Análisis y alerta del tráfico en las comunicaciones mediante técnicas de MITM

Autor: Sergio Vega Fernández
Tutor: Joan Caparrós Ramírez
Profesor: Helena Rifà Pous

Máster Universitario en Seguridad de las Tecnologías de la Información y de las Comunicaciones
Análisis de datos

Mayo 2020

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Análisis y alerta del tráfico en las comunicaciones mediante técnicas de MITM*
Nombre del autor:	*Sergio Vega Fernández*
Nombre del colaborador/a docente:	*Joan Caparrós Ramírez*
Nombre del PRA:	*Helena Rifà Pous*
Fecha de entrega (mm/aaaa):	*05/2020*
Titulación o programa:	*Máster Universitario en Seguridad de las Tecnologías de la Información y de las Comunicaciones*
Área del Trabajo Final:	*Análisis de datos*
Idioma del trabajo:	*Español*
Palabras clave	*Alerta, Tráfico, MITM*

Resumen del Trabajo:

Este trabajo tiene la finalidad de exponer una solución que permita detectar comportamientos extraños que delaten una posible infección o vulneración en dispositivos con acceso a Internet, por medio de un análisis del tráfico de red.

Para ello se ha generado una red local con acceso a Internet haciendo uso de un punto de acceso WiFi implementado en una Raspberry, mediante el cual dispositivos informáticos podrán acceder a Internet. Esta misma plataforma se encargará de realizar el análisis del tráfico de red generado utilizando técnicas de *man-in-the-middle*, gracias a la herramienta mitmproxy.

Posteriormente, el tráfico analizado se tratará a fin de detectar futuros riesgos. Esto se conseguirá mediante el desarrollo de un script escrito en Python, el cual se ejecutará con mitmproxy.

En caso de detectar comportamientos extraños, la solución implementada generará alertas en un canal privado de Telegram, que permitirá visualizar los intentos de conexiones catalogadas como sospechosas, las cuales han sido obtenidas de diferentes listas de reputación.

Los resultados obtenidos del producto desarrollado han sido satisfactorios, dado que se cumplen los objetivos establecidos. Aun así, no está de más mencionar que existe margen de mejora para futuras líneas de trabajo.

Finalmente, las conclusiones han sido bastante positivas en lo que a la generación de un sistema de estas características se refiere, puesto que se prevé que podrá ser de gran ayuda a la hora de mitigar riesgos en las comunicaciones de los dispositivos con acceso a Internet a un bajo coste.

Abstract:

The purpose of this work is to show a solution that allows the detection of strange behaviors that could reveal a possible infection in devices with Internet access through a network traffic analysis.
To emulate the purpose, a local network with Internet access has been generated using a WiFi access point implemented in a Raspberry. Through this access point computer devices could access to Internet. This platform will be responsible for analyzing the network traffic generated using man-in-the-middle techniques, by mitmproxy tool.
Subsequently, the analyzed traffic will be processed in order to detect future risks. This will be achieved by developing a script written in Python, which will be executed with mitmproxy.
In case of detecting strange behaviors the implemented solution will generate alerts in a private Telegram channel, which will allow visualizing the attempts of connections catalogued as suspicious. These connections catalogued as suspicious have been obtained from different reputation lists.
The obtained results from the developed product have been satisfactory, because established objectives were met. Even so, it should be mentioned that there is more room for improvement for future lines of work.
Finally, the conclusions have been quite positive as far as the generation of a system with these characteristics is concerned, because it is expected to be helpful at a low cost to mitigating risks in the communications of devices with Internet access.

Notaciones y Convenciones

A lo largo de este documento se hará uso de diferentes tipografías a fin de distinguir contenido en los textos presentados. Este apartado tiene como objetivo aclarar para que se utilizaran dichas tipografías. A continuación se mostrarán las tipografías usadas, las cuales se han clasificado en diferentes grupos teniendo en cuenta el uso que se les dará a las mismas.

Referencias

Cuando se haga referencia a apartados, imágenes y tablas del documento se hará uso de la tipografía cursiva, siendo el esquema:

- Para los apartados: *X.TítuloDeApartado*, siendo el valor *X* el numerador, como por ejemplo *2.1.3.Proxy a modo de MITM*.
- Para las figuras/imágenes: *Figura X: SubtítuloDeLaImagen*, siendo el valor *X* el numerador, como por ejemplo *Figura 10: Win 32 Disk Imager*.
- Para las tablas: *Tabla X: NombreDeLaTabla*, siendo el valor *X* el numerador, como por ejemplo *Tabla 1: Entregas*.

Cuando se haga referencia a cualquier elemento usado en la bibliografía se hará uso de un superíndice, siendo el esquema [*número*], en el cual el número enumerará a la cita correspondiente de la bibliografía, como por ejemplo [1]. El texto que previamente acompaña al número estará vinculado con la mencionada cita de la bibliografía.

Para referenciar a un elemento externo al documento, que podrá servir de ayuda para el lector del documento y el cual se pueda acceder vía web, se hará uso de un hipervínculo, siendo el esquema *elementoContendorDelHipervínculo* *número*, en el cual el número enumerará a la cita correspondiente de la referencia, como por ejemplo El canal [10]. Dentro de la bibliografía habrá un apartado donde se muestren las páginas web a las que estos elementos hacen referencia.

Valores y código

Cuando a lo largo del documento se quieran distinguir nombres de archivos, librerías, APIs, opciones o valores a presentar se hará uso de la fuente `Courier New`, como por ejemplo `mitmproxy`. Aclarar que a la hora de citar los nombres de diferentes herramientas o programas no se hará uso de la fuente `Courier New`, como puede darse el caso en mitmproxy, siendo en este caso usado para nombrar la herramienta, y siendo `mitmproxy` usado para nombrar la librería Python, o en su defecto el comando que se ejecuta por consola.

Para nombrar valores genéricos se hará uso de la cursiva a la vez que la fuente *Courier New* y ofreciendo una explicación de dicho valor genérico, como por ejemplo *NombrePublicoDelCanal*, el cual hace referencia al nombre que tomará un canal público, pudiendo este tomar el valor de, por ejemplo, Canal_TFM2020.

Cuando se haga referencia a cualquier comando que requiera ejecutarse por consola se hará siguiendo el siguiente esquema:

```
$ Comando
```

Como por ejemplo:

```
$ sudo passwd
```

Cuando se haga referencia a cualquier contenido de un archivo se hará siguiendo el siguiente esquema:

```
#Comentario
Contenido
```

En donde los valores o parámetros (el contenido) estarán representados con el siguiente color de fuente:

```
Parámetro o valor
```

Y los comentarios de archivo mediante el color de fuente que se presenta a continuación, aparte de empezar con el carácter #.

```
#Esto será un comentario
```

A continuación se muestra un ejemplo de un archivo de configuración:

```
interface=wlan0
#Ejemplo de direcciones
dhcp-range=192.168.100.2,192.168.100.20,255.255.255.0,24h
```

Notaciones

Cuando a lo largo del documento se quiera mostrar cualquier nota a fin de añadir alguna observación, advertencia o apunte, se hará siguiendo el siguiente esquema, el cual empezará siempre por **NOTA:** o **IMPORTANTE:**, tal y como se muestra a continuación:

NOTA: *LaNota*.

Como por ejemplo:

NOTA: Antes de iniciar la configuración, puede darse la posibilidad de que haya que habilitar la tarjeta de red inalámbrica.

Índice

Figuras y tablas

Índice de figuras

Índice de tablas

1. Introducción

1.1. Contexto

En plena revolución digital, el uso generalizado de diferentes tipos de dispositivos acompañado de mejoras en las redes de telecomunicaciones han cambiado la vida de la sociedad actual, haciendo posible efectuar acciones cotidianas de una forma digital.

Hoy en día se puede observar como las personas de diferentes países en su día a día, cada vez de forma más frecuente, hacen mayor uso de dispositivos digitales, siendo estos principalmente dispositivos móviles dotados de acceso a Internet con el fin de acceder a diferentes servicios. Un ejemplo de ello es por medio de aplicaciones de mensajería, instaladas en los mencionados dispositivos digitales, las cuales mediante el uso de Internet hacen posible la comunicación entre diferentes usuarios.

La tendencia al alza del uso de este tipo de tecnologías indica el impacto causado en la sociedad, aportando ventajas y facilidades para la mayoría de los usuarios. Entre las ventajas más destacables se pueden distinguir la comodidad, la globalización y la cobertura de necesidades. Se entiende por comodidad el hecho de facilitar la realización de diferentes tareas; cuando se hace mención al término globalización se refiere a un acceso de recursos para todo el mundo; finalmente, la cobertura de necesidades hace alusión a la resolución de los problemas existentes o de los problemas que van surgiendo.

No obstante, el extenso uso de las tecnologías mencionadas también tiene sus inconvenientes como la sobrecarga informativa, dicho de otra forma, el contar con demasiada información, y principalmente, en lo que al ámbito de seguridad se refiere, el inconveniente de los ataques informáticos. Este último inconveniente se ha incrementado exponencialmente a medida que se prolifera el uso de las tecnologías citadas.

Los diferentes tipos de ataques informáticos, el mencionado crecimiento y la evolución de estos hacen que la sociedad esté expuesta a una gran cantidad de distintos tipos de riesgos, a los que la mayoría de usuarios no están concienciados, principalmente por desconocimiento y seguido del hecho de tener actitudes reacias hacia el aprendizaje para la mitigación de los mencionados riesgos, en gran medida por la posición optada de no tener interés del saber cómo funciona, pero si tener interés de que funcione.

1.2. Problema a resolver

Teniendo en cuenta la actitud mostrada por una gran cantidad de personas en la sociedad hacia la tecnología y el crecimiento del uso de esta, mencionado en el anterior punto *1.1.Contexto*, mediante este trabajo se pretende minimizar los riesgos causados por el uso de la misma. Bien es cierto que existen diferentes tipos de riesgos y que es un campo muy amplio para abarcar, por eso se ha decidió que este proyecto se centrará en la mitigación de los riesgos en las comunicaciones de los dispositivos digitales dotados de acceso a Internet.

El trabajo pretende abordar este tema debido al uso diario y masivo de la tecnología presentada. Esto se puede ver reflejado en los datos aportados por el portal de estadísticas online Statista [1], los cuales dictan que en 2019 había una cantidad aproximada de 4.130 millones de usuarios que usa dispositivos con acceso a Internet en el mundo, es decir, aproximadamente el 53,64% de la población mundial, teniendo en cuenta que según las Naciones Unidas la población mundial a mediados del 2019 era de 7.700 millones de personas [2]. La estadística de Statista también muestra la tendencia al alza de los usuarios que usa dispositivos con acceso a Internet a medida que pasan los años.

En este proyecto se emulará un entorno para que dispositivos digitales tengan acceso a Internet, el cual a su vez analizará el tráfico entrante y saliente de dichos dispositivos para posteriormente llevar a cabo un tratamiento. Para ello se hará uso de un punto de acceso (AP) creado mediante una Raspberry Pi, que permitirá conectarse a Internet a los mencionados dispositivos digitales. A su vez se pretende que la Raspberry haga un análisis del tráfico entrante y saliente de los dispositivos digitales mediante técnicas de *man-in-the-middle* (MITM), llevando a cabo un tratamiento y alertando, si fuese necesario, sobre el tráfico sospechoso generado por medio de los dispositivos digitales con el fin de evitar futuros riesgos.

1.3. Objetivos generales

En este trabajo se trataran diferentes objetivos presentados en distintas categorías y mostrados en orden de relevancia.

1.3.1. Objetivos principales

Los objetivos principales de este proyecto son:

- Desarrollar un sistema *man-in-the-middle* para realizar un análisis de tráfico.
- Generar un punto de acceso con salida a Internet mediante una Raspberry Pi.
- Buscar una solución a implantar que ayude a tratar el tráfico generado por dispositivos digitales.
- Generar un sistema de alertas que permita obtener información sobre el tráfico sospechoso generado.
- Implantar un escenario donde se haga uso de todos los anteriores objetivos presentados.
- Elaborar un informe que recoja el trabajo desarrollado.

1.3.2. Objetivos secundarios

Los objetivos adicionales que se pretenderán alcanzar tras realizar este proyecto:

- Ofrecer la implantación de un sistema que cumpla con los objetivos principales de la manera más simple y sencilla posible.
- Ofrecer una demostración del producto desarrollado.
- Aumentar el conocimiento del autor del proyecto en los siguientes temas abordados:
    - Las diferentes maneras de generar un punto de acceso.
    - Las diferentes maneras de implantar un analizador de tráfico.
    - Las diferentes maneras de realizar un tratamiento de tráfico.
    - Las diferentes maneras de generar un sistema de alertas.
    - La configuración y el uso de las diferentes herramientas, programas, técnicas,… usados.

1.4. Metodología y proceso de trabajo

Para lograr los objetivos finales de este proyecto se ha planteado la línea de trabajo que se explicará a continuación.

Como punto de partida se obtendrá información relacionada con el tema que se pretende abordar y se hará una pequeña revisión del estado del arte. Esta acción ayudará a definir los objetivos del proyecto que se generará, obteniendo así un primer marco para el desarrollo del mismo.

Posteriormente, se hará un estudio logrando información relativa a los objetivos del trabajo previamente definidos. Para ello se recopilará información de diversas fuentes, en la mayoría de los casos se adquirirá de Internet, obteniendo así diferentes soluciones que ayudarán a cumplir con los objetivos establecidos.

A continuación se analizará la información recopilada y se escogerá la que se prevé que será la solución más adecuada, la cual se implantará a lo largo del proyecto. El requisito mínimo será que la solución propuesta será un recurso actualmente existente y activo o en uso, se descartarán soluciones que se han quedado obsoletas.

Tras optar por la solución que se usará para lograr los fines definidos en el proyecto, se procederá al diseño en donde se tratarán los requisitos necesarios del producto para su posterior desarrollo.

Finalmente se desarrollara el producto con el fin de intentar lograr los objetivos definidos en el plan de trabajo.

Cabe mencionar que todo el trabajo realizado se recogerá en un parte de horas para realizar un detallado control y seguimiento del proyecto. También se irán recogiendo las acciones realizadas a lo largo del proyecto en la memoria del trabajo.

1.5. Planificación

En este punto se mostrará la ejecución estructurada de trabajo que se seguirá con el fin de conseguir los objetivos establecidos. Esta estructura se dividirá en diferentes fases: la gestión del proyecto, el producto y la entrega final.

La fase de la gestión del proyecto incluye el plan de trabajo que se aplicará a lo largo del proyecto y el control y seguimiento de este. El control y seguimiento del proyecto tiene la finalidad de comprobar el estado del mismo, y observar así que los objetivos se están cumpliendo en base a lo previsto. En este proyecto el plan de trabajo se establecerá en la primera entrega (ver *Tabla 1: Entregas*), en cambio, el control y seguimiento se realizará a lo largo de todo el proyecto hasta la fase de la entrega final.

La fase del producto recoge el desarrollo realizado para cumplir los objetivos establecidos. Esta fase consta de dos elementos principales, el análisis y el desarrollo. El producto se realizará en las fechas desde que comienza la segunda entrega, relacionada con el análisis, hasta que termina la tercera entrega, relacionada con el desarrollo.

La fase de la entrega final contiene los elementos para dar por concluido el proyecto. Los elementos de la entrega final del proyecto son la memoria final de este, la cual se establecerá en la cuarta entrega (aunque a medida que se vaya realizando el proyecto esta se irá redactando), la presentación en video, la cual se establecerá en la quinta entrega, y finalmente, la defensa del mismo.

Los hitos de este proyecto coincidirán con las diferentes entregas marcadas por la UOC en la asignatura del Trabajo Final de Máster (TFM) y que se recogen en la siguiente tabla:

Nombre de la entrega	Fecha de la entrega	Descripción
Entrega 1	03/03/2020	Plan de trabajo
Entrega 2	31/03/2020	PEC 2
Entrega 3	28/04/2020	PEC 3
Entrega 4	02/06/2020	Memoria final
Entrega 5	09/06/2020	Presentación en video
Defensa	19/06/2020	Defensa del TFM

Tabla 1: Entregas

1.5.1. Tareas

Cada uno de los elementos descritos en las diferentes fases del apartado *1.5. Planificación* están formados por distintas tareas. En este apartado se recogen dichas tareas para cada uno de los elementos de las diferentes fases.

Fase 1: Gestión del proyecto

- Plan de trabajo, cuyas tareas son las siguientes:
  - Contexto y la explicación del problema a resolver.
  - Objetivos del trabajo.
  - Descripción de la metodología usada.
  - Planificación del trabajo.
  - Presupuesto.
  - Revisión del estado del arte.
  - Análisis de riesgos.
  - Requisitos legales.
- Control y seguimiento.

