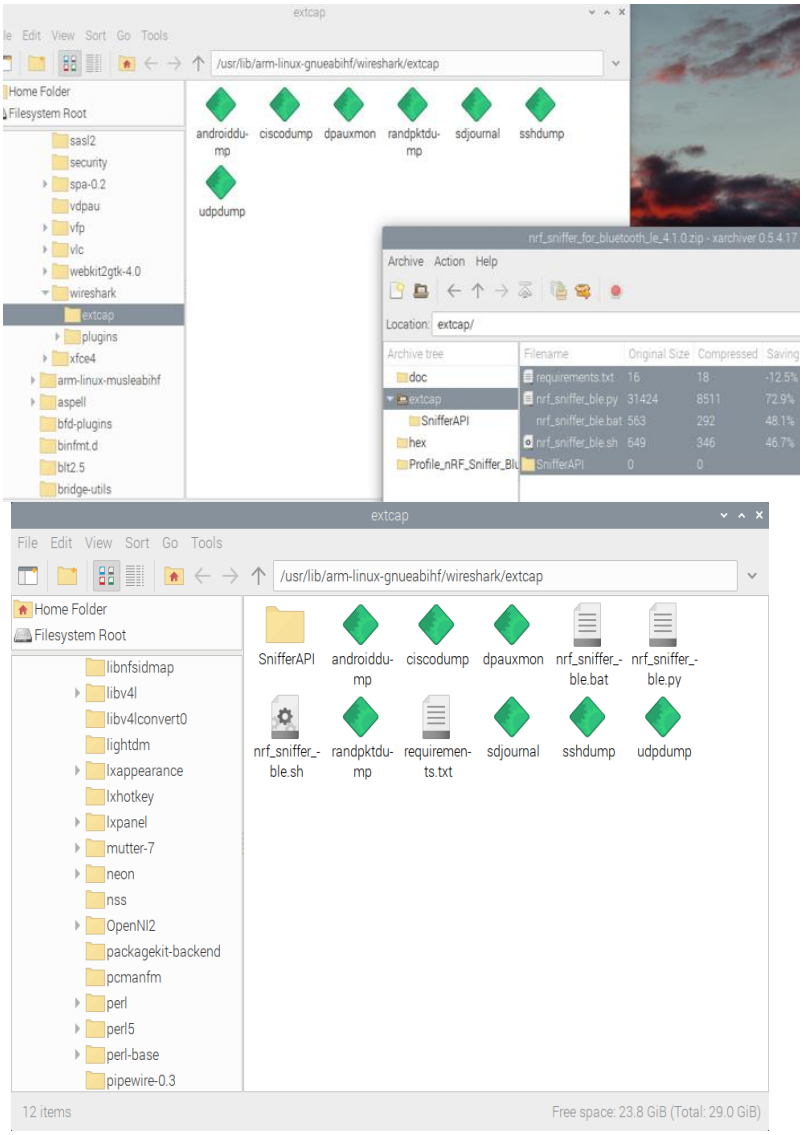


Figura 5.12 Instalación de *plugin Adafruit* en *Wireshark*

Una vez instalado el *plugin* y reiniciado *Wireshark,* la opción para capturar tráfico desde la interfaz del adaptador *USB* estará disponible:

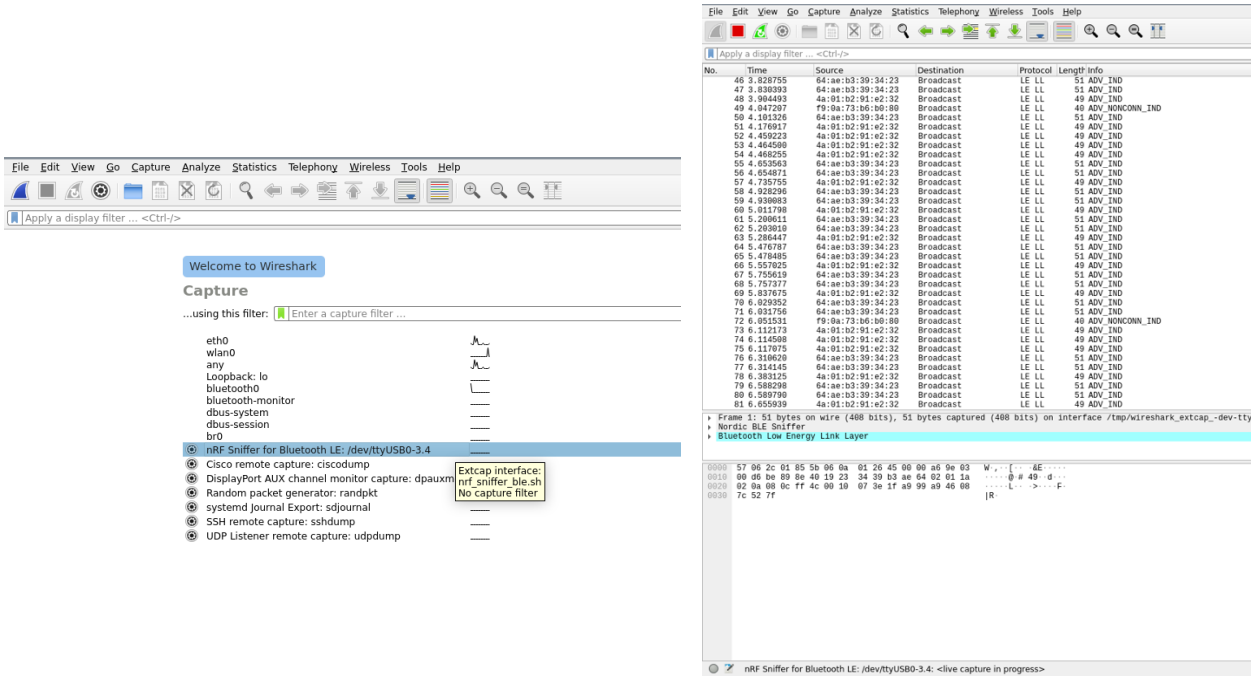


Figura 5.13 Captura de tráfico *BLE* utilizando el adaptador *USB*

6. Captura de tráfico de red

Con el fin de poner en práctica diferentes técnicas de análisis del tráfico, se han realizado una serie de capturas de las trazas tráfico utilizando el entorno de pruebas creado y diferentes dispositivos.

6.1. CAPTURA DE PAQUETES DE COMUNICACIÓN BÁSCULA INTELIGENTE

La gran mayoría de dispositivos conocidos como *weareables* utilizan la comunicación *BLE* o *Bluetooth Low Energy*. Esta tecnología *bluetooth* [23] se centra en permitir operaciones de comunicación entre dispositivos con un coste energético bajo. Además, permite diversas topologías (punto a punto, *broadcast*, *mesh*) y permite una gran flexibilidad a los desarrolladores para crear productos que requieran de comunicación con un bajo impacto energético. Gracias a la incorporación del *BLE sniffer* en el entorno de pruebas desarrollado, fue posible capturar el intercambio de paquetes *BLE* por una báscula y su aplicación acompañante instalada en un *smartphone*:

Figura 6.1 Báscula inteligente *BLE*

Dicha comunicación acontece cada vez que un usuario se sube a la báscula y efectúa una operación de pesado. Para su captura, se ha utilizado *Wireshark* junto con el controlador de *Silicon Labs* y el *plugin* necesario para que *Wireshark* permita utilizarlo como interfaz de captura, cuya instalación

está descrita en la sección 5.3 *INSTALACIÓN Y CONFIGURACIÓN DE CAPTURA DE INTERFAZ BLUETOOTH LOW ENERGY* de esta memoria.

6.2. CAPTURA DE PAQUETES DE COMUNICACIÓN CÁMARA IP

Shenzhen Sricctv Technology CO., LTD es un fabricante [24] y líder en el sector de las cámaras *IP*, monitorización y centralización de soluciones de seguridad para *smart homes* y negocios de hoy y del mañana. Tiene su propia fábrica y cuenta con muchos años de experiencia en el sector de la vigilancia. Entre sus productos se encuentran diversos modelos de cámaras *IP*, y entre ellos el modelo utilizado en este estudio: *SP-006.*

Figura 6.2 Cámara IP Sricam SP-006

Esta cámara *IP* es capaz de capturar vídeo y retrasmitirlo en 720p y puede ser utilizada de manera inalámbrica, conectándola a través de *Wi-Fi*, o alámbrica, conectando el cable *ethernet* que incluye. Para este estudio, se ha optado por conectarla directamente por cable a uno de los puertos del entorno de pruebas creado.

Una vez conectado el dispositivo al entorno de pruebas, se ha procedido a la consumición del *streaming* de vídeo *RTSP (Real Time Streaming Protocol)* expuesto por esta cámara, en el caso del modelo *Sricam SP006*, la *URL (Uniform Resource Locator)* completa de conexión es *rtsp://<IP>/onvif1* utilizando el reproductor *VLC* (*https://www.videolan.org*), abriendo el *stream* desde el menú de medios de la aplicación:

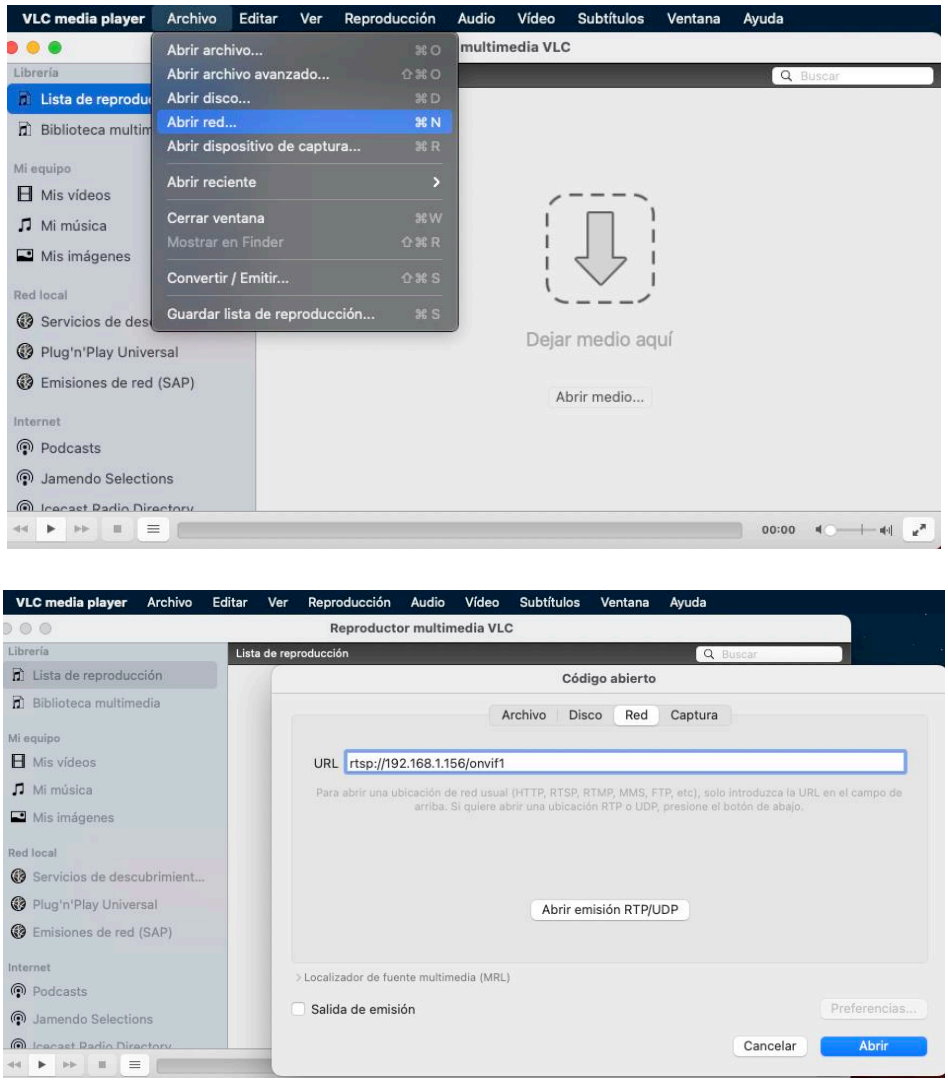


Figura 6.3 Consumición *streaming RTSP* utilizando el reproductor *VLC*

6.3. CAPTURA DE PAQUETES DE COMUNICACIÓN DE VARIOS DISPOSITIVOS

El último conjunto de datos fue recopilado durante varios días de funcionamiento de dichos dispositivos conectados al entorno de pruebas, siendo utilizados de manera normal y con el objetivo de obtener un conjunto de datos con paquetes de tráfico de red de diferentes dispositivos.

Los dispositivos conectados fueron los siguientes:

- ***Termostato Inteligente Tado*** *(https://www.tado.com/es-es/termostato-inteligente)*: se trata de un termostato inteligente que utiliza un puente para permitir la comunicación del termostato fijado a la pared y que controla la caldera con Internet y la aplicación que permite la gestión del termostato (temperatura, programación, etc.) a través de un *Smartphone* o la página web oficial del producto.

Figura 6.4 Termostato Inteligente Tado (Fuente: https://www.tado.com/es-es/termostato-inteligente)

- **Impresora multifunción HP LaserJet Pro MFP M28w** (*https://support.hp.com/lt-en/document/c05968109*) con conexión inalámbrica (*Wi-Fi*) que permite la impresión de documentos por red desde cualquier dispositivo conectado a la misma.

Figura 6.5 Impresora multifunción HP (Fuente: https://www.hp.com)

- **Iluminación Inteligente *Philips Hue*** *(https://www.philips-hue.com/es-es)*: producto de la marca *Philips* que permite controlar bombillas (encendido, apagado, brillo, etc.) a través de un puente que las controla, pudiendo ajustarlas a través de la aplicación oficial o su integración con *HomeKit*.

Figura 6.6 Bombillas Inteligentes Philips Hue (Fuente: https://www.philips-hue.com)

7. Análisis de tráfico

En este capítulo se recogen las diferentes técnicas de análisis aplicadas sobre los distintos conjuntos de datos de tráfico recopilados, así como el proceso para su ejecución y los resultados obtenidos.

7.1. INGENIERÍA INVERSA DEL PAYLOAD DE TRÁFICO BLE

La ingeniería inversa describe el proceso de, a partir de la observación de un producto o *software* terminado, descubrir su diseño, componentes y funcionamiento. Mediante el análisis de la captura de tráfico *BLE* y la observación de diferentes operaciones de pesado, se pudo extraer el esquema de los datos enviados por la báscula *BLE* al dispositivo controlador S*martphone* mediante dicha tecnología de comunicación:

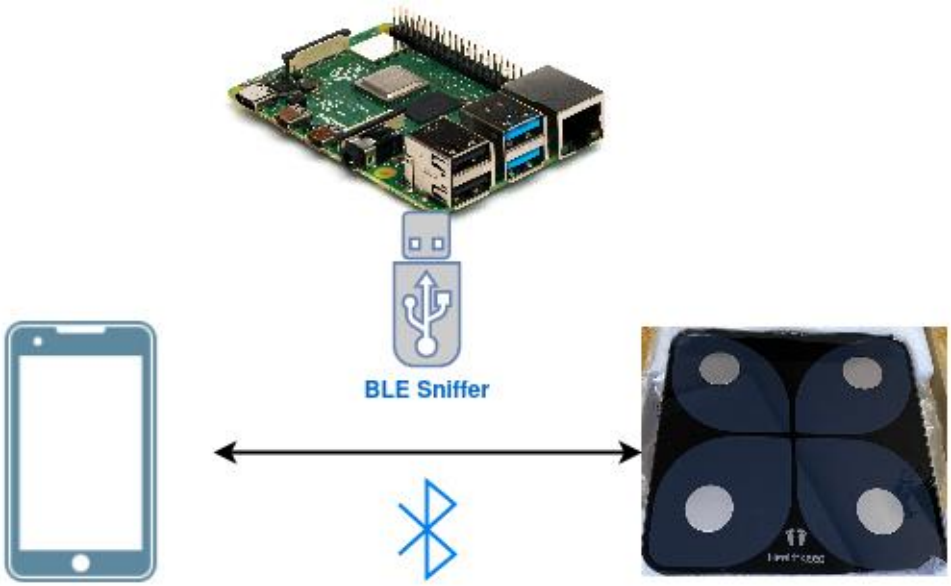


Figura 7.1 Diagrama captura tráfico de báscula *BLE*

Una vez capturado el intercambio de información, se procedió a su filtrado y a la disección de los paquetes relevantes utilizando *Wireshark*, analizador de tráfico de protocolos de red por excelencia. Permite mostrar a un nivel avanzado el intercambio de información de la red y es la herramienta por antonomasia utilizada por muchas empresas e instituciones.

Figura 7.2 Logo de *Wireshark* (Fuente: https://www.wireshark.org)

Algunas de sus funcionalidades son la inspección en profundidad de cientos de protocolos diferentes, la captura en vivo de tráfico, análisis mediante una interfaz gráfica o mediante línea de comandos, filtrado de paquetes y creación de reglas, entre otras. La disección de paquetes [25] consiste en analizar los datos de cada región del paquete encapsulados por cada protocolo, en el caso de la pila *TCP/IP:*

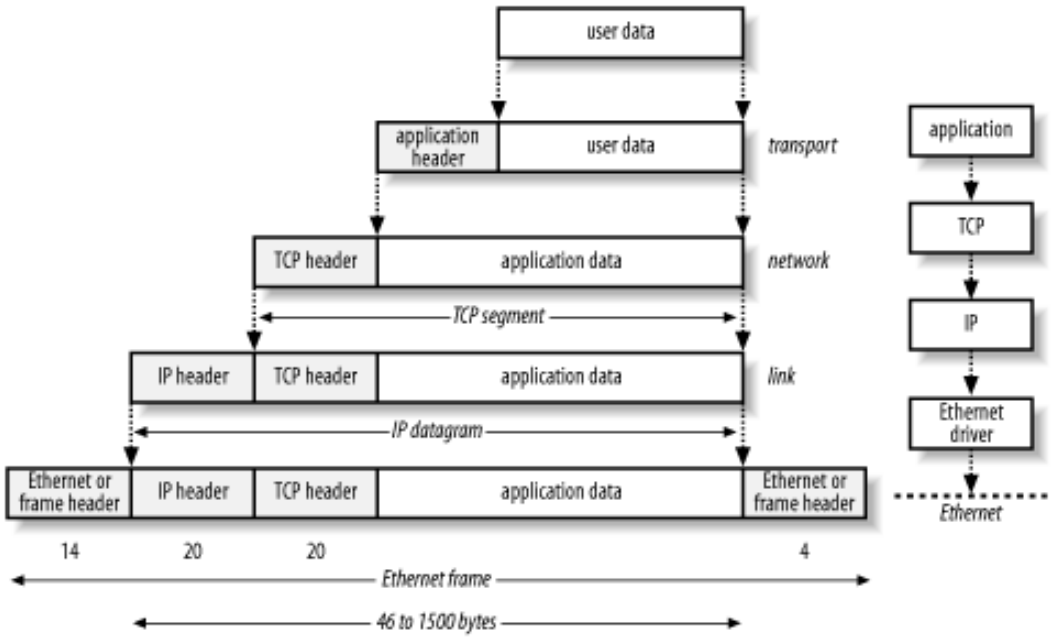


Figura 7.3 Diagrama encapsulación paquete TCP (Fuente: [25])

En cuanto al tráfico capturado, se trata de paquetes pertenecientes a *Bluetooth Low Energy (BLE)*. Esta tecnología de comunicación [26] utiliza 40 canales físicos en la banda de los 2.4GHz, cada uno de ellos separados por 2MHz. Además, define dos tipos de comunicación, trasmisión de datos y trasmisión de anuncios. Todos los dispositivos, sea cual sea su propósito específico, siempre comienzan en modo anuncio (*advertising*). En cuanto al formato de los paquetes del tráfico, la especificación define únicamente un tipo de formato para ambos tipos de transmisiones y que consta de cuatro componentes: preámbulo (1 octeto), dirección de acceso (4 octetos) *PDU* o *Protocol Data Unit* (2 a 257 octetos) y *CRC* o *Cyclic Redundancy Check* (3 octetos)

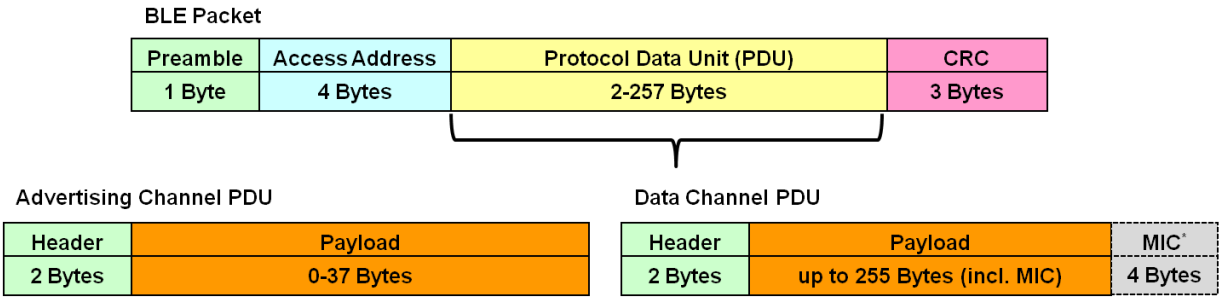


Figura 7.4 Formato paquetes Bluetooth Low Energy. Fuente:
https://microchipdeveloper.com/wireless:ble-link-layer-packet-types

Además, dentro de cada tipo de transmisiones (datos o anuncios), existen diferentes tipos de *PDU* existiendo los siguientes tipos de paquetes para las transmisiones de anuncio:

Tabla 7.1 Tipos de paquetes BLE de anuncio

Tipos de paquetes de anuncio	Tipo de anuncio
ADV_IND	*Connectable Undirected Advertising*
ADV_DIRECT_IND	*Connectable Directed Advertising*
ADV_NONCONN_IND	*Non-Connectable Undirected Advertising*
ADV_SCAN_IND	*Scannable Undirected Advertising*

Una vez identificados los paquetes de la comunicación *BLE* y mediante el filtrado de la captura, se pueden observar las diferentes capas del paquete que contiene los datos transmitidos por la báscula:

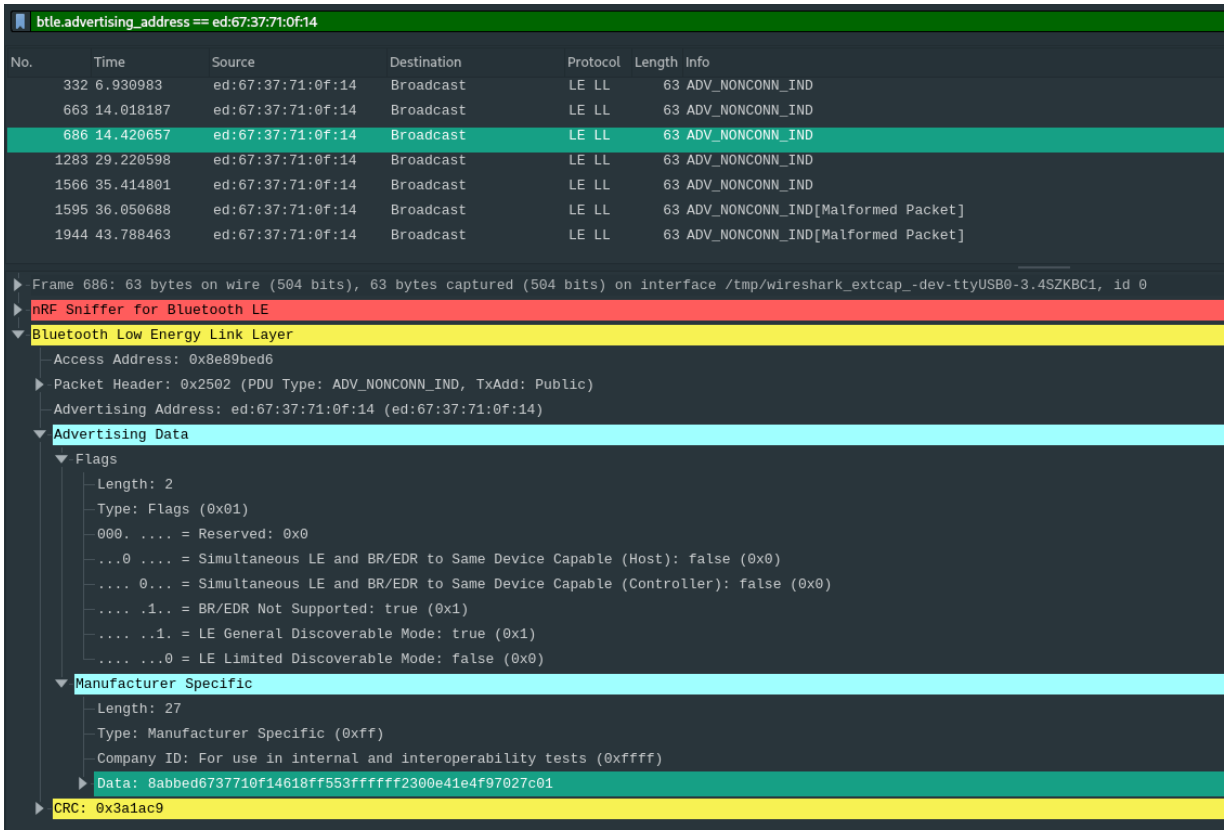


Figura 7.5 Filtrado de paquetes de captura de tráfico *BLE*

Como se puede observar en la captura, el campo de información indica que el tipo de *PDU* de los paquetes emitidos por la báscula son del tipo *ADV_NONCONN_IND*, lo que indica que se trata de un paquete que publicita información y que no permite la conexión ni el escaneo por otros dispositivos puesto que el dispositivo no está activamente escuchando y que se corresponde con la estructura de paquetes definidos para este tipo de paquetes (*broadcast*):

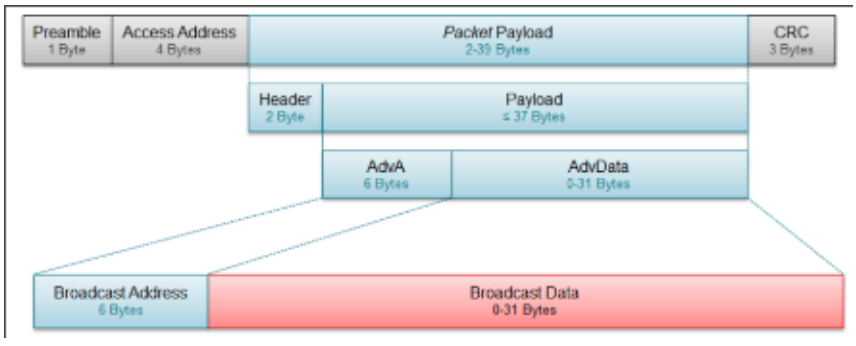


Figura 7.6 Formato de paquete de *broadcast BLE* (Fuente: https://www.ti.com/tool/TIDC-BLUETOOTH-LOW-ENERGY-BEACONS)

Una vez analizadas las partes que componen los paquetes, se procedió a observar el campo de datos incluido en el paquete en forma de una cadena de datos representada en hexadecimal:

```
aabbed4737710f14618ff553ffffff2300641e4fb7027c01
```

Tras la captura de diferentes mediciones (con diferentes pesos), se lograron identificar varios campos de dicha cadena de datos, para el ejemplo anterior:

Tabla 7.2 Identificación de campos de datos presentes en paquete de tráfico *BLE*

Constante	MAC	Marca temporal	Constante	Desconocido	Peso	Desconocido
aabb	ed4737710f14	618ff553	ffffff	2300	641e	4fb7027c01

Convirtiendo los valores de hexadecimal a decimal y teniendo en cuenta que el formato de los bytes del campo peso utiliza *little endian* (el *byte* menos significativo se ubica primero, en este caso *641e* se convierte como *1e64*):

Tabla 7.3 Conversión de campos de datos identificados en paquete de tráfico *BLE*

Constante	MAC	Marca temporal	Constante	Desconocido	Peso	Desconocido
aabb	ed4737710f14	1636824403	ffffff	2300	77.80	4fb7027c01

Gracias a la observación y al análisis en profundidad de los paquetes transmitidos por la báscula, ha sido posible extraer información de su uso mediante la ingeniería inversa de dicha comunicación, demostrando la facilidad para extraer información relevante del dispositivo sin la necesidad de intervención directa con el mismo.