Fase 2: Producto

- Análisis, cuyas tareas son las siguientes:
  - Investigación del analizador de tráfico, dentro del cual se incluirá lo siguiente:
    - Estudio de las alternativas.
    - Estudio de los dispositivos necesarios.
    - Estudio de la configuración.
    - Elección de la alternativa.
  - Investigación sobre los puntos de acceso, dentro del cual se incluirá lo siguiente:
    - Estudio de las alternativas.
    - Estudio de los dispositivos necesarios.
    - Estudio de la configuración.
    - Elección de la alternativa.
  - Investigación sobre el tratamiento del tráfico, dentro del cual se incluirá lo siguiente:
    - Estudio de las alternativas.
    - Estudio de la configuración.
    - Elección de la alternativa.
  - Estudio de la topología de red a usar.
- Desarrollo, cuyas tareas son las siguientes:
  - Configuración y diseño del entorno, dentro del cual se incluirá lo siguiente:
    - Preparación de los dispositivos necesarios.
    - Implantación del analizador de tráfico.

        - Implantación del punto de acceso.
        - Implantación del tratamiento del tráfico.
        - Generación del sistema.
    - Realización de las pruebas y demostración, dentro del cual se incluirá lo siguiente:
        - Conexión de los dispositivos.
        - Análisis de tráfico.
        - Tratamiento del tráfico.

Fase 3: Entrega final

- Memoria final, cuyas tareas son las siguientes:
    - Redactar la memoria.
    - Aplicar revisión, enriquecimiento y corrección de los textos a presentar.
- Preparación del video de presentación.
- Defensa del proyecto.

En resumen, las tareas del proyecto se reflejan mediante la estructura de descomposición del trabajo (EDT) que se muestra a continuación.

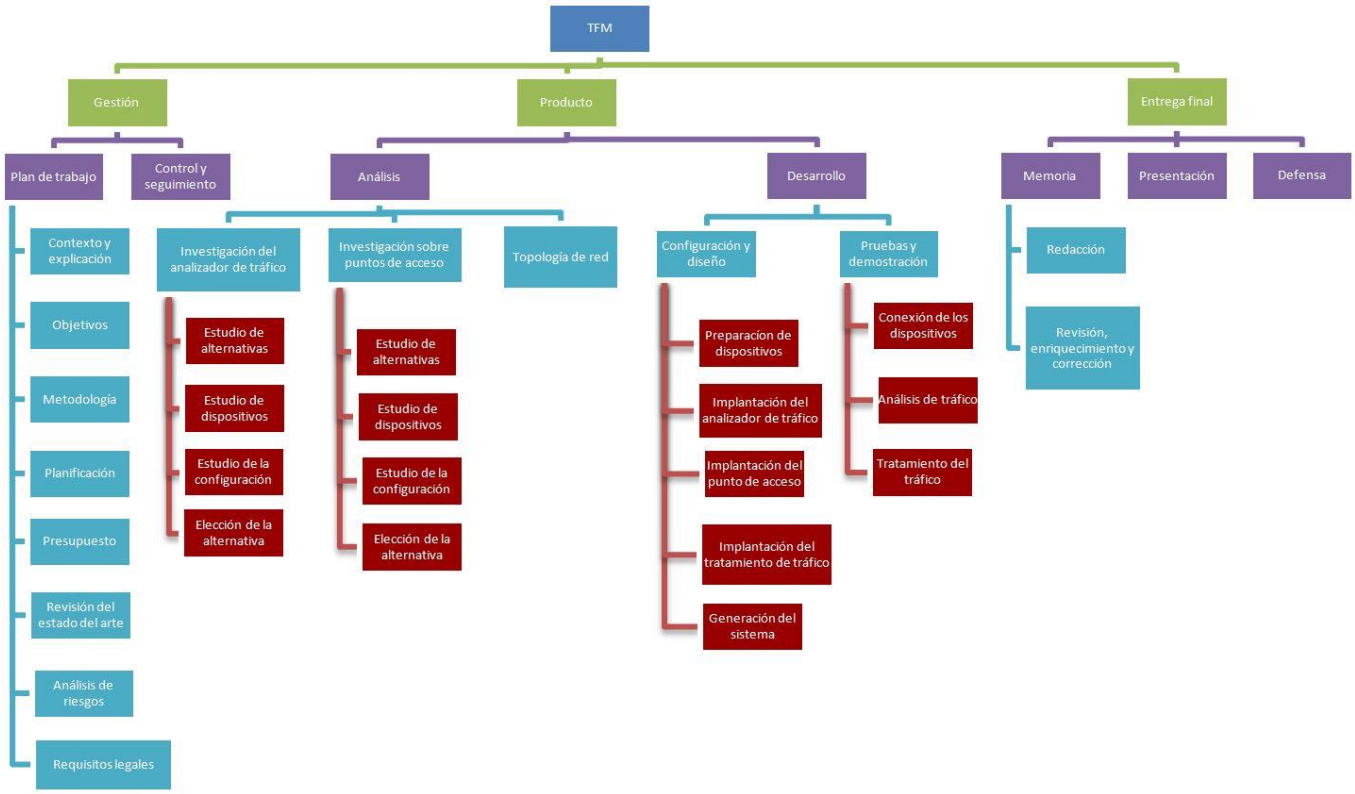


Figura 1: Estructura de Descomposición del Trabajo

1.5.2. Planificación temporal

La planificación temporal se ha generado en base al periodo que se proporciona para cursar el TFM y la dedicación prevista para desarrollar el mismo. El periodo proporcionado para realizar y dar por concluido el TFM es de 18 semanas. Por su parte, se intuye que se necesitarán una media de 14 horas semanales para efectuar el proyecto, lo cual hace un cálculo de 252 horas totales para la realización del proyecto. Teniendo en cuenta que la carga docente que el TFM le supone al alumno es de es de 9 créditos, lo que equivale a 225 horas, se dará por valido el tiempo de 252 horas estipulado para la realización del proyecto.

Los periodos y las horas se estima que dividirán entre las tareas mencionadas en el apartado *1.5.1.Tareas* de la manera que se muestra en la siguiente tabla.

Fase del proyecto	Tarea	Horas asignadas
Fase 1: Gestión	Plan de trabajo	26 h
	Contexto y explicación	6 h
	Objetivos	3 h
	Metodología	2 h
	Planificación	6 h
	Presupuesto	2 h
	Revisión del estado del arte	2 h
	Análisis de riesgos	2 h
	Requisitos legales	3 h
	Control y seguimiento	3 h
Fase 2: Producto	Análisis	65 h
	Investigación del analizador de tráfico	24 h
	Estudio de alternativas	12 h
	Estudio de dispositivos	1 h
	Estudio de la configuración	8 h
	Elección de la alternativa	3 h
	Investigación sobre puntos de acceso	20 h
	Estudio de alternativas	8 h
	Estudio de dispositivos	1 h
	Estudio de la configuración	8 h
	Elección de la alternativa	3 h
	Investigación del tratamiento del tráfico	11 h
	Estudio de alternativas	3 h
	Estudio de la configuración	7 h
	Elección de la alternativa	1 h

	Topología de red	10 h
	Desarrollo	55 h
	Configuración y diseño	40 h
	Preparación de dispositivos	8 h
	Implantación del analizador de tráfico	12 h
	Implantación del punto de acceso	8 h
	Implantación del tratamiento del tráfico	10 h
	Generación del sistema	2 h
	Pruebas y demostración	15 h
	Conexión de los dispositivos	5 h
	Análisis de tráfico	5 h
	Tratamiento del tráfico	5 h
Fase 3: Entrega final	Memoria	90 h
	Redacción	75 h
	Revisión, enriquecimiento y corrección	15 h
	Presentación	10 h
	Defensa	3 h

Tabla 2: Horas del proyecto

A continuación se muestra el diagrama de Gantt de este TFM, en el cual se expone la planificación de que tareas se desarrollarán en un periodo real de tiempo.

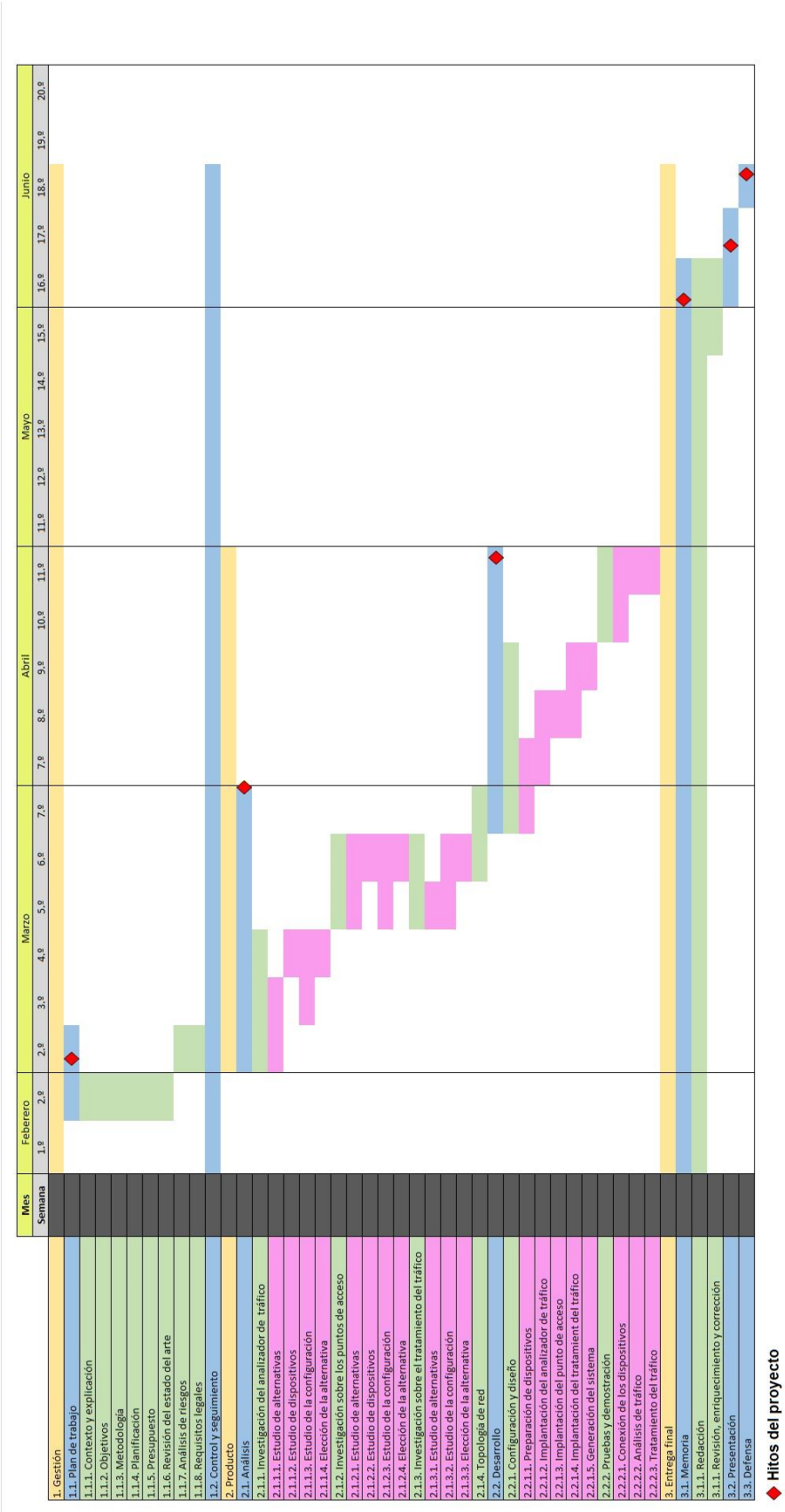


Figura 2: Diagrama de Gantt

1.6. Revisión del estado del arte

El desarrollo del proyecto se basa en dos principales componentes, la implantación de un analizador de tráfico, gracias al cual posteriormente se podrá hacer un tratamiento del tráfico, y un punto de acceso. Para ello se ha realizado una breve investigación de estos elementos con dos principales objetivos:

- Obtener información con el fin de estudiar la viabilidad del proyecto.
- Hacer un estudio del mercado con las alternativas existentes para el posterior desarrollo del producto.

Se han obtenido los resultados que se expondrán a continuación.

Analizador de tráfico

Existen diferentes alternativas para desarrollar un analizador de tráfico. A la hora de escoger la alternativa a usar en este proyecto, se tendrán en cuenta las características del dispositivo en donde se pretenderá implantar el analizador de tráfico.

En este caso, la implantación se pretenderá que sea en una Raspberry Pi, así pues, las principales características que se tendrán en cuenta serán características relativas a la arquitectura, a fin de que la alternativa del analizador de tráfico sea compatible con la Raspberry.

Cabe mencionar que también se han tenido en cuenta aquellas alternativas que se puedan implantarse en dispositivos cuya arquitectura sea diferente a la de la Raspberry, gracias a esto se conseguirá mayor margen para la viabilidad del proyecto.

A continuación se presentarán las diferentes alternativas estudiadas para realizar el desarrollo de un analizador de tráfico.

- La primera alternativa es la del uso de herramientas conocidas como *sniffers*, o analizadores de paquetes de tráfico. Destacan dos tipos de analizadores de tráfico, los que operan sobre entorno gráfico, por ejemplo Wireshark, y los que operan por consola, por ejemplo Tshark o tcpdump.
- La segunda alternativa es el uso de sistemas de detección de intrusos (IDS) entre los cuales se destacarán Snort y Suricata.
- La alternativa final es la del uso de proxys a modo de MITM, como por ejemplo las herramientas existentes mitmproxy o Burp Proxy.

Señalar que algunas de las alternativas presentadas dotan de la posibilidad de poder tatar el tráfico una vez lo hayan analizado.

Punto de acceso

Para la implantación de un punto de acceso se han estudiado dos principales alternativas presentadas a continuación.

- La primera alternativa estudiada es la del uso de un dispositivo el cual se configurará para que ejerza de punto de acceso. Dentro de los dispositivos físicos existentes hay una infinidad de combinaciones diferentes, ya sea por las características del dispositivo físico, por el sistema operativo, software a usar, etc.
- La segunda alternativa estudiada es al del uso de un punto de acceso cotidiano y usando en el día a día como es un encaminador. Los encaminadores son dispositivos con un firmware propio el cual está basado en sistemas operativos conocidos, como puede ser el caso de OpenWrt basado en un sistema operativo Linux.

Cada alternativa dispone de características que favorecen y perjudican la elección de su uso. En este proyecto se pretenderá configurar un dispositivo para que ejerza de punto de acceso. Para ser más exactos se pretenderá hacer uso de una Rasperry Pi, previamente mencionada, por el principal motivo de minimizar costes.

Mencionar que si se tienen en cuenta uno de los objetivos definidos en el proyecto (ver *1.3.1.Objetivos principales*), el cual dicta se deberá generar un punto de acceso, únicamente cabría la posibilidad de usar la primera alternativa presentada. Aun así, se ha querido presentar la segunda alternativa para ofrecer mayor margen en vistas a la viabilidad del proyecto.

1.7. Requisitos legales

Teniendo como referencia el REGLAMENTO (UE) 2016/679 DEL PARLAMENTO EUROPEO Y DEL CONSEJO [3], también conocido como el Reglamento general de protección de datos (RGPD), aplicado desde el 25 de mayo de 2018 en la Unión Europea, y la adaptación del mismo a la normativa española, la Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales (LOPD-GDD) [4], en este proyecto se deberán tener en cuenta una serie de cuestiones legales relacionados con la protección de datos personales del usuario debido a acciones llevadas a cabo que implican el tratamiento de los mismos.

Partiendo de la idea de que este proyecto basa su desarrollo en un análisis del tráfico de red, se identifican diferentes actores que forman parte de un tratamiento de datos personales (aquella información sobre una persona física identificada o identificable, en este proyecto, parte de la información recogida en el análisis del tráfico proveniente del dispositivo a analizar). Estos actores están obligados a cumplir el previamente mencionado RGPD.

El primer actor es el Responsable del tratamiento o *Data controller*, definido como la persona física o jurídica, autoridad pública, servicio u otro organismo que, solo o junto con otros, determine los fines y medios del tratamiento. En este proyecto se identifica como Responsable del tratamiento al administrador del sistema de análisis de tráfico, siendo este el propio desarrollador del proyecto una vez haya concluido con el mismo.