7.2. RECONSTRUCCIÓN DE STREAMING RTP

El análisis *DPI* o *Deep Packet Inspection*, consiste en la inspección a bajo nivel de los paquetes o datagramas de información intercambiados por los dispositivos conectados [27]. La observación a este nivel no solo incluye las cabeceras de los datagramas, sino que también incluye el análisis del *payload* de los paquetes, es decir, los datos transmitidos. *DPI* es utilizado con numerosos fines, siendo los más importantes la identificación de posibles fugas de datos no encriptados, detección de programas malignos o el bloqueo de tráfico abusivo o mal intencionado.

Para llevar a cabo este análisis, se utilizó la captura de paquetes de la cámara *IP*. Dicha captura representa el intercambio de paquetes durante la reproducción del *streaming* de vídeo expuesto por la cámara a través del protocolo *RTSP*. *RTSP* [28] es un protocolo de nivel de aplicación que se utiliza para la transmisión en tiempo real de datos como audio y vídeo y es utilizado por cámaras de seguridad y vídeo vigilancia como la que ha servido de objeto de este estudio:

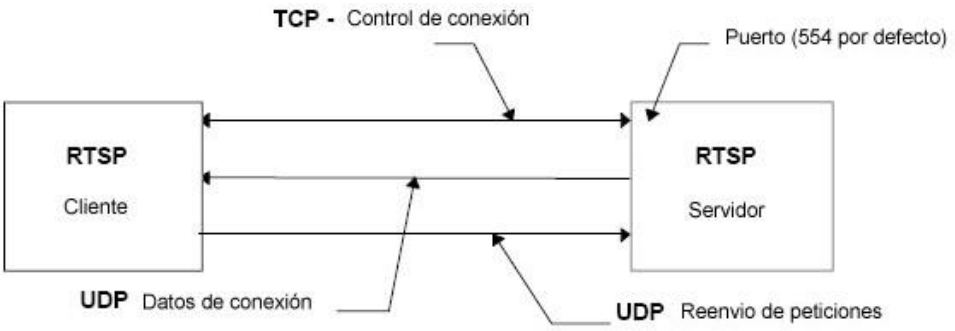


Figura 7.7 Diagrama de funcionamiento de protocolo *RTSP* (Fuente: https://es.wikipedia.org/wiki/Protocolo_de_transmisi%C3%B3n_en_tiempo_real)

Para llevar a cabo el análisis *DPI* se ha utilizado *Wireshark*, que, además de las funcionalidades previamente listadas, ofrece la posibilidad de instalar *scripts* escritos en el lenguaje de programación *Lua* y almacenarlos en forma de *plugins*. Una vez cargada la captura de tráfico en *Wireshark*, se puede observar el intercambio de paquetes entre la cámara *IP* y el dispositivo que consume el *feed* de vídeo:

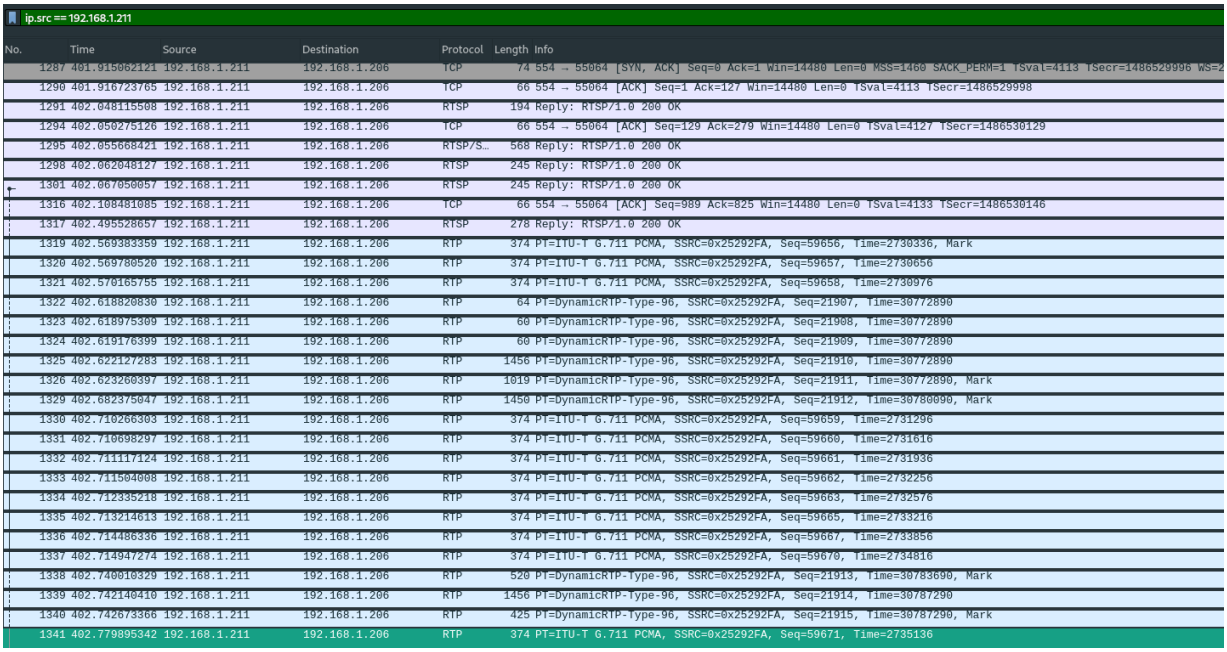


Figura 7.8 Filtrado de paquetes de tráfico *RTSP*

Además, *Wireshark* permite el análisis especifico de los *streams RTP*, permitiendo identificar su duración, *jitter* o fluctuación del retardo y estadísticas sobre el número de paquetes esperados y recibidos:

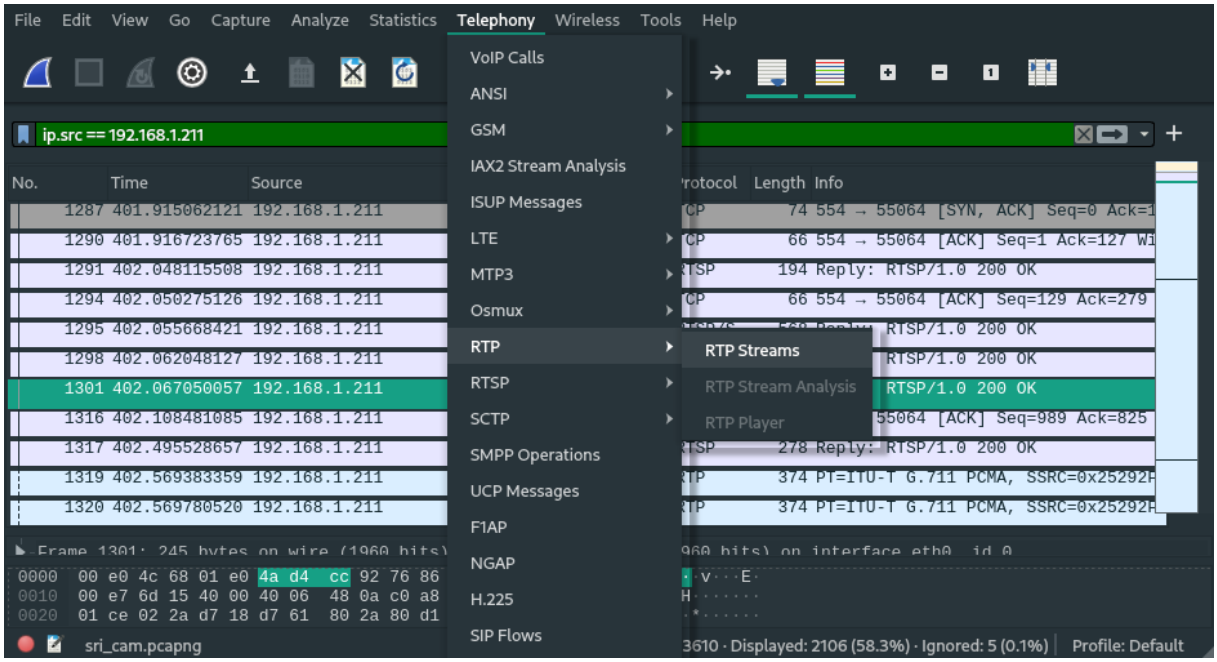


Figura 7.9 Función *RTP streams* de *Wireshark*

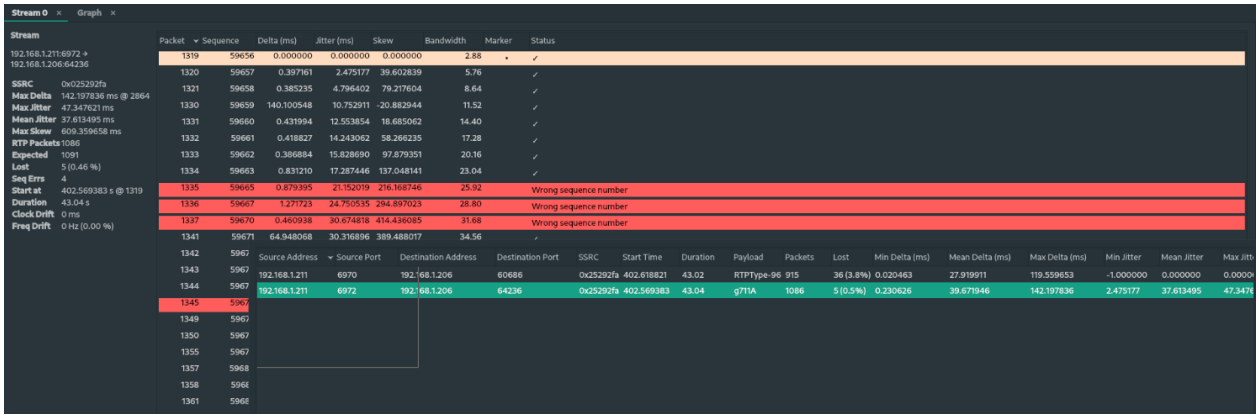


Figura 7.10 Análisis de *stream RTP* con *Wireshark*

Por último y con el fin de explotar al máximo la versatilidad de *Wireshark* como herramienta de análisis, se procedió a reconstruir el *streaming* a partir del contenido de los paquetes *RTP* (*payload*) intercambiados. Para ello se instaló un *plugin* escrito en *LUA* para la extracción de dicho *streaming* [29]. En primer lugar, se procedió a la instalación del *script LUA* siguiendo los siguientes pasos:

1. Desacargar y emplazar el *script* disponible en https://knowledgebase-iframe.polycom.com/kb/viewContent.do?externalId=34917 en el directorio en el que *Wireshark* está instalado (en *Windows C:\Program Files\Wireshark)*.
2. Reiniciar *Wireshark* y verificar que el *plugin* está disponible en el menú de herramientas o *tools:*

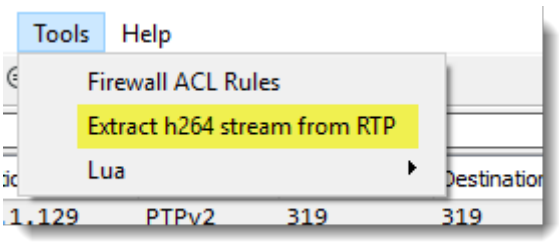


Figura 7.11 Instalación *plugin Lua* para extraer *streams* RTP en *Wireshark*

Este *script* extrae *streamings* codificados en formato *H.264* (códec de vídeo utilizado para grabar y distribuir señales de video y audio) de paquetes *RTP*, concretamente del *payload* de dichos paquetes. Para que *Wireshark* reconozca de manera correcta los paquetes RTP con la codificación *H.264* se indicó el *payload type* de dicha codificación en el panel de preferencias:

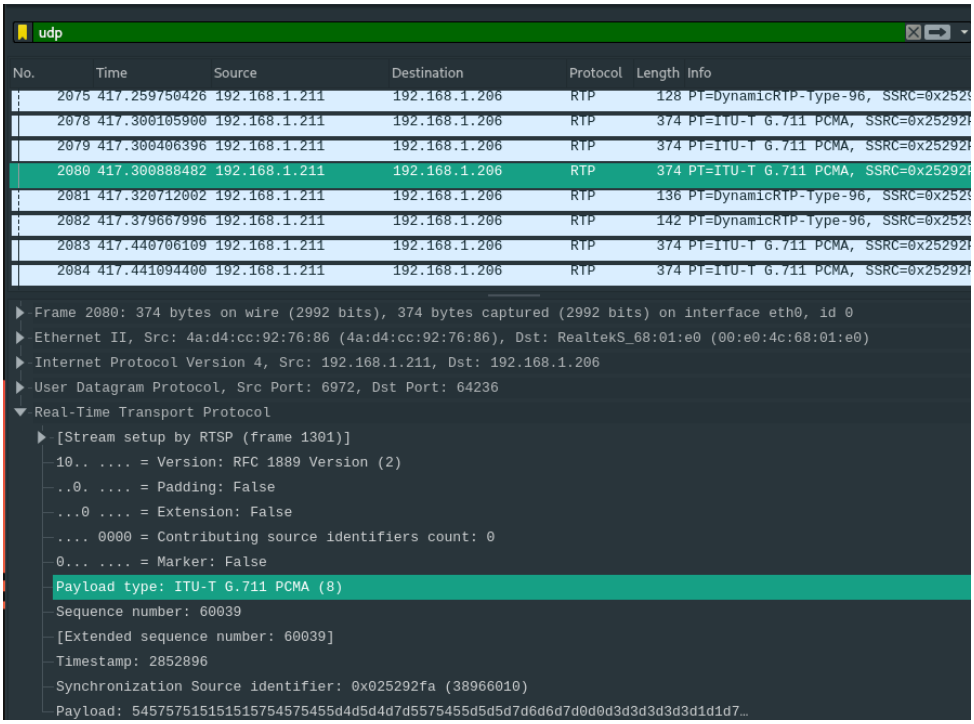


Figura 7.12 Identificación *payload type* de *RTP stream*

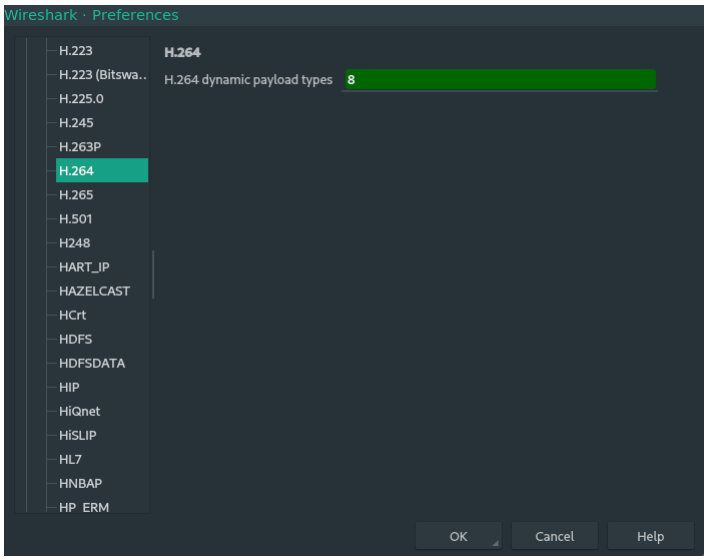


Figura 7.13 Configuración de tipo de *payload* dinámico de *H.264* en *Wireshark*

Seguidamente, se carga la captura de tráfico, filtrando únicamente el *streaming H264* que se quiere extraer, filtrando dichos paquetes e incluyéndolos en un nuevo archivo que almacene únicamente el *streaming*:

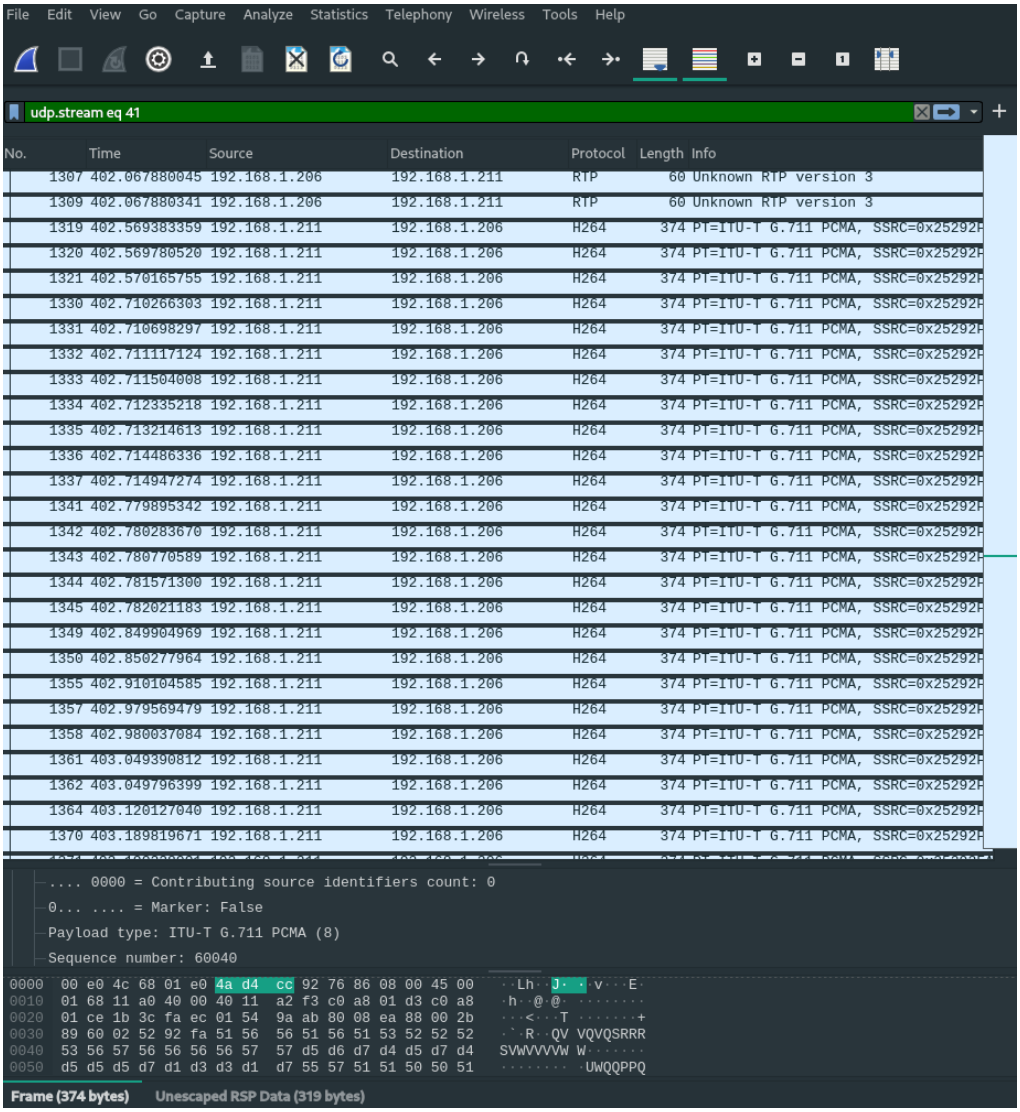


Figura 7.14 Filtrado de tráfico *RTP* en *Wireshark*

Tras el filtrado, se ejecuta el *plugin* previamente instalado en *Wireshark*:

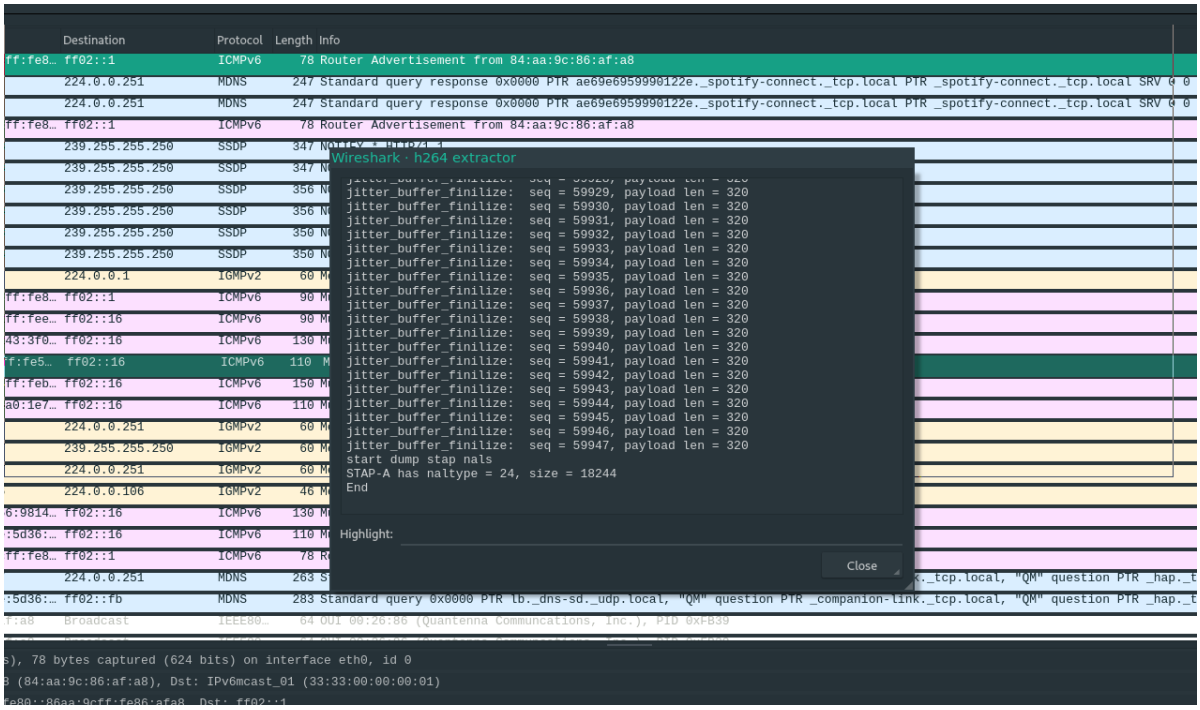


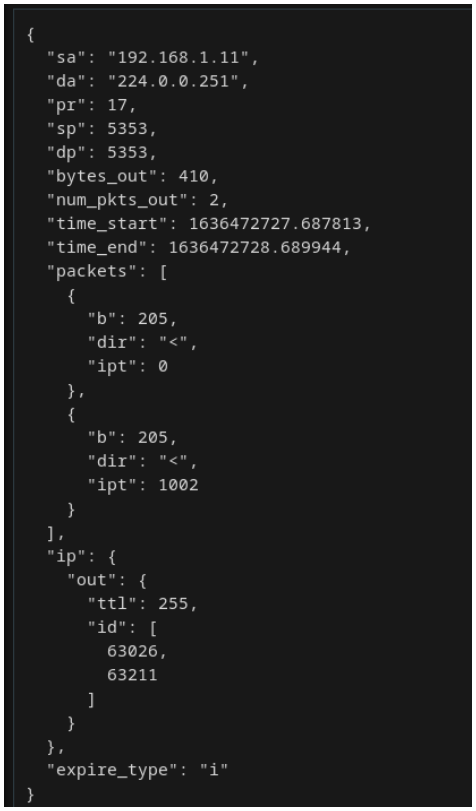
Figura 7.15 Ejecución del *plugin* de extracción de *streaming RTP*

La ejecución del *plugin* produce un archivo *dump.264* en el mismo directorio de la captura y que contiene el *stream* de vídeo de la cámara, que puede ser visualizado utilizando un reproductor compatible como *VLC*:

Figura 7.16 Reproducción de *streaming RTP* extraído

7.3. USO DE FLUJOS PARA FILTRADO DE PAQUETES

Utilizando la misma fuente de datos, es decir, la captura de paquetes de la cámara *IP* durante la consumición del stream RTP, se ha realizado un análisis del tráfico generado por dicha cámara durante su funcionamiento, convirtiendo dicha captura de paquetes en flujos. En resumen, un flujo es una serie de paquetes que comparten el mismo protocolo, la misma dirección *IP* y puertos fuente (*source*) y destino (*destination*) [30], también denominando flujo a un conjunto de flujos individuales, resumiendo los detalles de las conexiones en la red, sin incluir los datos intercambiados en dicha comunicación. Desde la liberación de *NetFlow* por *Cisco* como una característica para administradores de red en sus equipos, esta característica ha sufrido revisiones y otros protocolos similares como *sFlow* han competido con él, siendo la última contribución el estándar *IPFIX* (*IP Flow Information eXport*). Para realizar este análisis se ha utilizado la herramienta *Joy* [17], creada por *Cisco*. Se trata de un paquete software licenciado como *BSD* y que sirve para la extracción de características de datos de capturas de tráfico de red (*pcap*), utilizando un modelo orientado a flujos similar al de *IPFIX* o *Netflow*, representando estas características en formato *JSON (JavaScript Object Notatotion)*, un formato fácilmente consumible por otras herramientas de análisis. Ejemplo de flujo extraído del tráfico utilizando la herramienta:



```
{
  "sa": "192.168.1.11",
  "da": "224.0.0.251",
  "pr": 17,
  "sp": 5353,
  "dp": 5353,
  "bytes_out": 410,
  "num_pkts_out": 2,
  "time_start": 1636472727.687813,
  "time_end": 1636472728.689944,
  "packets": [
    {
      "b": 205,
      "dir": "<",
      "ipt": 0
    },
    {
      "b": 205,
      "dir": "<",
      "ipt": 1002
    }
  ],
  "ip": {
    "out": {
      "ttl": 255,
      "id": [
        63026,
        63211
      ]
    }
  },
  "expire_type": "i"
}
```


Figura 7.17 Ejemplo de flujo extraído con *Cisco Joy*

Como se puede observar el objeto *JSON* contiene diferentes campos:

JSON element	Data element	Notes
sa	Source Address	Internet Protocol
da	Destination Address	
pr	Protocol Number	
sp	Source Port	TCP or UDP
dp	Destination Port	
bytes_out	Number of bytes sa→da in TCP, UDP, ICMP, or IP Data fields	Any Protocol
num_pkts_out	Number of packets sa→da	
time_start	Start time, seconds since epoch	
time_end	End time, seconds since epoch	
packets	Array of packet lengths, directions, and times	
bytes_in	Number of bytes sa←da in TCP, UDP, ICMP, or IP Data fields	Bidirectional flows only
num_pkts_in	Number of packets sa←da	

Figura 7.18 Elementos objeto *JSON* que representan flujo (Fuente: https://github.com/cisco/joy)

Además, *Joy* incluye herramientas de análisis aplicables a estos ficheros de datos estructurados que produce, como es el caso de *sleuth* que, con los datos ya estructurados en objetos por cada flujo,

permite ejecutar cómputos y consultas en un formato parecido a la sintaxis *SQL (Structured Query Language)*.