El segundo y último actor es el Encargado del tratamiento o *Data processor*, definido como la persona física o jurídica, autoridad pública, servicio u otro organismo que trate los datos en nombre del Responsable del tratamiento. En este proyecto se identifica el analizador de tráfico como el Encargado del tratamiento. Al ser un proceso automatizado se le asignará un responsable una vez el desarrollo del proyecto haya concluido, en este caso, el propio desarrollador ejercerá de responsable.

Así pues, identificados los actores, el proyecto deberá asegurar que se establecen los siguientes principios regidos por el RGPD:

- Licitud, lealtad y transparencia: Los datos personales se deberán tratar de forma lícita y transparente. Se garantizará lealtad hacia las personas cuyos datos personales se está tratando.
- Limitación de la finalidad: El tratamiento de datos deberá tener fines específicos. Dichos fines se indicarán a las personas cuyos datos personales se quieran tratar.
- Minimización de datos: Solo se deberán recopilar y tratar los datos personales que sean necesarios para poder cumplir la finalidad.
- Exactitud: Se deberá asegurar que los datos recopilados están actualizados, sino, se aplicarán las correcciones necesarias.

- Limitación del plazo de conservación: Se deberán conservar los datos durante el tiempo necesario para los fines para los que fueron recopilados. Los datos no se podrán seguir usando con fines no compatibles con los fines definidos.
- Integridad y confidencialidad: Se deberán establecer garantías de que los datos recopilados no se trataran si no se tiene la autorización, se ofrecerán garantías contra su pérdida, daño, destrucción, etc.

Finalmente, se tendrá que informar al usuario los derechos otorgados por el RGPD que posee:

- Derecho a la información: Podrá saber sobre el tratamiento de los datos proporcionados.
- Derecho a obtener acceso a los datos que ha proporcionado.
- Derecho al olvido: Podrá solicitar que los datos personales sean borrados.
- Derecho a oponerse: Podrá oponerse al tratamiento de los datos personales.
- Derecho a solicitar limitación del tratamiento de sus datos personales.
- Derecho a la portabilidad de datos: Podrá solicitar y recibir los datos tratados. Dichos datos los podrá transmitir a otro responsable del tratamiento de datos.
- Derecho a solicitar decisiones basadas en un tratamiento automatizado sean tomadas por personas físicas y no solamente por ordenadores. También tiene derecho, en este caso, a expresar su punto de vista y a impugnar dicha decisión.

Con el fin de cumplir lo establecido a lo largo de este punto, en este proyecto se deberán seguir varias pautas, las cuales se recogen a continuación:

1. El desarrollador del proyecto deberá implantar soluciones para cumplir con los requisitos legales necesarios. Por ejemplo, limitando la disposición de los datos que se tratarán, siendo estos destruidos una vez se hayan tratado.
2. Una vez desarrollado el proyecto, se deberán generar los documentos/contratos necesarios donde se recoja la información junto con las obligaciones necesarias a fin de cumplir el RGPD. Lo ideal será generar los documentos/contratos adaptados al proyecto, aun así, se prevé que se podrá hacer uso de plantillas [1] obtenidas por terceros.
3. Se deberán fijar planes de mitigación en caso de ocurrir acciones inesperadas. Posteriormente estos planes se podrán enriquecer si se da el caso de que ocurra alguna acción inesperada, la cual no estaba contemplada. A la hora de generar este documento la única acción inesperada que se ha contemplado es la siguiente:
    - En caso de llevarse a cabo el tratamiento de datos de un usuario que no ha proporcionado su consentimiento, por causas como la de que hace uso del sistema sin aprobación previa, inmediatamente se deberá parar el tratamiento de dichos datos, se le deberá informar al usuario que datos han sido tratados, y finalmente, se deberán borrar todos los datos tratados y los resultados obtenidos a raíz de estos.

1.8. Recursos y presupuesto del proyecto

Para fijar el presupuesto total de este proyecto se han tenido en cuenta los diferentes tipos de costes que se dan en el mismo. En este proyecto se han localizado dos tipos de costes, el coste del desarrollo del proyecto y el coste del material usado para realizar el trabajo.

El coste del desarrollo del proyecto es el esfuerzo invertido por el alumno para llevar a cabo el proyecto en lo que ha dedicación de tiempo se refiere. La dedicación prevista se recoge en la anterior sección *1.5.Planificación* de este documento, mientras que gracias al control y seguimiento realizado en el proyecto se sabe que el esfuerzo total invertido ha sido de 278 horas.

En este punto no se entrará en detalle de la desviación del esfuerzo estimado al total invertido, puesto que se tratará en el punto *5.3.Seguimiento y control*, pero si se quiere destacar que si hacen cuentas al alumno le ha requerido invertir 26 horas más de las estipuladas.

Por otra parte, el material usado en el proyecto y su coste se lista en la tabla mostrada a continuación (ver *Tabla 3: Presupuesto del proyecto*).

Recurso	Empleo	Unidades	Coste
Ordenador HP Pavilion 15 Notebook	Redactar la memoria. Administrar la Raspberry vía SSH. Realizar pruebas como cliente.	1	500 €
Raspberry Pi 3	Implantar el analizador de tráfico. Implantar el punto de acceso. Tratar el tráfico.	1	35 €
Fuente de alimentación Raspberry	Alimentar la Raspberry.	1	11,99 €
Disipadores de calor para la Raspberry	Evitar que la Raspberry se caliente.	1	11,99 €
Carcasa protectora Raspberry	Aumentar la seguridad física de la Raspberry.	1	11,99 €
Tarjeta de memoria SanDisk Ultra 16GB	Implantar el sistema operativo que usará la Raspberry.	1	5,99 €
Cable RJ-45	Habilitar conexiones Ethernet de la Raspberry.	1	1,75 €
Encaminador TP-LINK modelo TL-WR740N	Proveer de conexión a Internet	1	16 €
Conexión a Internet	Salida hacia Internet para el tráfico generado	6 meses	25€ / mes
Teclado Logitech K400	Configurar la Raspberry	1	29,98 €
Monitor DELL P2319H	Visualizar el contenido de la Raspberry	1	183 €
Cable HDMI	Conectar la Raspberry al monitor	1	5,99 €
Dispositivo Android Samsung Galaxy A40	Visualizar alertas	1	180 €

Tabla 3: Presupuesto del proyecto

NOTA: Si hay materiales que se encuentran seguidos y tienen el mismo color de fondo entre sí (que no sea el color blanco), significará que han sido obtenidos en un pack, por ello tanto el valor de **Unidades** y **Coste** será el mismo. Este valor se deberá tener en cuenta una única vez.

1.9. Análisis de riesgos

La siguiente sección identifica los riesgos que pueden ocurrir a lo largo de las diferentes fases del proyecto. Con interés de que estos no influyan, en medida de lo posible, a la hora de alcanzar el objetivo o afectar en el desarrollo del proyecto se han planteado posibles soluciones.

Riesgo 1: Perdida de datos

El primer riesgo que se identifica es la de la pérdida de datos tales como desarrollos, soluciones, documentos,… a lo largo del proyecto. La posibilidad de que esto ocurra se prevé que será baja.

La solución propuesta para evitar males mayores a consecuencia de este riesgo es que en cada desarrollo llevado a cabo se hará una copia de seguridad que posteriormente se almacenará en la nube y en un dispositivo de almacenamiento externo.

Riesgo 2: Objetivos ambiciosos

Otro riesgo que se ha localizado es el haber generado una planificación ambiciosa. En este caso la probabilidad de que esto ocurra se prevé que será media/alta por dos motivos. El primero, y el principal, se debe a que el alumno que realiza el TFM, habitualmente, en base a lo estimado al principio suele tener desviaciones finales. El segundo motivo es que se prevé que existirán complicaciones a la hora del desarrollo del producto y seguramente haya desviaciones respecto a la planificación establecida.

La solución propuesta, con el fin de justificar el porqué no se ha ejecutado el plan de trabajo indicado, es que el alumno del TFM documentará las problemáticas obtenidas para no cumplir con lo establecido en dicho plan. Para ello, recogerá y mostrará las desviaciones ocurridas.

Riesgo 3: Fallos en herramientas/entornos

Finalmente, el último riesgo que se identifica es el fallo en las herramientas o en el entorno escogido para desarrollar el producto. La posibilidad de que esto ocurra se prevé que será baja/media. Tras hacer una breve revisión del estado del arte como se muestra en el punto *1.6. Revisión del estado del arte*, se ha dado por hecho que las soluciones mostradas parecen estables, aun así, quedaría por probar si dichas soluciones se adaptan a los recursos disponibles.

La solución propuesta para mitigar el riesgo es que se estudiará el uso de otro entorno u otras herramientas, recogiendo las justificaciones del cambio.

2. Fase de investigación

En este capítulo se mostrará la investigación llevada a cabo, proporcionando una visión general de las diversas soluciones implantadas a lo largo del proyecto. Cada solución implantada contará con una explicación dividida en tres puntos principales. En el primer punto y a modo de entrada, se ofrecerá una introducción. Posteriormente se presentarán las diferentes soluciones investigadas. Finalmente se especificará la solución escogida para su posterior implantación en el desarrollo del proyecto.

2.1. Analizador de tráfico

La idea principal de este proyecto se basa en el análisis del tráfico para la posterior mitigación de amenazas, así que teniendo en cuenta esta afirmación, se podrá decir que el analizador de tráfico es la herramienta la cual se considerará el núcleo para el desarrollo del proyecto.

El analizador de tráfico se encargará de inspeccionar los paquetes que se envían desde dentro de una red (adelantando hechos, desde dentro de una red local que se generará) hacia Internet. Para ello se han investigado diferentes alternativas las cuales se presentarán a continuación.

2.1.1. Sniffer

El *sniffer*, también conocido como el analizador de paquetes, es un dispositivo (en la mayoría de casos un software) que se utiliza para capturar los paquetes que viajan por la red de la cual es parte para posteriormente obtener información de los mismos.

Actualmente en el mercado existen diferentes alternativas, siendo las alternativas de software libre las analizadas en este proyecto, entre las cuales se encuentran las siguientes:

Wireshark [5]
Wireshark, anteriormente conocido como Ethereal, es un analizador de paquetes de red publicado bajo la licencia General Public License (GPL), de código abierto y desarrollado por Gerald Combs en julio de 1998. Actualmente se encuentra entre los *sniffers* más extendidos del mundo gracias a las diferentes características de las cual está dotado.

Esta herramienta multiplataforma opera sobre un entorno gráfico, o *graphical user interface* (GUI), y está disponible para sistemas operativos UNIX y Windows.

Es capaz de capturar datos del tráfico que circula en una red de la cual es parte, u obtener información a partir de datos almacenados en diferentes archivos de captura, como por ejemplo, archivos `pcap`.

Tiene compatibilidad con diferentes protocolos (ARP, TCP, HTTP,…), es decir, facilita información de los protocolos de diferentes niveles, como puede ser el nivel de red, de transporte, etc.

También dispone de compatibilidad con diferentes estándares o especificaciones, como pueden ser Ethernet, IEEE 802.11, Bluetooth, etc.

Mencionar que según sus desarrolladores esta herramienta dista de ser, por una parte, un sistema de detección de intrusos, puesto que no es capaz de generar alertas tras la detección de paquetes; y por otra parte, un dispositivo manipulador del tráfico dentro de la red en la que actúa, debido a que no dispone de la funcionalidad de envío de paquetes.

Finalmente, Wireshark dispone de una versión orientada a terminales llamada TShark, la cual soporta las mismas opciones que Wireshark. Está diseñada para capturar y mostrar paquetes cuando no se requiera de una GUI.

Tcpdump [6]

Esta herramienta es un analizador de paquetes de red publicado bajo la licencia Berkeley Software Distribution (BSD), de código abierto y desarrollado en el Laboratorio Lawrence Berkeley de la Universidad de California, en Berkeley en 1987.

La base de tcpdump es la librería `libpcap`, escrita en lenguaje C y creada por los mismos desarrolladores que tcpdump. Esta librería facilita la captura de paquetes a nivel de usuario. La mayoría del software cuya finalidad es el análisis de paquetes de red está basado en esta librería.

Tcpdump se ejecuta desde la línea de comandos o terminal y está disponible para la mayoría de sistemas operativos UNIX.

Es capaz de capturar datos del tráfico que circula en una red de la cual es parte para posteriormente mostrarlos por salida estándar formateados de diferentes maneras.

La interfaz dotada por la librería `libpcap` soporta un mecanismo de filtrado BSD Packet Filter (BPF), gracias al cual se podrá analizar únicamente un subconjunto de paquetes definidos en el filtro.

Al igual que Wireshark esta herramienta es compatible con diferentes protocolos (IP, TCP, UDP,…), aunque en comparación, soporta menos protocolos que Wireshark, como por ejemplo el protocolo HTTP. Cabe mencionar que hay protocolos no soportados por tcpdump que se pueden llegar a capturar mediante esta herramienta gracias al filtrado de puertos, lo cual sí que permite, como por ejemplo el filtrado del puerto 80, el cual se relaciona con el tráfico HTTP.

Finalmente, se quiere citar que existe una herramienta versionada para sistemas operativos Windows cuyo nombre es WinDump.

2.1.2. Sistema de Detección de Intrusos

Un sistema de detección de intrusos, también conocido como *intrusion detection system* (IDS), es un sistema dotado de herramientas o dispositivos que permite monitorizar un entorno con la finalidad de detectar eventos inusuales, los cuales, en la mayoría de los casos, suelen ser desviaciones del comportamiento habitual siendo estos dañinos o inadecuados dentro del entorno en el que el IDS está implantado.

Dentro de la clasificación de los sistemas IDS, en el análisis de este proyecto, se recalcará el sistema de detección de intrusos en red, también conocido como *network intrusion detection system* (NIDS), debido a que este tipo de IDS son los que tienen la funcionalidad de analizar el tráfico de red.

No estará de más mencionar los sistemas de prevención de intrusos, también conocidos como *intrusion prevention system* (IPS). Los IPS son los IDS que aparte de tener la funcionalidad de monitorizar también disponen la funcionalidad de usar mecanismos de control, ejecutando así acciones sobre el evento detectado. Dentro de estos sistemas existen variantes como en los IDS, entre las cuales se recalcará los sistemas de prevención de intrusiones en red o *network intrusion prevention system* (NIPS).

Las diferentes alternativas de software libre analizadas en este apartado son las presentadas a continuación:

Snort [7]

Snort es un software de código abierto, actualmente publicado bajo la licencia GPLv2, cuyo desarrollo comenzó en 1998 por Martin Roesch. Se autodefine como un sistema de prevención de intrusiones en red capaz de realizar análisis de tráfico en tiempo real y generar registros de los datagramas IP.

Snort basa su funcionamiento en dos componentes principales, el primero, un motor de detección el cual contiene los algoritmos necesarios para hallar la anomalía; y segundo, un conjunto de reglas las cuales dictarán la acción a ejecutar.

Las reglas Snort pueden ser de dos tipos, reglas definidas y adaptadas a necesidades particulares, o reglas predefinidas disponibles dentro de la comunidad Snort.

Esta herramienta posee diferentes características de las cuales se querrán recalcar dos. La primera que está disponible para diferentes sistemas operativos UNIX y Windows; y la segunda, que aunque

posea una versión (Snort 3) *multithread,* la versión recomendada a usar no admite el uso de diferentes hilos para llevar a cabo las tareas (*multithreading*).

Figura 3: Logo Snort

Suricata [8]

Suricata es un motor de detección de amenazas en red de código abierto, desarrollado por Open Information Security Fundation en diciembre de 2009 y publicado bajo la licencia GPLv2.

Suricata sigue la idea básica de Snort con la principal diferencia del diseño, debido a que Suricata está diseñado para aprovechar las funcionalidades de las CPUs modernas con el fin de ofrecer un mayor rendimiento, como por ejemplo, mediante la repartición de los procesos entre los diferentes hilos o *threads*, lo que se conoce como el previamente mencionado *multithreading*.

Esta herramienta es compatible con el sistema de reglas que usa Snort, lo cual hace posible la reutilización de las reglas Snort desarrolladas durante años.

Entre las diferentes características de Suricata se podrá destacar que está disponible para diferentes sistemas operativos UNIX y Windows; que permite la extracción de archivos para el posterior análisis; y finalmente, que ofrece soporte para el lenguaje LUA con el objetivo de generar scripts y así poder detectar anomalías que mediante la sintaxis del conjunto de reglas no se podrían detectar.

Figura 4: Logo Suricata

2.1.3. Proxy a modo de MITM

Para un mejor entendimiento del concepto de proxy a modo de MITM como analizador de tráfico primero, se explicará que es cada uno de los términos, y posteriormente se enlazarán los conceptos.

Un proxy es un dispositivo que actúa a modo de intermediario entre las comunicaciones en red de dos nodos, siendo uno de los nodos el emisor y otro de los nodos el receptor.

Se denominará MITM a la técnica que consiste en interponerse entre la comunicación de dos nodos sin que ambos nodos tengan constancia de ello. Normalmente, este tipo de técnicas suelen relacionarse con ataques informáticos, puesto que el que se intercala en la comunicación tiene como objetivo alguna acción no consentida, como por ejemplo, llevar a cabo una escucha sin permiso, hacer una manipulación de las comunicaciones, etc.

Por consiguiente, un proxy a modo de MITM se podría definir como un dispositivo intermediario entre la comunicación de dos nodos en la cual se interpone para llevar a cabo alguna acción, para este proyecto en concreto, un análisis de tráfico.

A continuación se presentan las diferentes alternativas analizadas para la posible implantación en el proyecto.

mitmproxy [9]
Introducido por Aldo Cortesi en marzo de 2010 y actualmente figurando como autores del proyecto Aldo Cortesi, Maximilian Hils, Thomas Kriechbaumer, and contributors, mitmproxy es una herramienta de código abierto publicada bajo licencia MIT (licencia del Instituto Tecnológico de Massachusetts).

Esta herramienta, que opera mediante la consola de comandos, tiene las funcionalidades de un proxy HTTP/HTTPS permitiendo así interceptar, examinar y modificar el tráfico de una red de la cual es parte.

Dentro de las características de mitmproxy cabe destacar entre otras, aparte de las funcionalidades previamente mencionadas, que es una herramienta capaz de guardar conversaciones HTTP para su posterior análisis; que es capaz de realizar cambios en el tráfico HTTP mediante scripts escritos en Python; y que es una herramienta multiplataforma disponible para sistemas operativos Linux, Windows y Mac OS.

Según Aldo Cortesi [10], mitmproxy está dirigido a los desarrolladores de software y a los *pentesters* que necesitan manipular y monitorear de manera intensa el tráfico, debido a que tiene el control total sobre las solicitudes y las respuestas interceptadas, pudiendo editar los encabezados, el contenido y pudiendo cambiar los métodos de solicitud HTTP de estas. También permite establecer *sticky cookies* en las solicitudes, lo que significa que las solicitudes sin una cookie podrán heredar valores de cookies vistos anteriormente.

Figura 5: Logo mitmproxy

Burp Proxy [11]

Burp Proxy es una de las herramientas que ofrece la aplicación Burp Suite desarrollada por PortSwigger en junio de 2003, siendo un software de código cerrado, también conocido como software propietario.

Burp Suite es un programa que opera sobre un entorno gráfico el cual dispone de tres ediciones: una edición comunitaria que puede ser descargada gratuitamente, una edición profesional y una edición empresarial. Todas las ediciones están disponibles para diferentes sistemas operativos Linux, Windows y Mac OS X.

Como ya se ha presentado, dentro del programa se incluye la herramienta Burp Proxy, la cual tiene la funcionalidad de un servidor proxy web permitiendo así la interceptación, inspección y modificación del tráfico.

Figura 6: Logo Burp Suite

2.1.4. Solución propuesta para el analizador de tráfico

Entre las diferentes propuestas presentadas a la hora de escoger el analizador de tráfico, se ha tenido en cuenta, por una parte, las características de cada una de las soluciones, y por otra parte, los recursos de los que se dispone para poder ejecutar dichas soluciones.

El analizador de tráfico pretende ser ejecutado en una Raspberry Pi 3, con lo que a priori todas las soluciones son viables. Cabe mencionar que la Raspberry en este punto del proyecto no está provista de ningún sistema operativo, con lo que este se podrá adaptar dependiendo de la solución escogida.

Tomando como referencia las características de cada una de las soluciones, en primer lugar se ha descartado la solución de los *sniffers*, mencionados en el punto *2.1.1.Sniffer*, ya que no están dotados de la funcionalidad de generar alertas tras capturar los paquetes de red.

Entre las soluciones no descartadas, IDS y proxy a modo de MITM, se prevé que ambas podrán ser válidas para usarse como analizador de tráfico en este proyecto. El problema es que en este proyecto únicamente se hará uso de una única solución a la hora de implantar el analizador de tráfico, y a estas alturas, tal y como se han mencionado, existen dos posibles candidatos a solución final.

A fin de facilitar la elección de la solución final, la cual se implantará en el proyecto como analizador de tráfico, se ha decidido hacer una criba de herramientas presentadas por cada una de la soluciones no descartadas hasta el momento, dejando únicamente una herramienta por cada solución no descartada o candidato.

Posteriormente, tras la criba, se procederá a analizar las características de cada una de las herramientas mantenidas y las respectivas soluciones candidatas. Esta acción ayudará a escoger la herramienta y la solución definitiva que se usará en el proyecto como analizador de tráfico.

A continuación se presenta las herramientas mantenidas por cada una de las soluciones candidatas tras la criba.

Siendo Suricata la herramienta mantenida entre las herramientas propuestas para la solución candidata de los IDS (ver *2.1.2.Sistema de Detección de Intrusos*) por los siguientes motivos:

- Es compatible con las reglas Snort.
- Ofrece mayor rendimiento aprovechando las funcionalidades de las CPUs modernas.

Siendo mitmproxy la herramienta mantenida entre las herramientas propuestas para la solución candidata de los proxy a modo de MITM (ver *2.1.3.Proxy a modo de MITM*) por las siguientes razones:

- Es una herramienta de código abierto.
- Es capaz de realizar tratamientos de tráfico mediante scripts escritos en Python.
- Funciona desde la terminal, por lo general, esto equivale a un consumo de recursos menor.

Siendo válidas ambas soluciones candidatas, tal y como se ha recalcado previamente, se ha optado por usar mitmproxy como solución final para implantarlo como analizador de tráfico en este proyecto. La razón principal de la elección, es que esta herramienta es capaz de realizar tratamientos de tráfico mediante scripts escritos en un lenguaje más conocido como es el lenguaje Python.

Finalmente mencionar también el ímpetu del desarrollador del proyecto por el uso de técnicas MITM a modo de mitigación de ataques, a causa de la noción que este tiene sobre el uso de MITM únicamente para la realización de ataques y no en modo alguno de mitigación.

2.2. Topología de la red

La topología de una red se podrá definir como el modo en el que están distribuidos los nodos que forman dicha red. Se entenderá como nodo cualquiera de los diferentes dispositivos que son parte de la red.

En un proyecto de este estilo existen diferentes alternativas a la hora de hacer la elección de la topología de la red. Las alternativas han sido limitadas por las diferentes premisas que se dan en este proyecto en concreto:

- El entorno en el que este proyecto se desarrollará es un entorno doméstico, el cual tiene sus limitaciones, entre las cuales se destacarán por una parte, las limitaciones a la hora de la distribución de los dispositivos de la red, y por otra parte, la limitación de dispositivos de red disponibles.
- Como bien se ha definido en el punto anterior *2.1.4.Solución propuesta para el analizador de tráfico*, uno de los dispositivos a usar en este proyecto será un proxy a modo de MITM, el cual se encargará de analizar el tráfico.

Se plantearán dos tipos de topologías diferentes, las cuales se mostrarán a continuación, adaptadas a la segunda premisa presentada, pero diferenciadas por la primera premisa, en concreto, por la distribución de los dispositivos de la red.

2.2.1. Redireccionamiento del tráfico de red al proxy

Mediante esta topología de red se pretende que todo el tráfico generado desde la red interna (la red en la que se encuentran los dispositivos a proteger) con destino a Internet sea redireccionado hacia el proxy, con el fin de llevar a cabo el análisis del tráfico. Finalmente, y una vez llevado a cabo el análisis, el proxy será el encargado de direccionar el tráfico recibido hacia Internet. Este proceso se muestra en la imagen mostrada a posteriori (ver *Figura 7: Topología de red redirrecionamiento de tráfico hacia el proxy)*, en donde los dispositivos a proteger (los clientes) y el proxy están conectados al mismo punto de acceso.

Se han planteado dos soluciones para poder realizar la redirección:

- Configurar un dispositivo dentro de la red interna que se encargue de hacer el redireccionamiento del tráfico hacia el proxy, por ejemplo mediante un conmutador.
- Configurar los dispositivos a proteger indicándoles que deberán hacer uso de un servidor proxy para las conexiones externas de red.

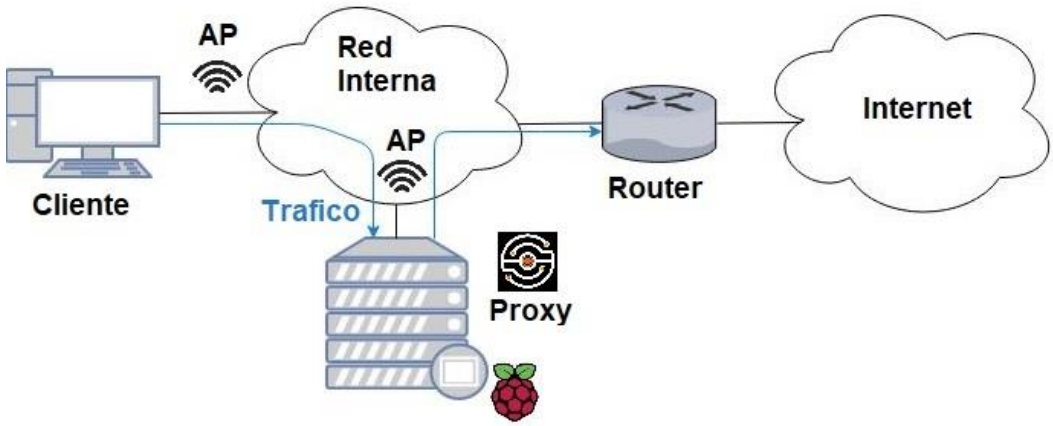


Figura 7: Topología de red redirrecionamiento de tráfico hacia el proxy

En la imagen se muestran los diferentes dispositivos que forman la topología:

- Los clientes o los dispositivos a proteger.
- La Raspberry con la funcionalidad de proxy.
- El encaminador o router.
- El enlace de conexión a la red interna, es decir, el punto de acceso.

Mencionar que aunque en la imagen anterior (ver *Figura 7: Topología de red redirrecionamiento de tráfico hacia el proxy*) no se muestre, pero como bien se ha mencionado anteriormente, cabría la posibilidad de hacer uso de diferentes dispositivos dentro de la red interna con funcionalidades de redireccionamiento.

Se han detectado una serie de ventajas e inconvenientes a la hora de usar esta topología.

Entre las ventajas detectadas se destacarán las siguientes:

- En caso de fallar algún dispositivo únicamente fallará la funcionalidad del mismo. Es decir, si únicamente falla la Raspberry, la cual aloja el proxy, los dispositivos a proteger no tendrán salida posible hacia Internet, en cambio podrán seguir funcionando en la red interna sin problema.

Entre los inconvenientes detectados se destacarán las siguientes:

- Se deberá configurar el redireccionamiento y los enlaces. En caso de ser necesario, se tendrán que añadir los dispositivos necesarios para conseguir las funcionalidades nombradas en este punto.
- Existen mayores dificultades a la hora de que la topología sea escalable en comparación con la topología que se presentará en el siguiente punto (ver *2.2.2.Proxy con funcionalidad de punto de acceso*). Esto se debe a que la configuración que requiere esta topología se centra en diferentes dispositivos.

2.2.2. Proxy con funcionalidad de punto de acceso

Mediante esta topología se pretende que todo el tráfico generado por los dispositivos a proteger se canalice directamente al proxy, así este podrá realizar el análisis del tráfico correspondiente. Una vez que el proxy haya efectuado el análisis de tráfico, será el mismo quien se encargue de direccionar el tráfico hacia el siguiente destino.

Para lograr los objetivos de esta solución se configurará el dispositivo que ejerce de proxy, la Raspberry, a modo de punto de acceso. Este tipo de topología se muestra en la siguiente imagen (ver *Figura 8: Topología de red proxy con funcionalidad de punto de acceso*).

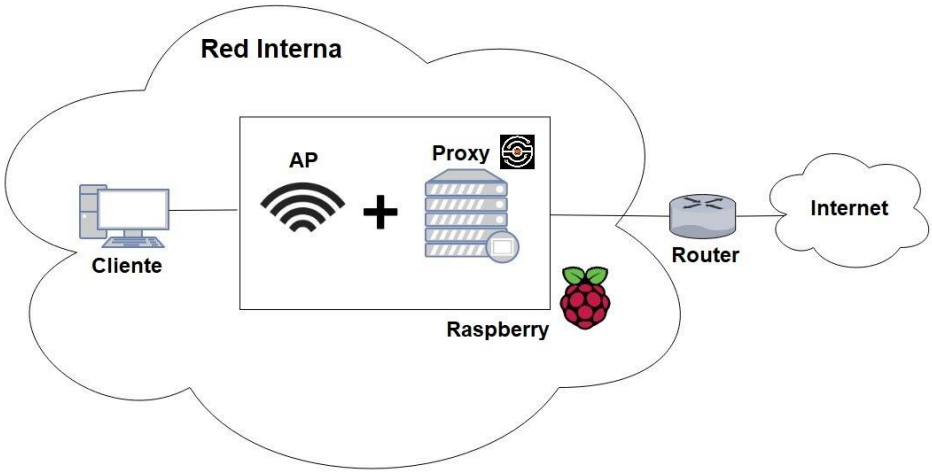


Figura 8: Topología de red proxy con funcionalidad de punto de acceso

En la imagen se muestran los diferentes dispositivos que forman la topología:

- Los clientes o los dispositivos a proteger.
- La Raspberry a modo de proxy y a modo de punto de acceso.
- El encaminador o router.

Las ventajas e inconvenientes detectados a la hora de usar esta topología se mostrarán a continuación.

Entre las ventajas detectadas se destacarán las siguientes:

- En comparación con la topología anterior, se necesitarán menos dispositivos en la red, o mejor dicho, las configuraciones se limitarán a un único dispositivo, la Raspberry.
- La escalabilidad de esta topología, esto se debe a que la configuración solo se centra en un único dispositivo.

Entre los inconvenientes detectados se destacarán las siguientes:

- Debido a las limitaciones en el *hardware* que tiene una Raspberry es posible que el rendimiento se vea penalizado y que las probabilidades de que ocurran desbordamientos y denegaciones de servicio serán mayores.
- Las funcionalidades se alojan en un único dispositivo, así pues, si el dispositivo falla los dispositivos se quedarán sin acceso a la red, tanto interna como externa, aparte de que el proxy preparado para analizar el tráfico dejará de funcionar.

2.2.3. Solución propuesta para la topología de red

Teniendo en cuenta las características de las diferentes topologías de red presentadas, se escogerá la topología que implementa el punto de acceso en el dispositivo que ejerza de proxy (ver *2.2.2.Proxy con funcionalidad de punto de acceso*). Esta elección se debe a los principales motivos que se presentarán a continuación.

Por una parte se intuye que las desventajas presentadas en la implementación escogida dentro del entorno en el que se desarrollará el proyecto no ocurrirán, puesto que se prevé que el tráfico generado para las pruebas que se realizarán no superará las cotas para que se den casos de desbordamientos, denegaciones de servicio, etc.

Por otra parte se prevé que para desarrollar este proyecto será más fácil configurar un único dispositivo (la Raspberry a modo de proxy y a modo de punto de acceso) que no estar configurando diferentes dispositivos como un proxy en la Raspberry, el redireccionador de tráfico, etc.

2.3. Punto de acceso

Un *access point* (AP) o punto de acceso es un dispositivo encargado de generar un enlace inalámbrico el cual habilitará la interconexión entre diferentes dispositivos. La mayoría de los AP, incluido el de este proyecto, tienden a usar estándares IEEE 802.11.

A la hora de explorar las posibles vías para implantar el punto de acceso se han tenido en cuenta las soluciones que se han ido planteando a lo largo del documento, así pues, tal y como recoge la topología planteada en el punto *2.2.3.Solución propuesta para la topología de red*, la Raspberry que ejerza de proxy tomará la funcionalidad de punto de acceso.

El propio fabricante de la Raspberry ofrece una solución para añadirle la funcionalidad de punto de acceso al dispositivo [12]. Confiando en el fabricante, se optara por el uso de dicha solución, ya que se presupone que durante el desarrollo del proyecto en caso de que se produzca algún tipo de dificultad se podrán obtener soluciones dentro de la extensa comunidad de Raspberry.