Para instalar *Joy* se deben ejecutar los siguientes comandos:

```
$: sudo apt-get install build-essential libssl-dev libpcap-dev libcurl4-openssl-dev
$: git clone https://github.com/cisco/joy.git
$: cd joy
$: ./configure
$: make clean;make
```

Una vez ejecutados estos comandos, el binario *joy* estará disponible en la carpeta *bin* del repositorio y el binario *sleuth* en la carpeta base del repositorio. Un ejemplo que puede ser utilizado para comprobar el número de flujos según el tipo de protocolo (*pr*) utilizando *sleuth* sobre los flujos extraídos previamente con *joy*:

```
user@server:~/joy$ ./sleuth sri_cam.out $@  --select "pr" --groupby "pr" --dist
read 428 lines
{"pr": 6, "count": 1, "total": 1}
{"pr": 17, "count": 43, "total": 43}
{"pr": 2, "count": 16, "total": 16}
{"pr": 58, "count": 15, "total": 15}
{"pr": 1, "count": 3, "total": 3}
```

Figura 7.19 Ejemplo de consulta de flujos utilizando *sleuth* para comprobar distribución de flujos por protocolo

Lo que significa una distribución de flujos en la captura del tráfico producido por la cámara *IP* y tras consultar la numeración de protocolos [31], según la tabla siguiente:

Tabla 7.4 Distribución de flujos según protocolo del tráfico de la cámara IP

PR	Protocolo	Flujos
17	UDP	43
2	IGMP	16
58	IPv6-ICMP	15
1	ICMP	3
6	TCP	1

Otro ejemplo es comprobar las peticiones *DNS (Domain Name System)*, que como se puede observar, algunas de ellas hacen referencia a dominios en China:

```
user@server:~/joy$ ./sleuth sri_cam.out $@  --select "dns[{qn}]" --groupby "dns[{qn}]" --dist
read 428 lines
{"count": 2, "total": 2, "dns": [{"qn": "p2p4.videoipcamera.com"}]}
{"count": 2, "total": 2, "dns": [{"qn": "p2p6.videoipcamera.com"}]}
{"count": 2, "total": 2, "dns": [{"qn": "p2p2.videoipcamera.com"}]}
{"count": 2, "total": 2, "dns": [{"qn": "p2p3.videoipcamera.cn"}]}
{"count": 2, "total": 2, "dns": [{"qn": "p2p5.videoipcamera.cn"}]}
{"count": 2, "total": 2, "dns": [{"qn": "p2p1.videoipcamera.cn"}]}
```

Figura 7.20 Ejemplo de consulta de flujos utilizando *sleuth* para comprobar consultas *DNS*

Como se ha visto, *Joy* puede ser utilizado para la exploración de conjuntos agregados de datos, especialmente datos relevantes en cuanto a seguridad y amenazas.

7.4. ENRIQUECIMIENTO Y VISUALIZACIÓN DE FLUJOS UTILIZANDO ELK

Con el fin de enriquecer y poder visualizar de una forma más clara y dinámica el tráfico presente en la captura de varios dispositivos durante varios días, se ha desplegado la pila de servicios *ELK*. *ELK* [32] son las siglas que representan tres proyectos de código abierto, *Elasticsearch*, *Logstash* y *Kibana*:

- *Elasticsearch* es un motor de búsqueda y analítica, semejante a una base de datos.
- *Logstash*, *pipeline* para el procesamiento y transformación de datos que luego envía a *Elasticsearch*.
- *Kibana*, que permite la visualización y consulta de los datos ingerido en *Elasticsearch*.

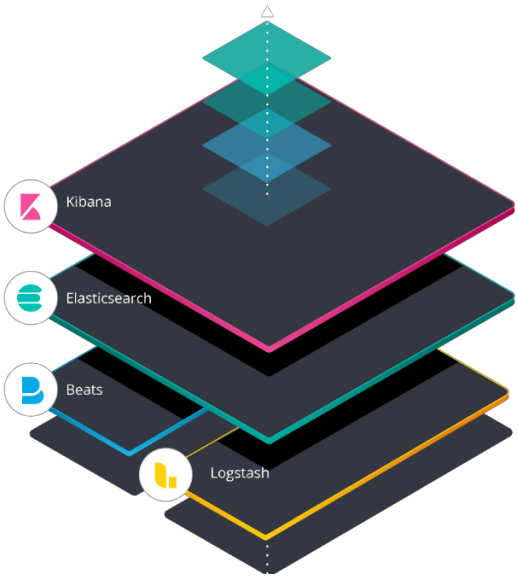


Figura 7.21 Diagrama de componentes pila ELK (Fuente: https://www.elastic.co/es/what-is/elk-stack)

Una vez instalados y configurados estos servicios (en el *Anexo II: Instalación y configuración de ELK y herramientas de análisis* se puede encontrar más información al respecto de la instalación y configuración) se procesó el archivo de la captura utilizando la herramienta *Joy*. Una vez procesados los flujos, el archivo resultante fue ingerido en la pila *ELK* utilizando *logstash* y llevando a cabo una serie de transformaciones con el fin de enriquecer los datos de los flujos. En general, los *pipelines* o configuraciones de procesado de *logstash*, constan de tres partes, el *input* u origen de entrada de los datos, un *filter* o filtro en el que se encuentran todas las transformaciones que se quieran realizar sobre los datos de entrada, y un *output* o salida de los datos resultantes.

En este caso, se ha configurado el *input* como un directorio del sistema en el que se instaló la pila *ELK* y utilizando el códec de *JSON*, pues este es el formato de salida de la ejecución de *Joy*:

```
input {
  file{
    path => "/home/user/flows/*.json"
    start_position => "beginning"
    sincedb_path => "/dev/null"
    codec => json
  }
}
```

Figura 7.22 Configuración *input logstash*

En cuanto al filtrado y transformación, se han incluido los siguientes *plugins*:

- Traducción, a través de un filtro *translate*, del campo *pr* producido por *Joy* a un nuevo campo indicado como *destination* (*[network][protocol]*) que contiene el nombre del protocolo extraído del diccionario (*dictionary)* que contiene como índice el número de protocolo y como valor el protocolo al que hace referencia dicho número:

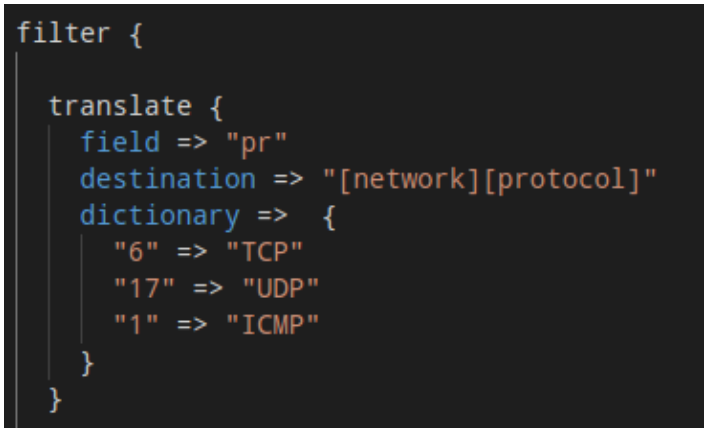


Figura 7.23 Configuración *filter logstash* para traducción de protocolos

- Traducción de la *IP* asignada a los dispositivos conectados durante la captura a un nombre identificable, utilizando el campo *sa* producido por *Joy* como fuente y un diccionario con las *IPs* de los dispositivos y su nombre correspondiente, siendo el *destination* un nuevo campo llamado *source_name*:

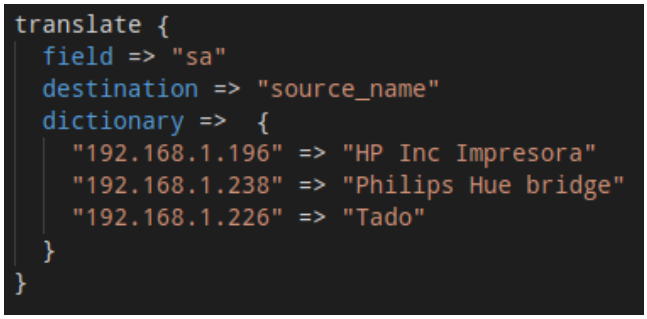


Figura 7.24 Configuración *filter logstash* para traducción de *IP* fuente

- Traducción del campo *time_start*, utilizando el filtro *date,* que corresponde al tiempo de inicio de los flujos, a un nuevo campo especificado como *target, eventTime*, que será utilizado como

campo de referencia para la consulta en *Kibana* y utilizando *UNIX* como el formato de entrada del filtro:

```
date {
  match => [ "time_start","UNIX" ]
  target => "eventTime"
}
```

Figura 7.25 Configuración *filter logstash* para traducción del campo *time_start* del flujo

- Resolución inversa (*reverse*) de dominios gracias al *plugin dns* de los campos *IP* fuente (*sa*) y destino (*da*) de los flujos, utilizando para ello como *nameserver* el de Google (8.8.8.8 y 8.8.4.4) y almacenando la información en campos creados a tal efecto y cuyo valor inicial es el mismo que *da* y *sa*:

```
mutate {
  add_field => { "dst_ip" => "%{da}" }
  add_field => { "src_ip" => "%{sa}" }
}

dns {
  nameserver => {
    address => ["8.8.8.8", "8.8.4.4"]
  }
  reverse => [ "dst_ip", "src_ip" ]
  action => "replace"
}
```

Figura 7.26 Configuración *filter logstash* para resolución inversa de dominios

- Enriquecimiento con datos de geolocalización gracias al *plugin geoip* [33] que se encarga de buscar en una base de datos la *IP* fuente (*sa*) y destino (*da*) de los flujos y almacenar los datos en un nuevo campo *geo*:

```
geoip {
  default_database_type => "City"
  source => "da"
  target => "geo-da"
  tag_on_failure => ["geoip-city-failed"]
}
geoip {
  default_database_type => "ASN"
  source => "da"
  target => "geo-da"
  tag_on_failure => ["geoip-asn-failed"]
}

geoip {
  default_database_type => "City"
  source => "sa"
  target => "geo-sa"
  tag_on_failure => ["geoip-city-failed"]
}
geoip {
  default_database_type => "ASN"
  source => "sa"
  target => "geo-sa"
  tag_on_failure => ["geoip-asn-failed"]
}
```

Figura 7.27 Configuración *filter logstash* para el enriquecimiento con datos de geolocalización

Por último, la salida de *logstash* se envía a *elasticsearch*, utilizando un *template* predefinido y configurado en *elasticsearch*:

```
output {
  elasticsearch {
    index => "joy"
    template_name => "joy_template"
  }
}
```

Figura 7.28 Configuración *output logstash*

Una vez procesados los registros por *logstash* y enviados a *elasticsearch*, es posible consultarlos a través de *Kibana* tras crear un patrón, especificando el campo que servirá para construir la línea temporal de los registros (*eventTime*):