El prerrequisito mínimo para implantar esta solución es que la Raspberry deberá soportar conexiones inalámbricas de red y conexiones Ethernet. La Raspberry Pi 3 que se usará para el proyecto soporta las conexiones mencionadas, encima, es el modelo de Raspberry usado en el documento del fabricante.

Tras cumplir con el prerrequisito mínimo también se necesitará una serie de software instalado el cual se recoge en el siguiente punto *2.3.1.Raspberry con funcionalidad de punto de acceso*.

Cabe mencionar que a la hora de estudiar otras vías se ha explorado otra solución posible, en caso de no obtener el resultado esperado con la solución propuesta se podrán intentar sacar adelante ciertos objetivos del proyecto mediante esta solución.

La solución alternativa consiste en el uso de un punto de acceso de carácter común, y usado a diario, como es un encaminador. Este tipo de dispositivos, en concreto el modelo TL-WR740N de TP-LINK disponible, dotan de un firmware propio, con lo que se ha estudiado la opción de integrarle el firmware OpenWrt. El firmware propuesto está basado en un sistema operativo conocido como es Linux, con el propósito de a la hora de desarrollar el proyecto (en caso de necesitar configurar, instalar,... algún tipo de software) poder tener facilidades, cosa que a priori el modelo TL-WR740N de TP-LINK disponible admitiría.

2.3.1. Raspberry con funcionalidad de punto de acceso

Para que una Raspberry pueda funcionar como punto de acceso se necesitará tener instalado:

- Un programa que le permita al dispositivo actuar a modo de AP.
- Un programa que permita al dispositivo actuar de servidor DHCP con el fin de asignar IPs de una manera dinámica.

Siguiendo las recomendaciones del fabricante de la Raspberry se ha escogido el software que se presentará a continuación para cada una de las funcionalidades.

Hostapd [13]

Hostapd es un proceso demonio o *daemon* que permite actuar a las tarjetas de red del dispositivo en el cual este se ejecuta a modo de punto de acceso.

Desarrollado por Jouni Malinen, y actualmente bajo licencia BSD, esta herramienta de código abierto se puso a disposición como primera versión estable en julio de 2004.

La herramienta incorpora la implementación de los protocolos estándar que debe tener un AP, gracias a los cuales permitirá llevar a cabo las funcionalidades asignadas a un punto de acceso. Dentro de las diferentes funcionalidades se destacará, el envío de paquetes baliza o *beacon frames* para hacer saber la existencia del punto de acceso; el uso de diferentes protocolos con el fin de poder establecer conexión con los clientes; y finalmente, el poder codificar y decodificar las comunicaciones, siempre y cuando este la opción establecida.

Dnsmasq [14]

Esta herramienta desarrollada por Simon Kelley en 2001, y publicado bajo la licencia GPLv2/GPLv3, proporciona un infraestructura de red pensado en pequeñas redes por su diseño para ser ligero y de tamaño pequeño. Proporciona diversas funcionalidades, como pueden ser las de servidor DNS y servidor DHCP, para sistemas con recursos limitados.

Dentro de las características del servidor DHCP se recalcará que soporta diferentes versiones del protocolo DHCP (DHCPv4 Y DHCPv6); asignaciones de direcciones estáticas; y finalmente, soporta múltiples redes.

Mencionar que para cada protocolo DHCP ofrece diferentes características concretas, como por ejemplo, para el protocolo DHCPv4 permite nombrar a los clientes en uso.

2.4. Tratamiento del tráfico

Si tras el análisis de tráfico efectuado se prevé que surgirá algún futuro riesgo se hará saber a través de la generación de alertas. Estos futuros riesgos se identificarán gracias a la obtención del destino del tráfico analizado, puesto que no es descartable que dicho tráfico pueda emprender rumbo hacia algún servidor identificado como malicioso. Este tipo de escenarios se deberán a consecuencia de acciones, normalmente ilegales, llevadas a cabo en el dispositivo que generará el tráfico, como por ejemplo la instalación de algún malware, etc.

Como se ha definido previamente (ver *2.1.4.Solución propuesta para el analizador de tráfico*), la solución que se implantará para realizar el análisis del tráfico será la herramienta mitmproxy.

Teniendo en cuenta la anterior premisa, en este apartado se presentará la estrategia escogida para el desarrollo del tratamiento del tráfico. La base de la estrategia escogida será aprovecharse de una de las funcionalidades que ofrece mitmproxy, la cual permitirá la ejecución de un script escrito en Python. Para ser más exactos, la ejecución hará uso de un script generado específicamente para el desarrollo de este proyecto, el cual se ejecutará desde la Raspberry.

La generación del script se realizará, por una parte, analizando los diferentes aspectos necesarios para su posible desarrollo, y por otra parte, mediante una descomposición de escritura del mismo. Todo ello se presentará en los apartados mostrados a continuación.

2.4.1. Características de la generación de alertas

En este punto se presentaran las características que se tendrán en cuenta para la generación de alertas, las cuales se tendrán en consideración cuando se vaya a generar el script. Se prevé que estas características serán viables a la hora de aplicarse dentro del entorno planteado hasta ahora.

La primera característica que se considerará es la rapidez con la que se generarán las alertas, puesto que entre que menos tiempo se tarde a la hora de generar la alerta, más margen se tendrá para actuar en caso de que se detecte cualquier acción que pueda suponer un peligro potencial.

Otra característica que se tendrá en cuenta es la disponibilidad de las alertas generadas. Esto se traducirá en la facilidad con la que se podrá consultar dichas alertas. Todo esto hablando dentro de un marco seguro y con permisos limitados hacia los elementos/personas que puedan consultar o tratar dichas alertas.

Finalmente la última característica que se destacará es la facilidad con la que se podrán interpretar las alertas. Dentro de la mencionada característica se remarcará que las alertas generadas deberán poseer un mismo esquema y unos mismos elementos identificativos, los cuales ayudarán a entender la generación de la misma. Los elementos mínimos que deberán poseer las alertas serán los siguientes:

- Una marca de tiempo de cuando se ha generado la alerta;
- Quien o que ha sido el generador de la alerta.

2.4.2. Alternativas para la visualización de alertas

En este apartado se pretenderá mostrar la tecnología o método que se usará (junto al script escrito en Python) a lo largo del proyecto con el fin de visualizar las alertas que se generarán.

Para ello, primero se realizará un estudio de diferentes alternativas válidas y disponibles para usar con el script que se generará para el desarrollo de este proyecto. Entre las diferentes alternativas se destacarán las siguientes:

- Creación y actualización de archivos de registro, o dicho de otra manera, uso de archivos log.
- Envío de correos electrónicos.
- Envío de mensajes mediante el uso de la mensajería instantánea.

Si bien es cierto que el escenario ideal será el uso de todas las alternativas o la combinación de algunas, siempre y cuando se dispongan recursos para ello, para el desarrollo de este proyecto únicamente se escogerá una única alternativa.

La alternativa que se preferirá no es otra que la del envío de mensajes haciendo uso de la mensajería instantánea. Esta alternativa se ha escogido por dos principales motivos:

- La posible integración de este tipo de tecnologías en un script escrito en Python.
- La característica de la disponibilidad, puesto que hoy en día, y teniendo en cuenta el escenario en el que el proyecto está desarrollado, quien no dispone a mano de un dispositivo móvil con alguna aplicación de mensajería instantánea.

Cabe mencionar que en lo que respecta a las otras dos características, presentadas en el anterior punto (ver *2.4.1.Características de la generación de alertas*), haciendo uso de la mensajería instantánea se podrá afirmar lo siguiente:

- La rapidez con la que la alerta se genera mediante el uso de la misma es más que aceptable;
- La facilidad con la que se podrá interpretar la alerta, dependerá de lo escrito en el script, disponiendo del control para lograr la interpretación más sencilla posible.

2.4.3. Elección de la herramienta para la visualización de alertas

Tal y como se ha especificado en el anterior punto (ver *2.4.2.Alternativas para la visualización de alertas*), para poder ver las alertas se hará uso de alguna aplicación de mensajería instantánea junto al script escrito en Python. A día de hoy existen dos aplicaciones de mensajería instantánea que predominan en el mercado, la primera WhatsApp y la segunda Telegram.

WhatsApp [15]

WhatsApp es una aplicación multiplataforma de mensajería instantánea desarrollada por Jan Koum y Brian Acton en el año 2009, la cual está publicada bajo licencia de software propietario.

Las características que se destacarán de WhatsApp para el posible uso en el desarrollo del proyecto son las siguientes:

- El uso de grupos y contactos individuales para el envío/recepción de mensajes por parte de la aplicación.
- El envío de diferentes tipos de mensajes (mensajes de texto, videos, audios, etc.).
- El uso posible en diferentes plataformas.
- El uso de cifrado de extremo a extremo en los mensajes.
- La posible integración de la aplicación en scripts.

Telegram [16]

Telegram es una aplicación multiplataforma de mensajería instantánea desarrollada por los hermanos Nikolai y Pavel Durov en el año 2014 y publicada bajo licencia bajo la licencia GPLv2/GPLv3.

Entre las múltiples características que ofrece Telegram dentro del marco del desarrollo de este proyecto se recalcarán las siguientes:

- El uso de diferentes elementos para el envío/recepción de mensajes por parte de la aplicación (canales, grupos, bots,etc.).
- El envío de diferentes tipos de mensajes (mensajes de texto, videos, audios, etc.).
- El uso posible en diferentes plataformas.
- El uso de cifrado de extremo a extremo en los mensajes.
- La posible integración de la aplicación en scripts.
- Las facilidades que proporciona hacia los usuarios para el cumplimiento del RGPD.

NOTA: Dispone de muchas más características las cuales se podrán consultar en la página web de las preguntas frecuentes sobre Telegram [2].

Tras la información recopilada de ambas aplicaciones de mensajería instantánea y tras obtener conocimiento de que ambas podrán ser integradas en un script escrito en Python, la aplicación que se usará en el desarrollo de este proyecto será Telegram por dos motivos principales:

1. La existencia de alternativas a la hora de integrar Telegram en el script en las que no se proporcionará ningún tipo de dato personal, como puede ser un teléfono móvil.
   Por mencionar a las alternativas exploradas haciendo uso de WhatsApp en un script escrito en Python, todas ellas requerían de un número de teléfono para poder generar algún mensaje que contuviese la alerta.
2. El desarrollador del proyecto tiene mayor experiencia y se dispone de mayor documentación de scripts escritos en Python para el uso de la aplicación Telegram.

2.4.4. Visualización de alertas

Tras definir en el anterior punto (ver *2.4.3.Elección de la herramienta para la visualización de alertas*) la herramienta que se usará para la visualización de alertas, se procederá a explicar el esquema que se deberá seguir para poder visualizar las alertas.

Primero se crearán los elementos necesarios para visualizar las mencionadas alertas. Se distinguirán dos elementos:

- Un canal de Telegram;
- Un bot de Telegram.

El canal de Telegram, es el elemento de Telegram que se usará para poder visualizar las alertas. A fin de proporcionar mayor seguridad, este canal deberá ser privado. Gracias a esto, únicamente deberá tener acceso al mismo los componentes que tengan permiso, ya sean procesos, personas, etc.

El bot de Telegram se encargará de generar las alertas que se mandarán al canal. Las alertas las originará mediante mensajes automatizados los cuales se generarán en el script de Python. Para la creación de los mensajes hará uso de la interfaz de programación para bots basada en HTTP `TelegramBot API` ofrecida por Telegram.

Una vez se tengan los elementos necesarios creados se integrará el bot dentro del script desarrollado para este proyecto. Para ello se hará uso de la librería Python `telepot`, de la cual se ofrecerá una breve explicación al final del apartado.

Tras tener los elementos generados, integrar el bot en el script y poner el script en funcionamiento con mitmproxy, se generarán las alertas enviándolas al canal privado cuando se detecte que se haya generado tráfico que pueda suponer un peligro potencial.

Finalmente, una vez generada la alerta, Telegram avisará al usuario que el canal dispone mensajes nuevos (las alertas) mediante las notificaciones generadas por la propia aplicación.

Telepot [17]

Telepot es una librería Python desarrollada por Nick Lee y publicada bajo licencia MIT, la cual ayuda a generar funcionalidades para la interfaz `Telegram Bot API`, proporcionando la programación y gestión del código de bots de Telegram en el lenguaje Python.

2.4.5. Reputación IP/URL

Este será otro de los aspectos que se deberá tratar cuando se desarrolle el script. La reputación se podrá definir como el prestigio o valoración que se tendrá de una dirección, vinculado a la confianza de la misma (ya sea de un modo positivo o negativo).

Gracias a este aspecto se podrá tener la noción de cuáles podrán ser las direcciones de servidores maliciosos a la hora de generar las alertas, una vez se haya analizado el tráfico y habiendo obteniendo el destino del mismo.

Las reputaciones suelen estar recogidas y clasificadas en diversas fuentes tales como archivos, bases de datos, páginas web, etc. La clasificación se hará bien por conocimiento propio o bien por información obtenida de terceros, a la cual se le da credibilidad (generalmente porque son entidades en las que se puede confiar).

A la hora de clasificar las direcciones se suele tener en cuenta la confianza de las mismas, pudiendo así generar fuentes de direcciones con mala reputación (conocidas como *blacklist*) o fuentes de direcciones con buena reputación (conocidas como *whitelist*). Estas fuentes se conocerán como listas de reputación.

En este proyecto las listas de reputación se usarán dentro del script para obtener las direcciones sospechas y poder contrastarlas con las direcciones de destino del tráfico analizado, dicho de otra manera, se hará uso de diferentes *blacklist* para determinar si el destino del tráfico analizado es lícito. Las listas de reputación se obtendrán de terceros, aunque también se fijará la posibilidad de generar listas propias.

Entre las listas de reputación emitidas por terceros se recalcarán las siguientes:

- Listas de reputación emitidas por la empresa Bambenek Consulting. Estas listas de reputación [3] recogen direcciones IPs las cuales las etiquetan como maliciosas y las clasifican en diferentes categorías.
- Lista de reputación recogida en un archivo emitido por AlienVault. El archivo [4] contiene un listado de direcciones IP consideradas maliciosas por la propia entidad.
- Lista de reputación ofrecida por Emerging Threats división que pertenece a la empresa Proofpoint, Inc. Esta lista de reputación [5] está publicada en la web y recoge un listado de direcciones IP consideradas maliciosas por la propia entidad.
- Listas de reputación emitidas en el portal de badips.com. Estas listas de reputación recogen direcciones IPs etiquetadas como maliciosas y clasificadas en diferentes categorías, las cuales se obtendrán mediante el uso de una API [6] que ofrece el propio portal.

Teniendo en cuenta el tiempo estimado para el desarrollo del proyecto, en un principio se optará por usar en el script que se desarrollará listas de reputación emitidas por la empresa Bambenek Consulting.

2.4.6. Diseño del script

En este punto se detallará la estructura del script en Python que se generará para el desarrollo de este proyecto, gracias al cual se obtendrán las alertas necesarias que permitan identificar conexiones a servidores informados como vulnerables, maliciosos o susceptibles de haber sido comprometidos, tras llevar a cabo un previo tratamiento del tráfico de red.

Antes de definir la estructura se detallará el flujo seguido por el script. El script recogerá información del tráfico de red analizado, dentro de la cual se podrá obtener el destino del mismo. Este destino se comparará con la lista de IPs/URLs sospechosas definidas en el script, las cuales se han obtenido de diferentes listas de reputación y estarán actualizadas cada cierto periodo de tiempo. Únicamente si el destino del tráfico coincide con algún valor que contenga la mencionada lista, se generará una alerta mediante el bot de Telegram integrado en el script, el cual enviará un mensaje al canal privado de Telegram. Todo el flujo se recoge en la siguiente imagen *Figura 9: Esquema para el tratamiento del tráfico*.

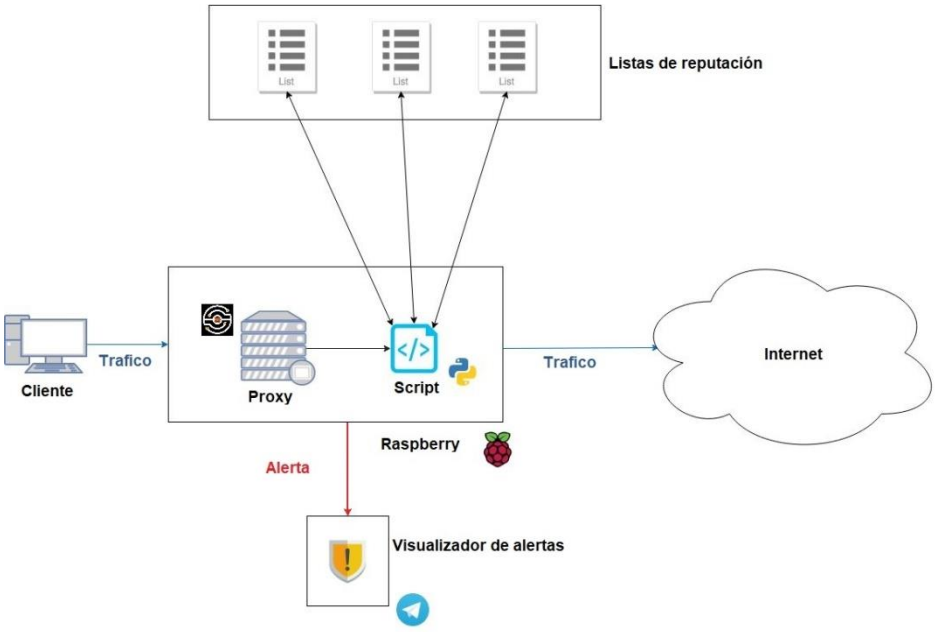


Figura 9: Esquema para el tratamiento del tráfico

A continuación se definirá la estructura del script, la cual se dividirá en tres componentes principales:

- Los módulos Python necesarios.
- Las variables globales.
- Las funciones que se crearán para la generación de alertas.

Dentro de los módulos Python necesarios que se usarán, como mínimo, se prevé que se deberán definir los siguientes módulos:

- El submódulo `http` integrado en el módulo Python `mitmproxy`, proporcionado por la aplicación mitmproxy, gracias al cual se podrá hacer uso de los eventos [7] definidos y tratar el tráfico analizado por la propia herramienta.
- El módulo `telepot`, el cual servirá para generar las alertas en Telegram.
- El módulo `urllib.request`, el cual proveerá de funcionalidades para poder obtener información usando URLs, en este caso, para obtener los valores de las listas de reputación alojados en URLs de terceros.

En lo que a las variables globales se refiere, como mínimo, se prevé que se hará uso de diferentes variables para inicializar los siguientes valores:

- Los valores de las consideradas IPs y URLs sospechosas.
- El valor del bot de Telegram, el cual será responsable de la generación de alertas.

Finalmente se definirán las funciones que se usarán para el proceso de generación de alertas. Se prevé que como mínimo se hará uso de las siguientes funciones:

- La función principal, la cual se encargará de obtener la información respectiva del tráfico analizado para posteriormente llevar a cabo el tratamiento del mismo, y generar la alerta si esta fuese necesaria, en base a las condiciones definidas.
- Una función que se encargará de mantener actualizadas las listas de IPs/URLs sospechosas definidas en el script. Las listas IPs/URLs sospechosas definidas se basarán en las listas de reputación.

3. Fase de desarrollo

En este capítulo se mostrarán las diversas soluciones implantadas a lo largo del desarrollo del proyecto.

3.1. Preparación de la Raspberry

Este punto recogerá toda la configuración necesaria de la Raspberry Pi. Teniendo en cuenta lo definido en la anterior fase (ver 2.*Fase de investigación*) la Raspberry realizará labores de proxy a modo de MITM, de punto de acceso y se encargará de tratar el tráfico. Para desarrollar las labores a realizar previamente, se dejará el dispositivo preparado.

Los materiales usados en este apartado son los siguientes:

- Una Raspberry Pi 3 modelo B.
- Fuente de alimentación para la Raspberry.
- Una tarjeta de memoria SanDisk Ultra para la Raspberry.
- Monitor HDMI para visualizar el contenido o lo desarrollado en la Raspberry.
- Cable HDMI para conectar la Raspberry con el monitor HDMI.
- Teclado para ejecutar acciones en la Raspberry.
- Encaminador que proveerá de conexión a Internet.
- Conexión a Internet.

3.1.1. Sistema operativo

Se deberá elegir el sistema operativo bajo el cual operará la Raspberry, para ello se deberán tener en cuenta ciertos puntos a la hora de hacer la elección del sistema operativo.

El primero de los puntos es las compatibilidades exigidas por los diferentes programas que se usarán a la hora de realizar el desarrollo del proyecto. Las implantaciones de software que se emplearán a lo largo del desarrollo son bastante flexibles, ya que permiten ser desplegadas en diferentes sistemas operativos. En este punto no habría limitaciones para hacer la elección del sistema operativo.

Sin perder de vista el primer punto, el segundo punto que se tendrá en cuenta es el manejo o la facilidad con la que el desarrollador se desenvuelve en el sistema operativo elegido. En este caso se ha escogido un sistema operativo Debian, o mejor dicho, un sistema operativo basado en Debian el cual se presentará en el siguiente párrafo.

El tercer y último punto que se ha tenido en cuenta es la adaptación de dicho sistema operativo a la Raspberry Pi. Existe un tipo de sistema operativo libre, recomendado por la propia fundación Raspberry Pi, basado en Debian y optimizado para el hardware de las Raspberry conocido como Raspbian [18], que es el sistema operativo que se usará para este proyecto.

Así pues, tras escoger el sistema operativo a usar se obtendrá una imagen de este para instalarlo. Entre las diferentes imágenes que ofrece Raspbian para este proyecto se escogerá la versión Raspbian Buster Lite de Febrero de 2020 obtenido de la página web oficial de Raspberry [8]. Dicha versión se escogerá por dos motivos, el primer motivo, para no tener instalado en la Raspberry software innecesario; y el segundo motivo, para evitar consumir recursos innecesariamente por culpa de la GUI, ya que una vez que este el dispositivo listo este solo dependerá de un cable de red y el cable de corriente.

Tras obtener la imagen de Raspbian se procederá a su instalación. La instalación se realizará en una tarjeta SD, en este proyecto la tarjeta SanDisk Ultra usada tiene una capacidad de 16GB.

Para la integración del sistema operativo en la tarjeta SD se hará uso de un programa que permita escribir imágenes de sistemas operativos en dispositivos USB, en este caso el programa escogido es Win 32 Disk Imager [9], tal y como se muestra en la siguiente imagen (ver *Figura 10: Win 32 Disk Imager*).

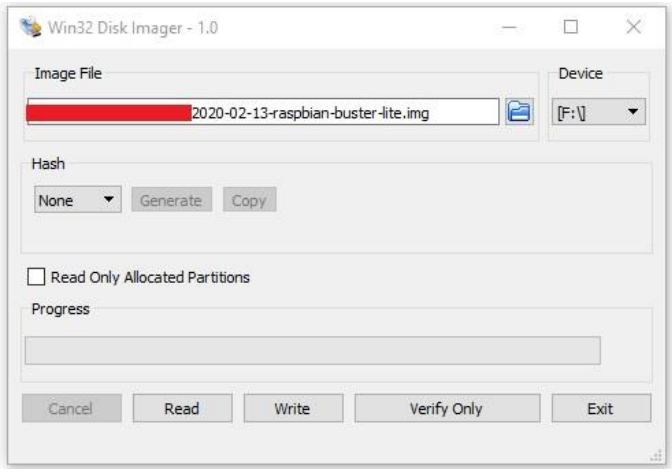
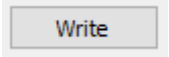


Figura 10: Win 32 Disk Imager

Los pasos a seguir en el programa Win 32 Disk Imager son los siguientes:

- Se seleccionará la imagen descargada pulsando en el icono de la carpeta. 
- Se seleccionará el dispositivo donde se hará la escritura en la *select-box* del apartado *Device*.
- Finalmente se pulsará el botón para ejecutar la escritura. 

Para concluir se añadirá la tarjeta SD a la Raspberry. En este punto ya estará la Raspberry dispuesta del sistema operativo Raspbian.

3.1.2. Acceso a la Raspberry Pi

Se arrancará la Raspberry, para ello será necesario conectarla a una fuente de alimentación con conector micro USB. En este proyecto se usará el modelo Aukru micro USB con interruptor para Raspberry (5 V, 3000 mA).

Al ser el primer arranque, para poder hacer la configuración necesaria se le tendrá que conectar a la Raspberry un teclado (en este caso el modelo ha sido un Logitech K400) a uno de los puertos USB, y un monitor (en este caso el modelo DELL Professional P2319H) mediante un cable HDMI a la salida HDMI de la Raspberry.

Se accederá al sistema operativo con el par de usuario/clave que tiene configurado por defecto.

Usuario: `pi`

Clave: `raspberry`

Una vez se ha accedido se recomendará cambiar la contraseña usando el siguiente comando:

```
$ passwd
```

Se recomendará modificar también la contraseña del administrador mediante el siguiente comando:

```
$ sudo passwd
```

NOTA: La contraseña que se introducirá para permitir ejecutar el comando con permisos de administrador (mediante `sudo`), será la misma contraseña del usuario con la que se ha iniciado sesión.

3.1.3. Cambio de idioma, distribución del teclado y zona horaria

La versión del sistema operativo obtenida viene configurada por defecto en inglés, lo cual dificulta el manejo a la hora de continuar con el desarrollo, puesto que entre otras cosas, hace uso de un diseño ingles en la distribución del teclado.

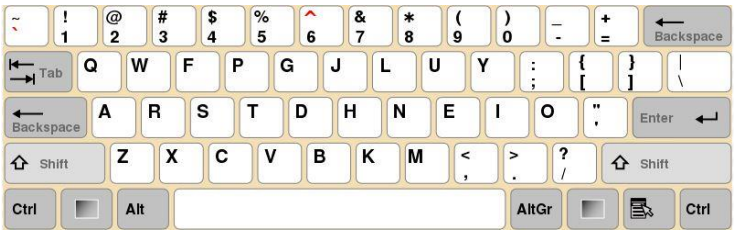


Figura 11: Ejemplo de teclado con distribución inglesa

Primero se modificará la codificación del idioma mediante los siguientes pasos:

- Se ejecutará el siguiente comando:

```
$ sudo dpkg-reconfigure locales
```

- Entre las diferentes opciones disponibles se escogerá, mediante la barra espaciadora, la codificación y el idioma. En este caso se elegirá el español de España con codificación UTF-8, siendo la opción `es_ES.UTF-8 UTF-8`.
- Se volverá a seleccionar la opción escogida para posteriormente generar los locales.

Una vez se haya cambiado la codificación y el idioma se modificará la distribución del teclado:

- Se ejecutará el siguiente comando:

```
$ sudo dpkg-reconfigure keyboard-configuration
```

- Posteriormente se escogerá el modelo de teclado, en este caso se ha escogido el modelo de teclado `PC genérico 105 teclas (intl)`.
- Se seleccionará la distribución del teclado, en este caso una distribución española. Para ello se hará la elección de la opción `Español`.
- Se elegirá en la configuración del teclado la opción `Distribución por omisión del teclado`.
- Siguiendo con la configuración del teclado finalmente se escogerá la opción `Sin tecla modificadora`.

Para concluir con este punto se modificará la zona horaria:

- Se ejecutará el siguiente comando:

```
$ sudo dpkg-reconfigure tzdata
```

- Se escogerá el continente, en este caso la opción `Europa`.
- Finalmente, se escogerá la zona horaria. Para este proyecto se hará uso de la opción `Madrid`.

3.1.4. Configuración WiFi

A lo largo del desarrollo del proyecto se necesitarán instalar y actualizar ciertos programas. Para llevar a cabo ambas acciones la Raspberry requerirá de acceso a Internet. Esta conexión a Internet se hará vía WiFi y a continuación se describirá los pasos seguidos.

Antes de empezar la configuración cabe comentar que en este punto se conocía el Service Set IDentifier (SSID) de la red WiFi o el identificador de la red inalámbrica, con su respectiva contraseña.

Se deberá editar el fichero `/etc/wpa_supplicant/wpa_supplicant.conf` añadiendo los datos necesarios en el formato especificado para poder conectarse a la WiFi. Los datos que como mínimo habrá que añadir son el SSID de la red inalámbrica y su clave de acceso.

A la hora de editar el fichero la clave de acceso podrá ser encriptada como un número hexadecimal de 32 bytes, entendiendo que esta opción es mejor que guardarla en texto plano. Para ello se hará uso del siguiente comando:

```
$ sudo wpa_passphrase NombreDeLaRedWiFi ContraseñaDeLaRedWiFi >>
/etc/wpa_supplicant/wpa_supplicant.conf
```

La ejecución de comandos presentada generará los datos relativos a la red con el formato necesario y los añadirá al final del archivo. Dichos datos tendrán el siguiente aspecto:

```
network={
  ssid="NombreDeLaRedWiFi"
  #psk="ContraseñaDeLaRedWiFi"
  #Ejemplo de contraseña de la red wifi encriptada encriptada
  psk= 131e1e221f6e06e3911a2d11ff2fac9182665c004de85300f9cac208a6a80531

}
```

Se borrará la línea comentada que recoge el dato `psk`, puesto que es la contraseña en texto plano. Posteriormente se aplicarán los cambios reiniciando la Raspberry con el siguiente comando:

```
$ sudo reboot
```

NOTA: Antes de iniciar la configuración, puede darse la posibilidad de que haya que habilitar la tarjeta de red inalámbrica.

3.1.5. Actualización de los paquetes de Raspbian

Se actualizarán los paquetes disponibles en el sistema operativo Raspbian, para ello se usarán los siguientes comandos:

```
$ sudo apt-get update
$ sudo apt-get upgrade
```

3.1.6. Configuración SSH

Para poder acceder remotamente a la Raspberry se necesitará habilitar la funcionalidad Secure Shell (SSH). Será necesario debido a las limitaciones que se tiene a la hora de distribuir los equipos para llevar a cabo el desarrollo de este proyecto, principalmente, por la posición física en la que se encontrará la Raspberry y la máquina del desarrollador, teniendo como referencia la topología presentada anteriormente (ver *2.2.3.Solución propuesta para la topología de red*).

En este caso, el propio sistema operativo dispone de la funcionalidad SSH, lo único que habrá que hacer es activar dicha funcionalidad. Para ello se seguirán los pasos que se indican a continuación:

- Se ejecutará el siguiente comando:

```
$ sudo raspi-config
```

- Dentro del menú se seleccionará mediante la tecla *enter* la opción `Interfacing Options`.
- Posteriormente se seleccionará mediante la tecla *enter* la opción `SSH`.
- Se escogerá que se quiere activar el SSH, para posteriormente aceptar el mensaje que mostrará el estado en el que se encuentra la funcionalidad SSH.
- Finalmente, se deberá volver a visualizar el menú obtenido tras la ejecución del comando `raspi-config`, punto en el cual se escogerá la opción `Finish`.

3.1.7. Instalación pip3

Algunos paquetes expuestos desde los diferentes repositorios usados por el sistema operativo Raspbian pueden estar desactualizados, incluso a veces, esos paquetes pueden llegar a no existir, con lo que no estará de más buscar otras vías para llevar a cabo una obtención de paquetes.

Para el desarrollo de este proyecto se instalará la herramienta `pip`*`X`*, la cual permitirá instalar paquetes desde el repositorio de Python.

Primero se tendrá que tener en cuenta si alguna versión de Python está instalada en la Raspberry, puesto que de no ser así, se deberá instalar Python en el dispositivo. En este caso, el sistema operativo Raspbian dispone de las versiones `Python 2.7.16` de Python 2 y `Python 3.7.3` de Python 3. Se comprobará mediante los siguientes comandos:

```
$ python --version
$ python3 --version
```

Se instalará la herramienta `pip`*`X`* para la versión 3 de Python, es decir, se instalará la herramienta `pip3` mediante el siguiente comando:

```
$ sudo apt install python3-pip
```

3.2. Implantación del punto de acceso

Se implantará la funcionalidad de punto de acceso en la Raspberry Pi. Para ello, y siguiendo la topología presentada en el punto previo (ver *2.2.2.Proxy con funcionalidad de punto de acceso*), previamente se conectará la Raspberry al encaminador mediante un cable de red Ethernet. Este encaminador es el mismo que proporciona la red WiFi configurada en el anterior punto (ver *3.1.4.Configuración WiFi*). Gracias a esta acción la Raspberry obtendrá acceso a Internet mediante conexiones de red Ethernet (`eth0`), en vez de usar conexiones de red inalámbricas (`wlan0`).

Los materiales usados en este apartado son los siguientes:

- La Raspberry Pi modelo 3 B con la tarjeta de memoria SanDisk Ultra previamente configurada.
- Fuente de alimentación para la Raspberry.
- Cable RJ-45 para conectar la Raspberry y el encaminador.
- Un cliente para probar la implementación, en este caso un dispositivo portátil HP Pavilion 15 Notebook PC con sistema operativo Windows 10.
- Encaminador que proveerá de conexión a Internet.
- Conexión a Internet.