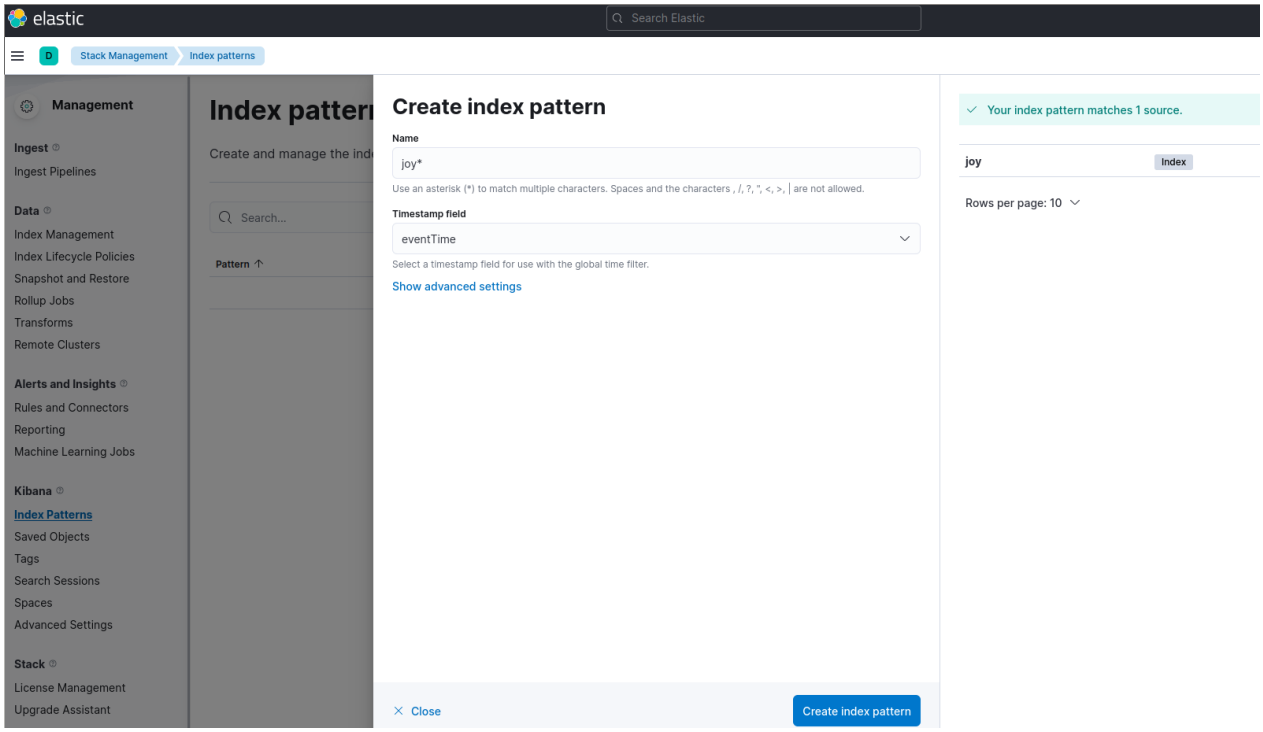


Figura 7.29 Creación de *index pattern* en *Kibana*

Una vez creado el patrón, desde la sección *Discover* de *Kibana* se pueden consultar los registros correspondientes a los diferentes flujos obtenidos con *Joy*, así como la línea temporal de los mismos:

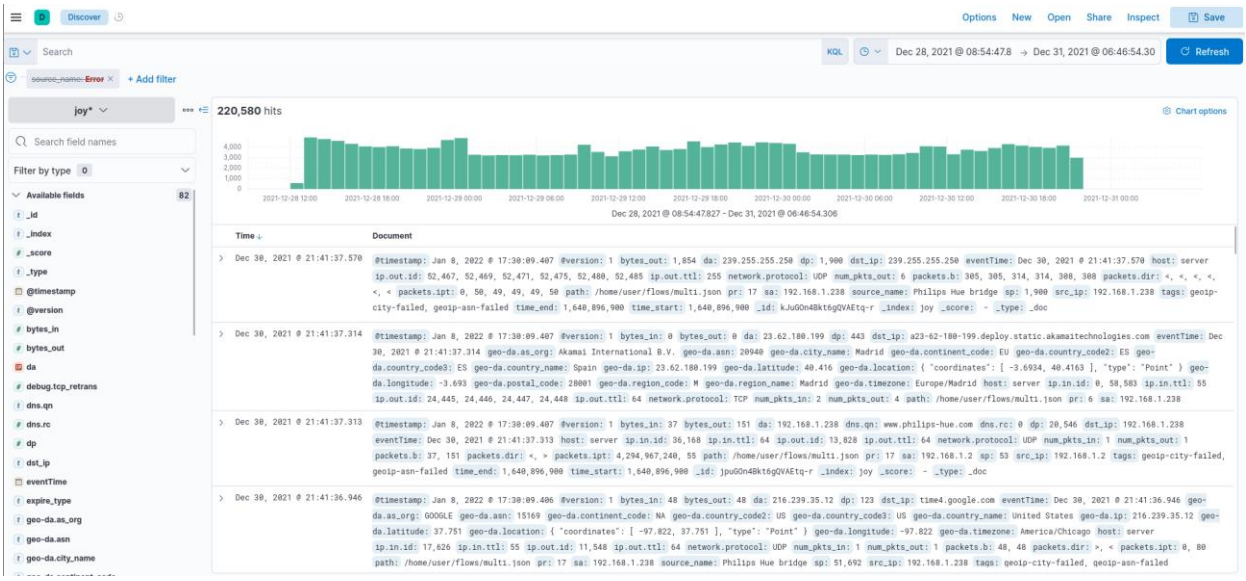


Figura 7.30 Visualización de datos en *Kibana*

Gracias a *Kibana*, no solo se pueden realizar consultas y operaciones de filtrado sobre los datos ingeridos, sino que también se pueden crear visualizaciones de estos. Con el fin de entender mejor

la distribución del tráfico de los diferentes dispositivos capturados, se ha optado por crear un diagrama *sankey*, aprovechando el soporte de *Kibana* para la librería *Vega* [34] que permite programar y desarrollar visualizaciones utilizando *JSON*:

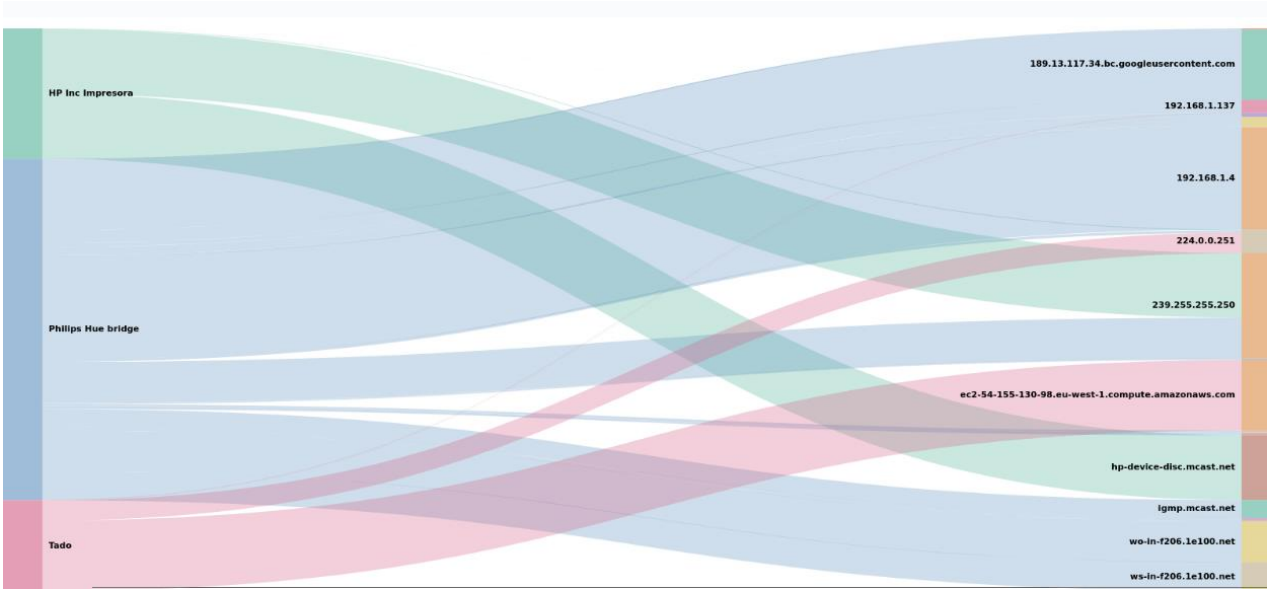


Figura 7.31 Diagrama *Sankey* creado en *Kibana* para visualizar la distribución del tráfico

Como se puede observar, la mayoría de los flujos (60,6%) han sido producidos por el puente *Philips Hue*, comunicándose no solo con dispositivos internos de la red doméstica (servidor *Home Assistant*, dispositivos *iPhone*), sino que también realiza conexiones con servicios fuera de la red doméstica. El siguiente dispositivo que genera más flujos es la impresora multifunción HP (23,2%), tráfico localizado únicamente en la red doméstica y de tipo *multicast*. Por último, el dispositivo *Tado* ha generado un 16,2% de los flujos cuyo destino ha sido tráfico *multicast* y una instancia *EC2* de *Amazon Web Services*. Todos los dispositivos tienen en común tráfico a direcciones *IP* del rango 224.0.0.0 a 239.255.255.255, reservadas para el tráfico *multicast* [35] cuyo objetivo principal es anunciar su presencia al resto de dispositivos de la red, a través de diferentes protocolos, como *UPnP (Universal Plug and Play)* o *SSDP (Simple Service Discovery Protocol)* y facilitar así su uso y configuración por otros dispositivos de la red.

En cuanto al análisis de flujos según protocolo utilizado para la totalidad de los flujos y utilizando otra visualización de tipo tarta, el 48,41% de los flujos correspondieron a *UDP (User Datagram Protocol)*, mientras que el 43.45% correspondieron a *TCP (Transmission Control Protocol)*, con el 8.14% restante siendo tráfico *ICMP (Internet Control Message Protocol)*:

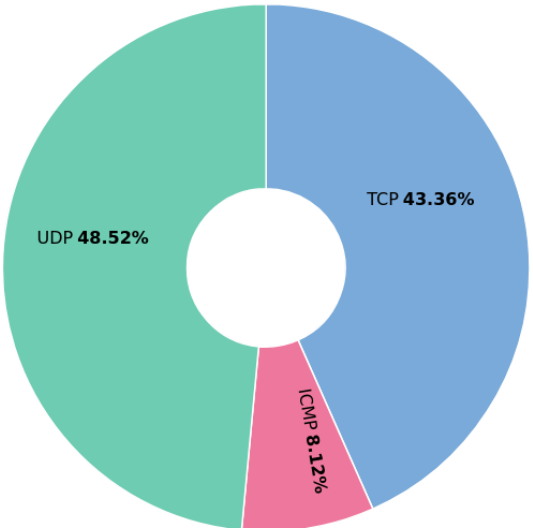


Figura 7.32 Gráfico *Donut* creado en *Kibana* para visualizar la distribución de protocolos

Por dispositivos, la distribución sigue de la manera siguiente:

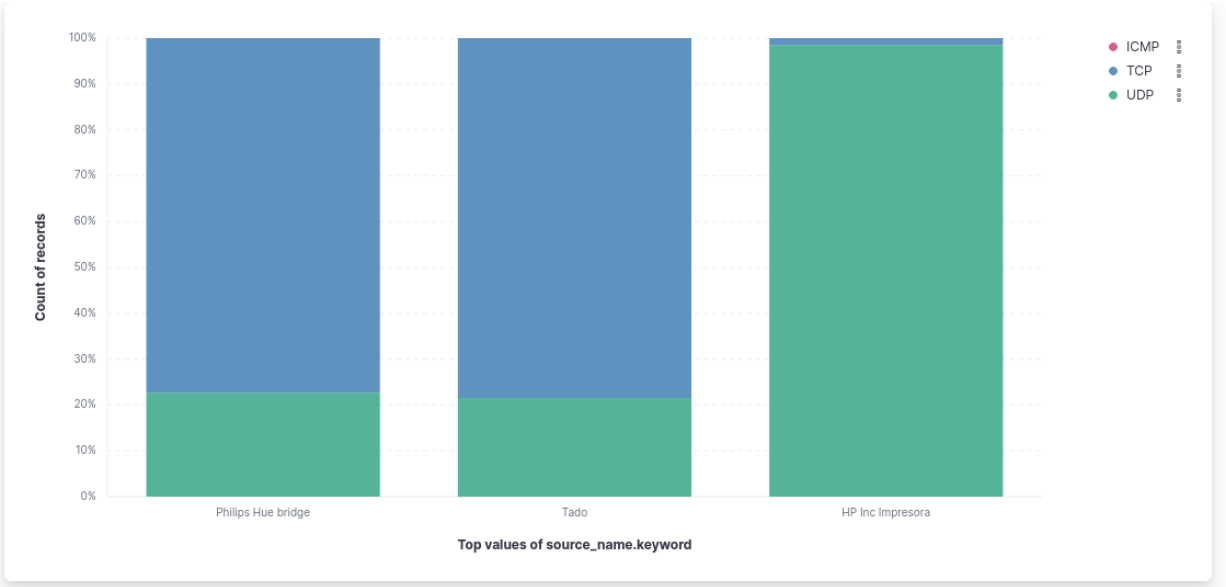


Figura 7.33 Gráfico de barras creado en *Kibana* para visualizar la distribución de protocolos por dispositivo

Comprobando los puertos de destino de cada uno de los dispositivos, se puede observar la siguiente distribución:

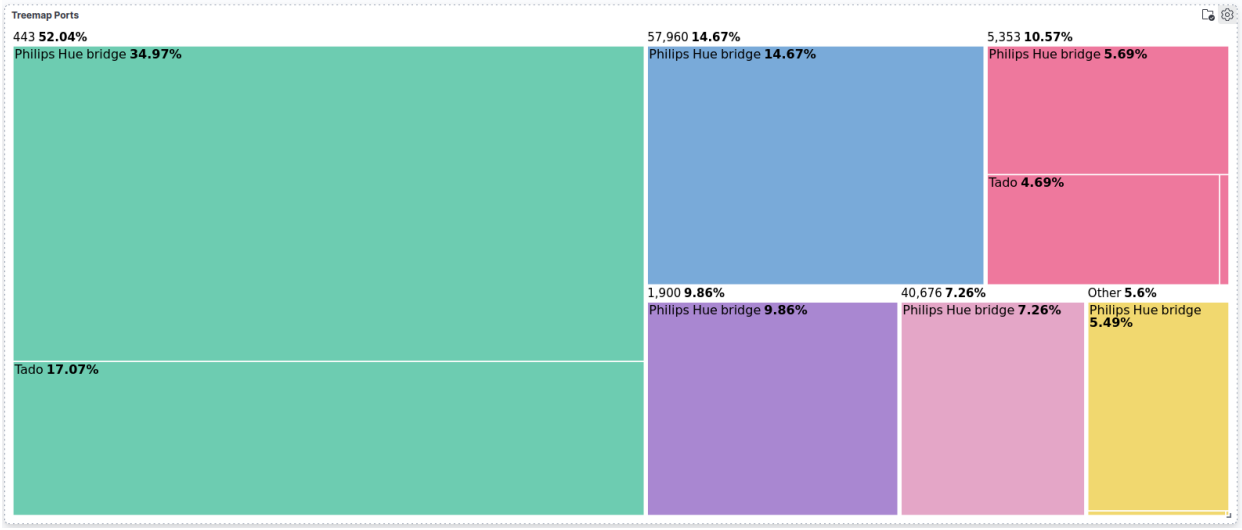


Figura 7.34 Diagrama *Treemap* creado en *Kibana* para visualizar la distribución de puertos destino por dispositivo

Muchos de los puertos destino utilizados por las conexiones pertenecen a puertos bien conocidos:

- Puerto 443, correspondiente a tráfico *HTTPS (Hypertext Transfer Protocol Secure)*.
- Puerto 1900, correspondiente a tráfico *SSDP* utilizado para el descubrimiento de servicios en red.
- Puerto 5353, correspondiente al protocolo m*DNS (multicast DNS)*.

Por último, con el fin de averiguar la geolocalización del tráfico destino de los flujos de cada uno de los dispositivos, se ha creado un mapa en *Kibana* para visualizar dichos datos:

- Las direcciones *IP* de destino correspondientes a los flujos extraídos por el tráfico generado por el dispositivo *Tado* se localizan, como puede comprobarse en la visualización de tipo mapa creada en *Kibana* a partir de los datos, en Irlanda, coincidiendo con la localización de la región *eu-west-1* de *AWS (Amazon Web Services)* donde se aloja la instancia *EC2 (Elastic Compute Cloud)* con la que ha realizado las comunicaciones fuera de la red doméstica:

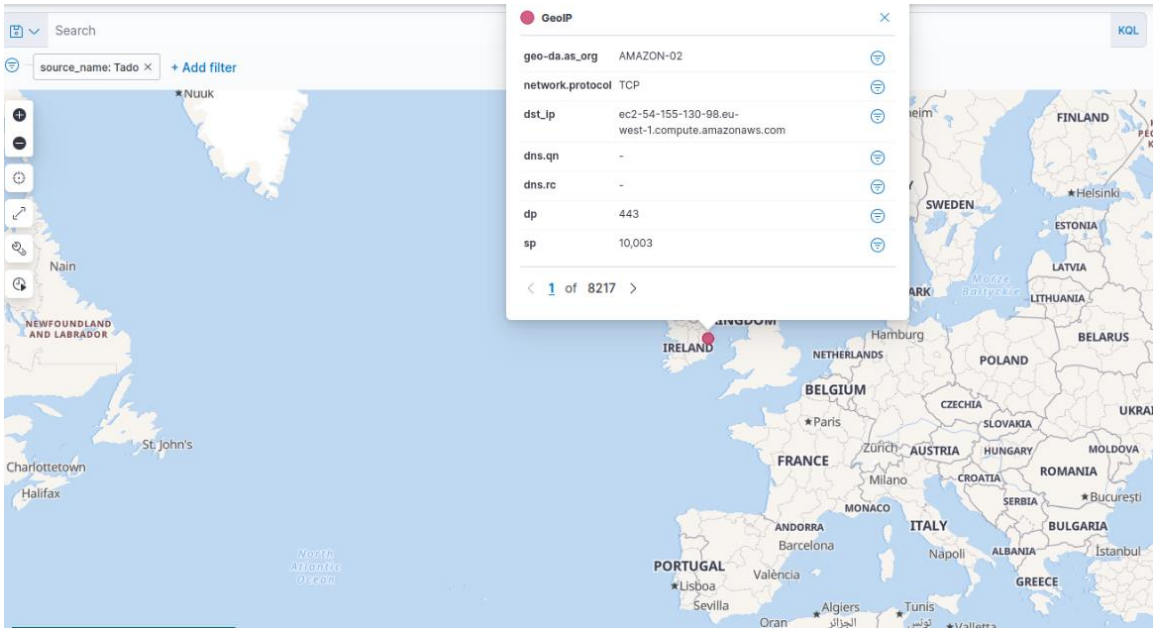


Figura 7.35 Mapa creado en *Kibana* para visualizar datos de geolocalización de flujos del dispositivo *Tado*

- Por otro lado, el dispositivo *Philips Hue* cuenta con más variedad de *IPs* de destino, localizándose estas no solo en Europa sino también en EE. UU siendo *Google* la organización al cargo de dichas *IPs*:

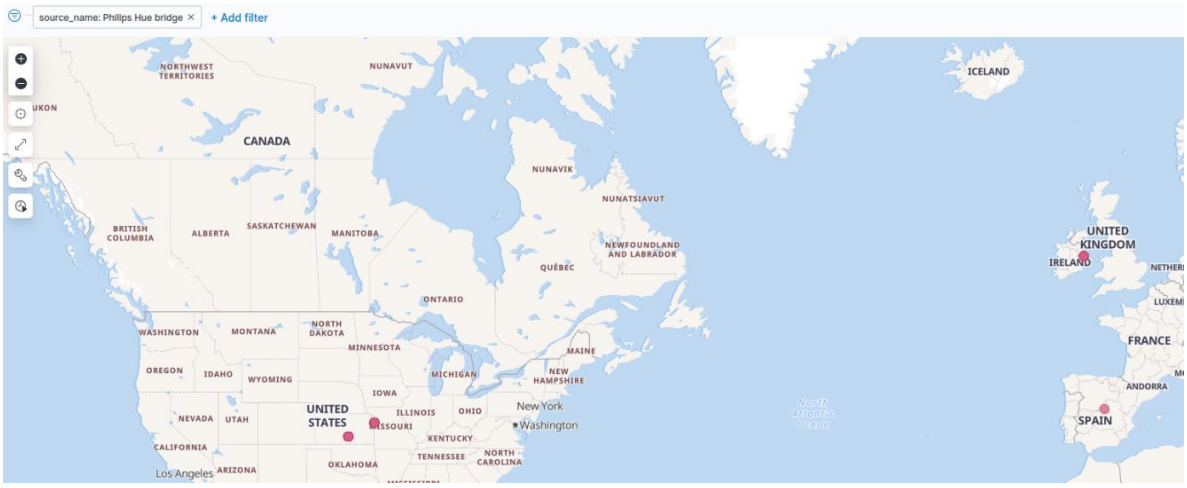


Figura 7.36 Mapa creado en *Kibana* para visualizar datos de geolocalización de flujos del dispositivo *Philips Hue*

- No se ha estudiado este tipo de visualización para la impresora HP multifunción, pues todo el tráfico que ha producido durante la captura, y como fue observado en el análisis de la anterior visualización, estaba destinado a la red interna.

8. Conclusiones

Este trabajo ha demostrado la diversidad de técnicas existentes para analizar tráfico de red, poniendo en práctica algunas de ellas y utilizando diferentes herramientas y un entorno de pruebas creado para facilitar la recopilación del tráfico de diferentes dispositivos *IoT*.

En primer lugar, y mediante técnicas avanzadas de ingeniería inversa, se ha demostrado que es posible comprender y explotar el comportamiento de muchos de los dispositivos inteligentes que están presentes en los hogares de muchas personas.

Por otro lado, la automatización del análisis de manera pasiva y mediante técnicas de simplificación, utilizando flujos en lugar del procesamiento en tiempo real de paquetes completos a bajo nivel, puede ayudar a identificar diferentes riesgos y problemas, detectando dispositivos comprometidos o que su patrón de comportamiento no esté dentro de los objetivos de seguridad de la red doméstica. Si bien los dispositivos *IoT* pueden ser de utilidad en diversos escenarios, no implementan un esquema apropiado de seguridad y pueden ser atacados exponiendo información que podría comprometer la privacidad de los usuarios, pues como se ha mencionado, la seguridad no es una prioridad en su diseño, siendo patente la búsqueda de un balance entre facilidad de uso y la asunción de implementaciones poco seguras.

Además, gracias al uso de herramientas de código abierto, es posible la visualización de los flujos obtenidos a partir de otras herramientas que ayudan a comprender, de manera agregada y simple, la distribución del tráfico de red de los dispositivos, el tipo de conexiones según su destino (internas o externas) y la obtención de datos sobre los destinos de dichas conexiones, como su nombre o localización.

En cuanto a los siguientes pasos que se pueden dar para continuar la investigación iniciada con este trabajo, uno de ellos podría ser la captura y análisis de otros dispositivos para entender su funcionamiento y problemas de seguridad, la aplicación de técnicas de *Machine Learning* o aprendizaje automático para crear modelos del tráfico de los dispositivos de manera automática a partir de las características específicas del tráfico de estos dispositivos, así como técnicas de *fingerprinting* para crear una base de datos con las huellas producidas por la comunicación encriptada de estos dispositivos. Otro tema de continuación de este trabajo está relacionado con la

automatización de la detección de conexiones poco seguras o inesperadas de estos dispositivos, apoyándose en perfiles *MUD.*

Por último, destacar el aprendizaje personal que ha supuesto este proyecto, abarcando conocimientos sobre técnicas avanzadas de análisis de tráfico de red utilizando herramientas disectoras de paquetes como *Wireshark* y otras basadas en flujos como *Joy*, la puesta en práctica de técnicas de ingeniería inversa y el uso de la pila *ELK* para el enriquecimiento y visualización del tráfico de red.

9. Referencias

[1] M. R. Shahid, G. Blanc, Z. Zhang and H. Debar, "IoT Devices Recognition Through Network Traffic Analysis," 2018 IEEE International Conference on Big Data (Big Data), 2018, pp. 5187-5192, doi: 10.1109/BigData.2018.8622243.

[2] Lueth, K. L. (2021, 8 noviembre). State of the IoT 2020: 12 billion IoT connections, surpassing non-IoT for the first time. IoT Analytics. Recuperado 21 de marzo de 2021, de https://iot-analytics.com/state-of-the-iot-2020-12-billion-iot-connections-surpassing-non-iot-for-the-first-time

[3] Dange, Smita. (2019). IoT Botnet: The Largest Threat to the IoT Network. 10.1007/978-981-15-0132-6.

[4] Stratosphere. (2015). Stratosphere Laboratory Datasets. Recuperado 5 de marzo de 2021, de https://www.stratosphereips.org/datasets-overview

[5] Amar, Yousef & Haddadi, Hamed & Mortier, Richard & Brown, Tosh & Colley, James & Crabtree, Andy. (2018). An Analysis of Home IoT Network Traffic and Behaviour.

[6] Apthorpe, Noah & Reisman, Dillon & Feamster, Nick. (2017). A Smart Home is No Castle: Privacy Vulnerabilities of Encrypted IoT Traffic.

[7] B. K. J. Al-Shammari, N. Al-Aboody and H. S. Al-Raweshidy, "IoT Traffic Management and Integration in the QoS Supported Network," in IEEE Internet of Things Journal, vol. 5, no. 1, pp. 352-370, Feb. 2018, doi: 10.1109/JIOT.2017.2785219.

[8] Nguyen An, Hung & Silverston, Thomas & Yamazaki, Taku & Miyoshi, Takumi. (2021). IoT Traffic: Modeling and Measurement Experiments. IoT. 2. 140-162. 10.3390/iot2010008.

[9] Roy Chowdhury, Rajarshi & Aneja, Sandhya & Aneja, Nagender & Abas, Pg Emeroylariffion. (2020). Network Traffic Analysis based IoT Device Identification.

[10] Anderson, Blake & McGrew, David. (2017). Machine Learning for Encrypted Malware Traffic Classification: Accounting for Noisy Labels and Non-Stationarity. 1723-1732. 10.1145/3097983.3098163.

[11] A. Sivanathan, H. Habibi Gharakheili, F. Loi, A. Radford, C. Wijenayake, A. Vishwanath and V. Sivaraman, "Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics", IEEE Transactions on Mobile Computing, agosto, 2018.

[12] Hamza, Ayyoob & Ranathunga, Dinesha & Habibi Gharakheili, Hassan & Roughan, Matthew & Sivaraman, Vijay. (2018). Clear as MUD: Generating, Validating and Applying IoT Behavioral Profiles. 8-14. 10.1145/3229565.3229566.