NOTA: Cabe mencionar que a lo largo de este punto se accederá a la Raspberry vía SSH mediante el usuario `pi`.

3.2.1. Preparación del entorno

Para implementar la funcionalidad de punto de acceso se necesitarán las herramientas Hostapd y Dnsmasq, como bien se ha descrito en el punto *2.3.1.Raspberry con funcionalidad de punto de acceso*.

Las herramientas se instalarán mediante los siguientes comandos:

```
$ sudo apt-get install hostapd
$ sudo apt-get install dnsmasq
```

En este desarrollo, el punto de acceso que se implantará deberá disponer de conexión a Internet, pudiendo así los dispositivos que se conecten al mismo disponer de salida a Internet. El esquema que se seguirá es el siguiente: El punto de acceso dotará de red a los dispositivos a proteger mediante la interfaz de red de conexión inalámbrica (`wlan0`), mientras que dispondrá de acceso a Internet gracias a la interfaz de red Ethernet (`eth0`), la cual estará conectada mediante un cable de red RJ-45 a un encaminador con acceso a Internet.

En los siguientes puntos se editarán las configuraciones de las herramientas instaladas. Para ello se detendrán los servicios relacionados con las herramientas Hostapd y Dnsmasq mediante los comandos que se muestran a continuación:

```
$ sudo systemctl stop hostapd
$ sudo systemctl stop dnsmasq
```

3.2.2. Configuración del cliente DHCP

Se configurará el cliente DHCP de la Raspberry editando el archivo `/etc/dhcpcd.conf` gracias al uso del siguiente comando:

```
$ sudo nano /etc/dhcpcd.conf
```

Con la edición del archivo se pretenderá realizar lo siguiente:

- Asignar una IP estática a la interfaz de red de conexión inalámbrica (`wlan0`). Por buscar un símil, esta IP asignada será la puerta de enlace de los dispositivos conectados al punto de acceso.

  IMPORTANTE: La IP asignada deberá estar fuera del rango de la red que da acceso a Internet.

- Desactivar la funcionalidad `wpa_supplicant`. Esta funcionalidad tiene como finalidad realizar labores de búsqueda, detección, asociación y autenticación de redes inalámbricas dentro de la zona en la que opera, por este motivo se podrán crear eventos de conexiones no deseados, con lo que convendrá desactivar dicha funcionalidad para la interfaz especificada.

Así pues, en la edición del archivo `/etc/dhcpcd.conf` se le añadirán al final las siguientes líneas:

```
#Asignar IP estatica
interface wlan0
#Ejemplo de IP estatica
static ip_address=192.168.100.1/24
#Desactivar la funcionalidad wpa_supplicant
nohook wpa_supplicant
```

3.2.3. Configuración del servidor DHCP

Se configurará un servidor DHCP, para ello se hará uso de la herramienta Dnsmasq previamente instalada (ver *3.2.1.Preparación del entorno*). Con el servidor DHCP se pretenderá distribuir de forma dinámica los parámetros de la configuración de la red generada en la interfaz en la que se implantará el punto de acceso (`wlan0`).

Por defecto Dnsmasq dispondrá de un archivo de configuración `/etc/dnsmasq.conf` predeterminado, el cual contiene un montón de información innecesaria para el desarrollo de este proyecto. Se guardará el archivo predeterminado con otro nombre mediante el comando que se muestra a continuación:

```
$ sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.orig
```

Posteriormente se creará desde cero un nuevo archivo de configuración para la herramienta Dnsmasq mediante el siguiente comando:

```
$ sudo nano /etc/dnsmasq.conf
```

Se editará el nuevo archivo de configuración añadiéndole la interfaz para la que se distribuirán las direcciones IP y en la cual se desplegará el punto de acceso (`wlan0`). Posteriormente se añadirá el rango de direcciones IP que podrá distribuirse entre los dispositivos que se conecten a dicho punto de acceso, en este caso con 19 direcciones IP se prevé que servirá.

El archivo deberá tener el siguiente aspecto:

```
interface=wlan0
#Ejemplo de direcciones
dhcp-range=192.168.100.2,192.168.100.20,255.255.255.0,24h
```

En el parámetro `dhcp-range` se definirá el rango de direcciones IP, la mascará de red de dichas direcciones y el tiempo de concesión de las direcciones.

NOTA: Como se podrá observar las IPs definidas en este archivo pertenecen a la misma red que la IP estática definida en el punto anterior (ver *3.2.2.Configuración del cliente DHCP*).

3.2.4. Configuración de Hostapd

Para la configuración del punto de acceso se hará uso de la herramienta Hostapd previamente instalada en el punto *3.2.1.Preparación del entorno*.

Primero se generará (en caso de que exista, se editará) el archivo de configuración de Hostapd `/etc/hostapd/hostapd.conf` mediante el comando mostrado a continuación:

```
$ sudo nano /etc/hostapd/hostapd.conf
```

Dentro se añadirán los siguientes parámetros:

```
interface=wlan0
driver=nl80211
hw_mode=g
channel=7
wmm_enabled=0
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
ssid=NombreDelSSIDdelPuntoDeAcceso
wpa_passphrase=ContraseñaParaAcceder
```

Los parámetros a destacar son los siguientes:

- `interface`: La interfaz de red en la que se ejecutará la herramienta Hostapd.
- `driver`: Nombre del controlador usado por la interfaz de red definida en el parámetro `interface`.
- `hw_mode`: La frecuencia bajo la que la red generada operará a consecuencia de la generación del punto de acceso. En este caso, se escogerá una red con frecuencia de 2.4 GHz, siendo esta red de tipo IEEE 802.11g.
- `channel`: El canal [10] dependerá de la frecuencia escogida en el parámetro previo `hw_mode`.
- `ssid`: El nombre que identificará a la red generada a consecuencia del punto de acceso.
- `wpa_passphrase`: Contraseña de acceso a la red, la cual deberá ser de una longitud entre 8 y 64 caracteres.

El resto de parámetros se han asignado siguiendo las recomendaciones ofrecidas por el fabricante de la Raspberry para la generación de un punto de acceso usando dicho dispositivo [12] (cabe mencionar que alguno de los previamente mostrados también se ha asignado en base a las citadas recomendaciones).

Para finalizar con este punto, se le mostrará al sistema la ubicación del archivo de configuración de Hostapd, el previamente generado `/etc/hostapd/hostapd.conf`. Esto se realizará editando el archivo `/etc/default/hostapd` mediante el siguiente comando:

```
$ sudo nano /etc/default/hostapd
```

En la edición del archivo se buscará la línea que recoge el valor `#DAEMON_CONF=""`. Tras localizar la línea, se eliminará el carácter `#` para que deje de ser un valor comentado. Finalmente, se le añadirá la ruta del archivo de configuración. El resultado de todo el proceso descrito se recoge en la siguiente línea:

```
DAEMON_CONF="/etc/hostapd/hostapd.conf"
```

3.2.5. Configuración del reenvío de tráfico

Se habilitará el reenvío de tráfico. Mediante esta acción se pretenderá que el tráfico generado en la Raspberry pueda propagarse, desde la red generada a consecuencia del punto de acceso (`wlan0`) hacia diferentes lugares y viceversa, principalmente siendo la propagación del tráfico hacia (o desde) Internet.

Para ello se editará el archivo de configuración `/etc/sysctl.conf` mediante el comando usado en la siguiente línea.

```
$ sudo nano /etc/sysctl.conf
```

Se buscará el valor que permitirá la acción de reenvío `net.ipv4.ip_forward`, el cual por defecto debería estar comentado, y se le eliminará el carácter `#` del principio de la línea con el fin de que deje de ser un parámetro comentado. Finalmente se le añadirá el valor `1` tal y como se muestra a continuación:

```
net.ipv4.ip_forward=1
```

3.2.6. Añadir reglas de tráfico, enmascaramiento

En este apartado, y habiendo seguido en orden los puntos anteriores, se tendrá un esquema de red mostrado en *Figura 12: Topología de red antes de añadir las reglas de tráfico*, el cual se describe a continuación:

- Por una parte se tendrá la red creada en la interfaz de conexión inalámbrica (`wlan0`) a consecuencia de la generación del punto de acceso, lo que se conocerá como una red interna.
- Por otra parte se tendrá la red con acceso a Internet implantada en la conexión Ethernet (`eth0`).

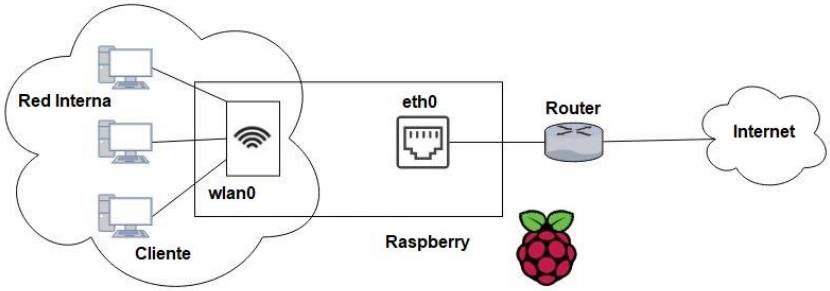


Figura 12: Topología de red antes de añadir las reglas de tráfico

Este esquema habilitará la comunicación entre ambas interfaces de red ya que estas estarán alojadas en la misma máquina. El problema surgirá cuando un dispositivo conectado a la red interna intente acceder a Internet mediante el encaminador (y viceversa), debido a que dicho encaminador, el cual dispone de acceso a Internet, tiene fuera de rango cualquier dispositivo de la red Interna. Para solucionar este problema se recurrirá al enmascaramiento de direcciones IP.

Tomando como referencia el problema que surgirá, el concepto del enmascaramiento resumidamente consistirá en lo siguiente:

1. Un dispositivo conectado a la red interna generará tráfico con dirección a Internet.
2. El tráfico será reenviado de la interfaz desde la que el dispositivo se encuentra conectado (`wlan0`) hacia la interfaz que tiene acceso a Internet (`eth0`).
3. En este punto el tráfico será enmascarado con la dirección IP de la interfaz que dispone de salida a Internet (una especie de asignación), pudiendo de esta manera tener acceso a Internet, ya que la interfaz `eth0` sí que está en el rango del encaminador.

Esto se generará mediante una regla de la herramienta `iptables`, y para ser más exactos, mediante el siguiente comando:

```
$ sudo iptables -t nat -A  POSTROUTING -o eth0 -j MASQUERADE
```

Posteriormente se guardará la regla mediante el comando mostrado a continuación:

```
$ sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"
```

Finalmente se editará el fichero `/etc/rc.local` para añadir la regla guardada antes del valor `exit 0`.

```
iptables-restore < /etc/iptables.ipv4.nat
exit 0
```

Con estos dos últimos pasos se asegurará de que al reiniciar la Raspberry la regla definida se volverá a cargar en el sistema.

3.2.7. Aplicación de la configuración

Tras definir todas las configuraciones necesarias, en este punto se recogerá como se aplicarán los cambios realizados o generados en las previas configuraciones.

Se reiniciará el servicio del cliente DHCP (este servicio previamente no se había parado) mediante el siguiente comando:

```
$ sudo service dhcpcd restart
```

Se pondrá en marcha el servicio de la herramienta Dnsmasq usando el siguiente comando:

```
$ sudo systemctl unmask dnsmasq
```

Se iniciará el servicio de la herramienta Hostapd, para ello primero se desbloqueará el servicio (por si fuese necesario), posteriormente se habilitará el servicio, y finalmente se pondrá en marcha el servicio mediante la secuencia de comandos mostrados a continuación.

```
$ sudo systemctl unmask hostapd
$ sudo systemctl enable hostapd
$ sudo systemctl start hostapd
```

Si bien es cierto que reiniciando la Raspberry se deberían aplicar los cambios realizados, se ha querido recoger la forma manual de hacerlo, puesto que en caso de que algún servicio falle se podrá consultar o investigar sobre el problema ocurrido mediante el siguiente comando:

```
$ sudo systemctl status NombreDelServicio
```

Por ejemplo, para el servicio vinculado a la herramienta Hostapd se usará el siguiente comando:

```
$ sudo systemctl status hostapd
```

3.3. Implantación del proxy a modo de MITM

Se implantará la funcionalidad de proxy a modo de MITM en la Raspberry Pi mediante la herramienta mitmproxy [19], gracias al cual se realizará el análisis de tráfico y permitirá el posterior tratamiento del mismo. En este punto el dispositivo dispondrá de todas las funcionalidades previamente implantadas.

Los materiales usados en este apartado son los siguientes:

- La Raspberry Pi modelo 3 B con la tarjeta de memoria SanDisk Ultra previamente configurada.
- Fuente de alimentación para la Raspberry.
- Cable RJ-45 para conectar la Raspberry y el encaminador.
- Un cliente para probar la implementación, en este caso un dispositivo portátil HP Pavilion 15 Notebook PC con sistema operativo Windows 10.
- Encaminador que proveerá de conexión a Internet.
- Conexión a Internet.

NOTA: Cabe mencionar que a lo largo de todo este punto se accederá a la Raspberry vía SSH mediante el usuario `pi`.

3.3.1. Instalación de mitmproxy

Se procederá a la instalación del proxy en la Raspberry, para ello se hará uso de la herramienta mitmproxy tal y como se ha especificado previamente en el apartado *2.1.4.Solución propuesta para el analizador de tráfico*.

La instalación se llevará a cabo mediante la herramienta `pip3`, previamente instalada en el apartado *3.1.7.Instalación pip3*, ya que a la hora de desarrollar este proyecto, `pip3` provee del paquete de instalación de mitmproxy más actualizado. La instalación se realizará ejecutando el siguiente comando en la terminal:

```
$ sudo pip3 install mitmproxy
```

Entre las diferentes funcionalidades de la herramienta que se instalarán, a lo largo del proyecto se hará uso de la versión de línea de comandos de la herramienta mitmproxy identificada como `mitmdump`, a fin de minimizar el consumo de recursos de la Raspberry.

3.3.2. Elección del modo de operación

El proxy instalado se podrá configurar para que opere de diferentes maneras. Teniendo en cuenta que parte del desarrollo de este proyecto consiste en implantar un proxy a modo de MITM, se escogerá la configuración del proxy en modo transparente [11], dado que esta opción no requiere de configuración en

el lado del cliente (en lo que al proxy se refiere), con lo que a priori, el cliente no deberá conocer de la existencia de dicho proxy.

NOTA: Si el cliente investiga a fondo teniendo en cuenta los certificados obtenidos, incluso si intenta rastrear el tráfico generado,… sí que llegará a conocer la existencia de algo intermediario.

El esquema de un proxy en modo transparente en desarrollo actual (ver *Figura 13: Proxy en modo transparente*) implicará lo siguiente:

1. Un dispositivo conectado a la red interna generará tráfico con dirección a Internet, para ello primero se dirigirá hacia la interfaz en la que el dispositivo se encuentra conectado (`wlan0`).
2. El tráfico será reenviado desde la `wlan0` hacia el proxy. El cliente en teoría no tendrá constancia de esta acción.
3. El proxy realizará las acciones para las que este programado, en este caso el análisis del tráfico y el tratamiento del mismo. Posteriormente direccionará el tráfico hacia la interfaz que tiene acceso a Internet (`eth0`). El cliente en teoría tampoco tendrá constancia de esta acción.
4. Finalmente el tráfico se enmascarará y la interfaz `eth0` lo pondrá rumbo Internet. El cliente en teoría no tendrá constancia de la acción del enmascaramiento, no así como las conexiones físicas de la topología, las cuales podrá apreciar.

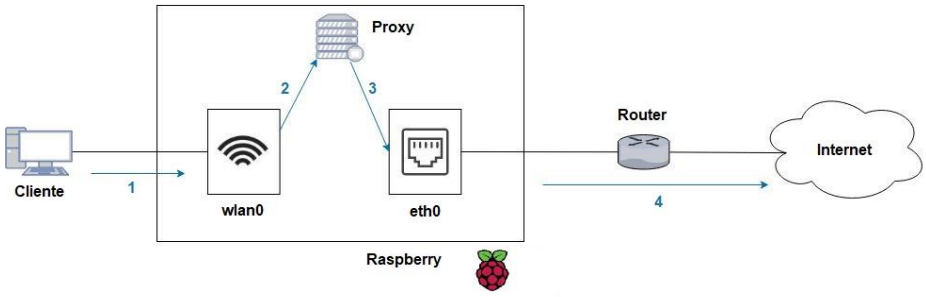


Figura 13: Proxy en modo transparente

En el esquema previamente mostrado el dispositivo conectado a la red (el cliente) no debiera saber de la existencia del proxy, tal y como se ha recalcado previamente, puesto que no interactuará directamente con él. La visión que el cliente deberá tener es que el tráfico generado con destino a Internet pondrá rumbo al punto de acceso, y de ahí, presupondrá que tomará dirección Internet, es decir, no tendrá la noción de que exista un dispositivo intermediario en la comunicación o un proxy realizando acciones *man-in-the-middle*.