[13] V. Thangavelu, D. M. Divakaran, R. Sairam, S. S. Bhunia and M. Gurusamy, "DEFT: A Distributed IoT Fingerprinting Technique," in IEEE Internet of Things Journal, vol. 6, no. 1, pp. 940-952, Feb. 2019, doi: 10.1109/JIOT.2018.2865604.

[14] TLS Fingerprinting. (2015). SquareLemon Blog. Recuperado 5 de mayo de 2021, de https://blog.squarelemon.com/tls-fingerprinting

[15] Anderson, Blake & McGrew, David. (2020). Accurate TLS Fingerprinting using Destination Context and Knowledge Bases.

[16] Husák, Martin & Cermak, Milan & Jirsik, Tomas & Celeda, Pavel. (2015). Network-based HTTPS Client Identification Using SSL/TLS Fingerprinting. 10.1109/ARES.2015.35.

[17] Cisco, “Joy: A package for capturing and analyzing network flow data and intraflow data, for network research, forensics, and security monitoring,” 2019. Recuperado 15 de mayo de 2021, de https://github.com/cisco/joy

[18] Hejcman, Lukas, Hynek, Karel, Cejka, Tomas (2021) JA3cury - A new approch to TLS fingerprinting by merging fingerprinting methods.

[19] Meghanathan, Natarajan & Allam, Sumanth & Moore, Loretta. (2010). Tools and techniques for Network Forensics. CoRR. abs/1004.0570.

[20] Baker, J. (2 de febrero de 2015). What is a Raspberry Pi? Recuperado el 10 de abril de 2021, a partir de https://opensource.com/resources/what-raspberry-pi

[21] Network bridge. (s. f.). ArchWiki. Recuperado 22 de abril de 2021, de https://wiki.archlinux.org/title/Network_bridge

[22] Introducing the Adafruit Bluefruit LE Sniffer. (2014, 19 noviembre). Adafruit Learning System. Recuperado 4 de junio de 2021, de https://learn.adafruit.com/introducing-the-adafruit-bluefruit-le-sniffer

[23] Bluetooth Technology Overview. (s. f.). Bluetooth® Technology Website. Recuperado 28 de junio de 2021, de https://www.bluetooth.com/learn-about-bluetooth/tech-overview/

[24] Shenzhen Sricctv Technology Co. Ltd. (s.f.). Recuperado 25 de agosto de 2021, de http://www.sricam.com/about_us.html?type=1

[25] Cox, K., & Gerg, C. (2007). Managing Security with Snort & IDS Tools. Van Duuren Media.

[26] Bluetooth Low Energy -It Starts with Advertising. (2017, 15 febrero). Bluetooth® Technology Recuperado 15 de septiembre de 2021, de https://www.bluetooth.com/blog/bluetooth-low-energy-it-starts-with-advertising/

[27] Jaswal, N. (2019). Hands-On Network Forensics. Van Haren Publishing.

[28] Real-Time Streaming Protocol Version 2.0. (s.f.). Recuperado 30 de septiembre de 2021, de https://tools.ietf.org/html/rfc7826+

[29] How to playback video from Network Traces. (s.f.). Recuperado 30 de septiembre de 2021, de https://knowledgebase-iframe.polycom.com/kb/viewContent.do;jsessionid=D89E4E28D117062BD8B37CF81C4FFD24?externalId=34917

[30] Lucas, M. (2010). Network Flow Analysis. Amsterdam University Press.

[31] Protocol Numbers. (s. f.). IANA. Recuperado 2 de octubre de 2021, de https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml

[32] ELK Stack: Elasticsearch, Logstash, Kibana. (s. f.). Elastic. Recuperado 2 de octubre de 2021, de https://www.elastic.co/es/what-is/elk-stack

[33] Geoip filter plugin | Logstash Reference [7.16]. (s. f.). Elastic. Recuperado 15 de octubre de 2021, de https://www.elastic.co/guide/en/logstash/current/plugins-filters-geoip.html

[34] Vega: A Visualization Grammar. (s. f.). Vega. Recuperado 23 de octubre de 2021, de https://vega.github.io/

[35] IPv4 Multicast Address Space Registry. (s. f.). IANA. Recuperado 12 de noviembre de 2021, de https://www.iana.org/assignments/multicast-addresses/multicast-addresses.xhtml

Agradecimientos

A mi tutor, Isaías García por la confianza depositada en mi para realizar este estudio y su orientación para la correcta realización del mismo.

A mi familia y, en especial a Laura Cantón, por su apoyo incondicional.

Anexos

ANEXO I: CREACIÓN DEL ENTORNO DE PRUEBAS PARA LA CAPTURA DE TRÁFICO

Con el fin de facilitar la replicación y creación del entorno de pruebas utilizado en este trabajo, se han creado una serie de *scripts* alojados en GitHub, en el repositorio *iot-network-analysis* (*https://github.com/ruben-rodriguez/iot-network-analysis*) y en concreto en la carpeta *raspi-switch*, validados en el sistema operativo *Raspbian.*

- El *script install,* instala los paquetes software necesarios para configurar un *bridge*, en particular *bridge-utils*, y para crear un punto de acceso *Wi-Fi,* en particular *hostapd.* Además, copia las configuraciones de *hostapd* y de *interfaces* presentes en el repositorio. Finalmente edita las rutas del *kernel*.

```
#!/bin/env bash

#Check if run as root
if [ "$UID" -ne 0 ] ; then
    echo "Run again as sudo!"
    exit 1
fi

echo "Checking connectivity..."
wget -q --tries=5 --timeout=20 --spider http://google.com
if [[ $? -eq 0 ]]; then
        echo "OK"
else
        echo "NO NETWORK: EXITING"
        exit
fi

echo "Updating packages database..."
apt-get update

echo "Installing required packages..."
apt-get install -y bridge-utils hostapd

echo "Copying config file under /etc/hostapd/hostapd.conf"
cp hostapd.conf /etc/hostapd/hostapd.conf

echo "Setting interfaces config, backup at /etc/network/interfaces.backup"
cp /etc/network/interfaces /etc/network/interfaces.backup
cp bridge-interfaces /etc/network/interfaces

echo "Disabling and enabling networking releated services..."
systemctl stop networking

systemctl unmask hostapd
systemctl start hostapd
systemctl start networking

echo "Modifying kernel routes..."
route add default gw 192.168.1.1 br0
route del default gw 192.168.1.1 eth0
ifconfig eth0 0.0.0.0

echo "Some bridge control commands to check the status:"
brctl show
brctl showstp br0
```

Figura Anexo I.1 *Script* de instalación y configuración de *switch* L2 en *Raspbian*

```
# Bridge name hostapd will be part of
bridge=br0
# Channel to use
channel=4
# Wifi driver to use
driver=nl80211
# 2.4GHz band
hw_mode=g
# What interface to bound hostapd to
interface=wlan0
# SSID name
ssid=IOT_TEST_LAB
# Auth section
wpa=1
wpa_passphrase=iottestlab
wpa_key_mgmt=WPA-PSK
```

Figura Anexo I.2 Configuración *hostapd* para la creación de punto de acceso *Wi-Fi*

```
# interfaces(5) file used by ifup(8) and ifdown(8)

# Include files from /etc/network/interfaces.d:
source-directory /etc/network/interfaces.d

auto lo
iface lo inet loopback

# Using 3 ports (built-in + 2 usb dongles) and wlan0
allow-hotplug eth0
iface eth0 inet manual
   pre-up   ifconfig $IFACE up
   pre-down ifconfig $IFACE down

allow-hotplug eth1
iface eth1 inet manual
   pre-up   ifconfig $IFACE up
   pre-down ifconfig $IFACE down

allow-hotplug eth2
iface eth2 inet manual
   pre-up   ifconfig $IFACE up
   pre-down ifconfig $IFACE down

allow-hotplug wlan0
iface wlan0 inet manual
 up hostapd -B /etc/hostapd/hostapd.conf

auto br0
iface br0 inet static
  bridge_ports eth0 eth1 eth2 wlan0
  address 192.168.1.156
  broadcast 192.168.1.255
  netmask 255.255.255.0
```

Figura Anexo I.3 Configuración de interfaces de red para crear *bridge* de red

- El *script uninstall* revierte los cambios de configuración de red aplicados por el *script install.*

```
#!/bin/env bash

#Check if run as root
if [ "$UID" -ne 0 ] ; then
    echo "Run again as sudo!"
    exit 1
fi

echo "Stopping hostapd..."
systemctl stop hostapd

echo "Restoring interfaces to default"
cp default-interfaces /etc/network/interfaces

echo "Delete br0 interface"
ip link delete br0

echo "Restart networking releated services..."
systemctl stop networking
systemctl start hostapd
systemctl start networking

echo "Modifying kernel routes..."
route add default gw 192.168.1.1 eth0

echo "Some bridge control commands to check the status:"
brctl show
brctl showstp br0
```

Figura Anexo I.4 *Script uninstall* para restaurar cambios en la configuración de red de *Raspbian*

Una vez ejecutado el *script install*, los puertos físicos de la *Raspberry Pi*, incluyendo la interfaz inalámbrica, funcionaran como un *bridge* de red, permitiendo la comunicación con la red de los dispositivos conectados a dichos puertos.

ANEXO II: INSTALACIÓN Y CONFIGURACIÓN DE ELK Y HERRAMIENTAS DE ANÁLISIS

Con el fin de facilitar la instalación, configuración y despliegue de la pila ELK y las herramientas de análisis utilizadas, se han creado una serie de *scripts* alojados en GitHub, en el repositorio *iot-network-analysis* (https://github.com/ruben-rodriguez/iot-network-analysis) y en concreto en la carpeta *ansible*, validados en el sistema operativo *Ubuntu Server 21.10. Ansible* es un motor que automatiza la ejecución de tareas para la preparación de infraestructura y gestionar su configuración.

Los *scripts* de *Ansible* reciben el nombre de *playbooks,* y se han desarrollado los siguientes:

- ***install-elk***: Instala los components de la pila ELK (*Elasticsearch, Logstash and Kibana*)
- ***configure-elk***: Configura la pila *ELK*, copiando los archivos de configuracion de la carpeta *elk-stack/config* del repositorio.
- ***install-joy***: Compila e instala *Cisco/joy*: *https://github.com/cisco/joy*
- ***install-mercury***: Compila e instala *Cisco/mercury: https://github.com/cisco/mercury*
- ***main***: Importa y ejecuta todos los *playbooks* listados anteriormente.

Para ejecutar los *playbooks* basta con tener instalado *Ansible* e invocar el siguiente comando, donde *hosts* es el archivo de inventario que utiliza *Ansible* para conectarse al servidor en el que se ejecutaran las diferentes acciones:

```
$: ansible-playbook -i hosts <playbook>.yaml
```

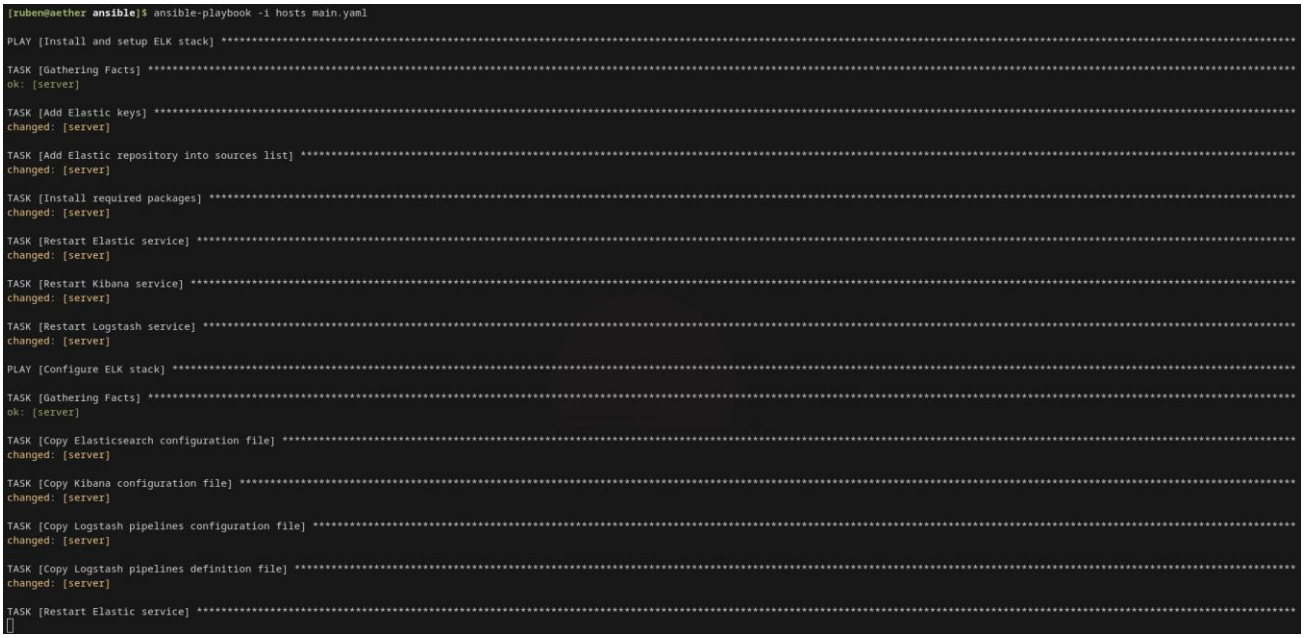


Figura Anexo II.1 Ejemplo de ejecución de los *playbooks* de *Ansible* desarrollados