3.3.3. Añadir reglas de tráfico, redirección

En este punto se añadirán las reglas de tráfico necesarias para redirigir el tráfico interno de la Raspberry hacia el proxy. Esta acción se realizará definiendo una regla mediante la herramienta `iptables` (usada en el apartado anterior *3.2.6.Añadir reglas de tráfico, enmascaramiento*) mediante el siguiente comando:

```
$ sudo iptables -t nat -A PREROUTING -i wlan0 -p tcp -m multiport --dports 80,443
                          -j REDIRECT --to-ports 8080
```

La regla, para ser algo más exactos, aplicará lo siguiente: Todo tráfico que haya sido direccionado desde la interfaz de conexión inalámbrica de la Raspberry (`wlan0`) con destino a los puertos `80` y `443` lo redirigirá al puerto `8080`.

Un par de menciones respecto a la regla:

- El valor del puerto `8080` es el puerto de escucha del proceso `mitmproxy` (`mitmdump` en el caso de este proyecto), el cual está asignado por defecto y así se mantendrá. Si se quiere cambiar se podrá hacer uso del argumento [12] `web_port` a la hora de ejecutar el proceso.
- El valor de los puertos `80` y `443` hace referencia a los puertos relacionados con los protocolos HTTP y HTTPS, dicho de otra manera, el valor de estos puertos hace referencia al tráfico HTTP y HTTPS que se generará a través de la interfaz `wlan0`.

En este caso, esta regla no se guardará para que una vez se reinicie el dispositivo esta vuelva a activarse, aunque se podrá llevar a cabo siguiendo los pasos previamente mostrados en el punto *3.2.6.Añadir reglas de tráfico, enmascaramiento*. Principalmente esto se ha decidido por los siguientes motivos:

1. El puerto de escucha del proxy podrá variar, con lo que la regla quedará inservible.
2. El tipo de tráfico a redirigir podrá variar, con lo que la regla quedará inservible.

3.3.4. Instalación del certificado

Con todo lo desarrollado hasta el momento, y una vez puesto `mitmdump` en marcha, el proxy únicamente podrá interceptar y analizar el tráfico HTTP, puesto que este a diferencia del tráfico HTTPS este no viajará usando protocolos seguros.

Para que el proxy sea capaz de interceptar y analizar el tráfico encriptado HTTPS necesitará que el cliente confíe en él. Esto se conseguirá de dos formas:

1. El cliente admitirá el certificado emitido por el proxy a base de confiar en la autoridad certificadora (CA) que lo verifica.

2. El certificado emitido por el proxy estará verificado por una autoridad certificadora en la cual el cliente previamente confía.

Lo ideal será que el certificado proporcionado por mitmproxy este verificado por una autoridad certificadora en la cual el cliente previamente confía, esto evitará hacer cualquier tipo de configuración en el lado del cliente.

Para el desarrollo de este proyecto se forzará que el cliente admita el certificado emitido por el proxy por tres principales motivos:

1. En lo que al tiempo de desarrollo se refiere, se desarrollará este proyecto sin depender de terceras entidades, en este caso, de una autoridad certificadora en la cual el cliente previamente confía.
2. En cuanto al coste se refiere, mediante la implementación optada saldrá más barato ya que no habrá que contratar servicios de autoridades certificadores.
3. Se prevé que para el desarrollo del proyecto esta opción será suficiente, puesto que se dispone del conocimiento de ambas soluciones.

Así pues, se procederá a forzar la admisión del certificado para el cliente con las siguientes características: Un dispositivo portátil HP Pavilion 15 Notebook PC con sistema operativo Windows 10 y el navegador Firefox.

Se pondrá el proxy en marcha en modo transparente mediante el siguiente comando:

```
$ sudo mitmdump --mode transparent
```

Se conectará el cliente a la red generada mediante la Raspberry. Una vez conectado se abrirá el Firefox. Si dicho cliente intentase hacer alguna petición mediante el protocolo HTTPS el navegador Firefox le mostrará una ventana de error tal y como muestra la siguiente imagen (ver *Figura 14: Problema de seguridad Firefox, falta certificado*).

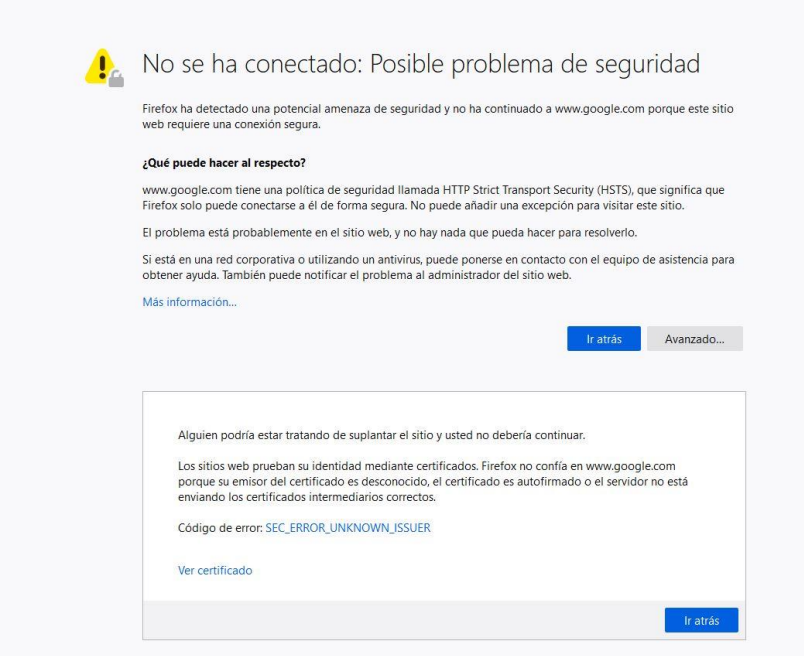


Figura 14: Problema de seguridad Firefox, falta certificado

Si el cliente en cambio intentase hacer alguna petición mediante el protocolo HTTP el navegador se lo permitirá. Esto se podrá comprobar a continuación, puesto que para obtener el certificado se deberá acceder mediante el protocolo HTTP a la dirección `http://mitm.it/` tal y como muestra la siguiente imagen.

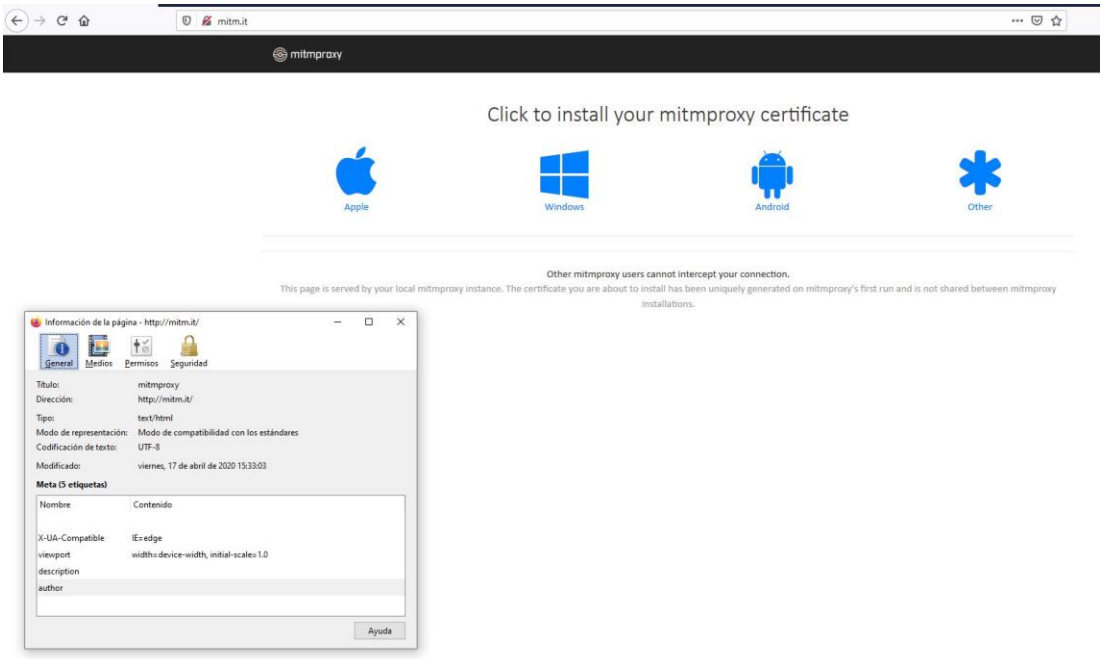


Figura 15: Página de http://mitm.it

NOTA: Mencionar que también se puede hacer uso de certificados autofirmados.

Se seleccionará la plataforma en la que se está operando para obtener su respectivo certificado, en este caso se seleccionará `Windows`.

Se obtendrá un archivo PKCS 12 con nombre `mitmproxy-ca-cert.p12` y se instalará siguiendo los pasos del asistente para importar certificados tal y como se muestra en la siguiente imagen.

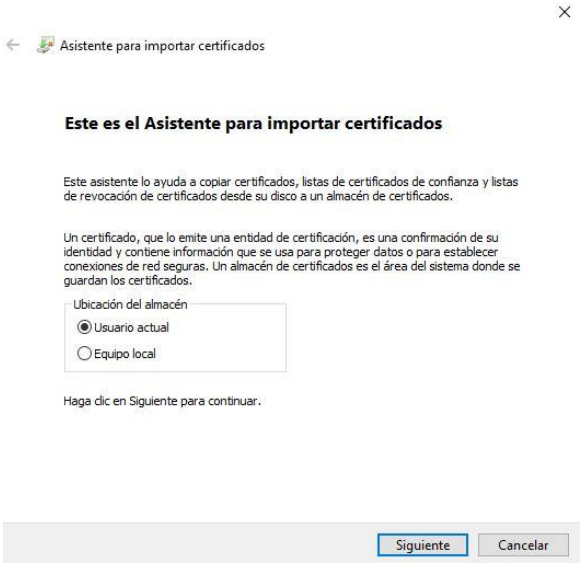


Figura 16: Asistente para importar certificados Windows

Cabe mencionar que las acciones llevadas a cabo a la hora de forzar la admisión del certificado variarán según el tipo de cliente. Por ejemplo, si el cliente hubiese sido algún dispositivo que opere bajo iOS dicho certificado deberá ser obtenido vía Safari por recomendaciones del equipo desarrollador de mitmproxy.

NOTA: En este caso, se hará uso del navegador Firefox, con lo que si se intenta hacer alguna petición mediante el protocolo HTTPS con el certificado en el sistema operativo ya importado, el navegador devolverá una página de error tal y como muestra la siguiente imagen (ver *Figura 17: Problema de seguridad Firefox, certificado no localizado en el almacén Mozilla CA*).

Esto se debe a que Firefox usa únicamente el almacén de Mozilla CA para verificar que la conexión sea segura, en lugar de los certificados proporcionados por el sistema operativo del usuario.

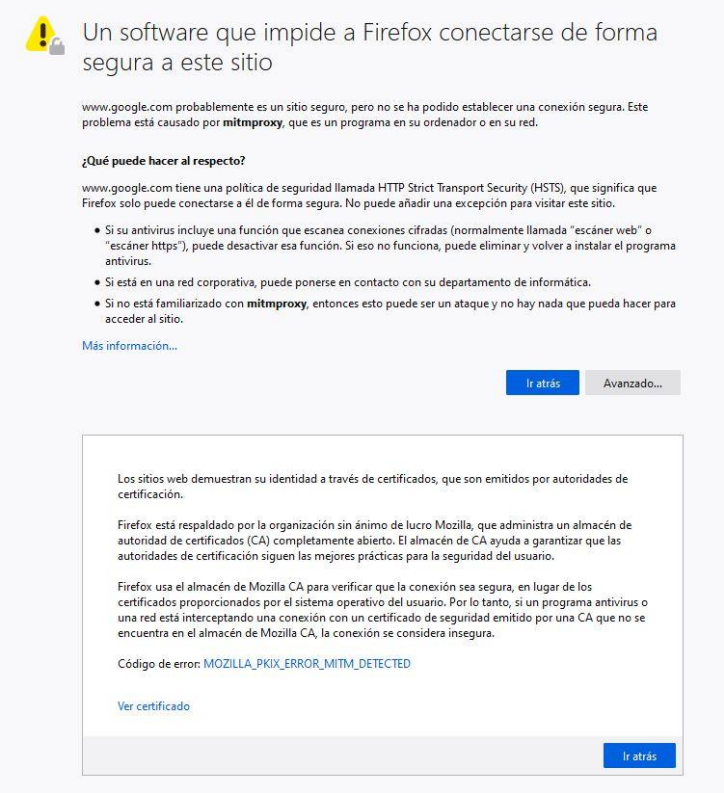


Figura 17: Problema de seguridad Firefox, certificado no localizado en el almacén Mozilla CA

Para solucionar este problema se deberá importar el certificado dentro del almacén de certificados de Firefox.

Para ello primero se tendrá que generar un certificado con un formato soportado por el almacén de Firefox adaptando el certificado previamente obtenido desde `http://mitm.it/`.

El certificado a importar al almacén de Firefox, `mitmproxy-ca.pem`, se generará mediante el siguiente comando en una terminal de Windows:

```
$ openssl pkcs12 -in mitmproxy-ca-cert.p12 -out mitmproxy-ca.pem -clcerts
                                    -nokeys
```

Finalmente se importará el certificado al almacén de certificados de Firefox mediante las opciones del navegador tal y como muestra la siguiente imagen (ver *Figura 18: Importación de un certificado al almacén de certificados Firefox*).

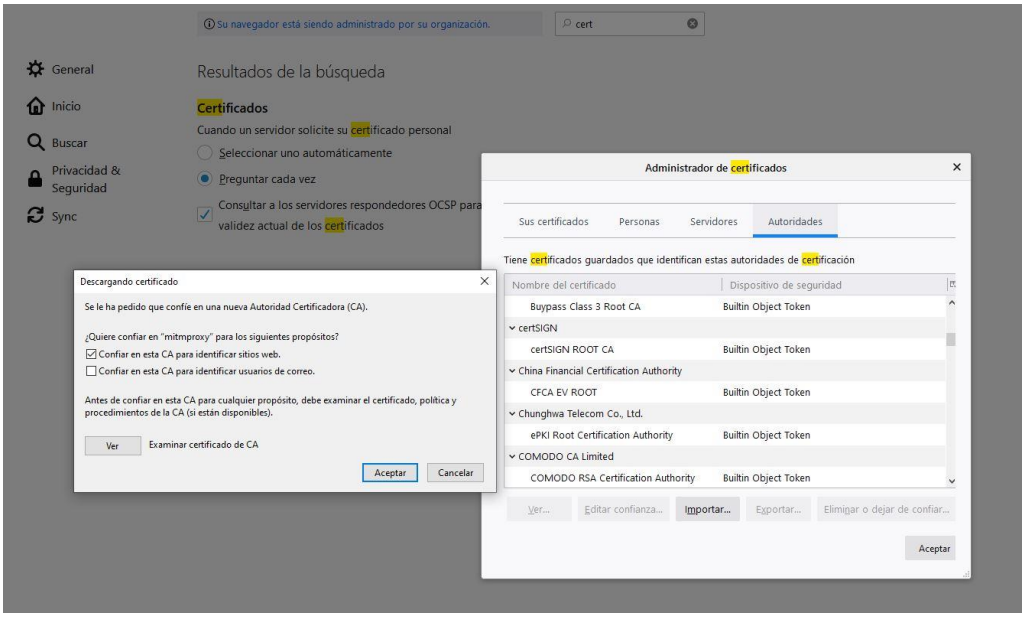


Figura 18: Importación de un certificado al almacén de certificados Firefox

3.4. Tratamiento del tráfico

En este punto se desarrollará el tratamiento que se le dará al tráfico que pueda llegar a suponer futuros riesgos, principalmente, se resaltará la funcionalidad de generación de alertas, la cual permitirá identificar posibles amenazas. Para llevar a cabo la generación de alertas se hará uso de scripts escritos en Python los cuales se ejecutarán junto a la herramienta mitmproxy [20] instalada en el apartado anterior (ver *3.3.Implantación del proxy a modo de MITM*). Las alertas generadas se visualizarán a través de la aplicación Telegram.

Los materiales usados en este apartado son los siguientes:

- La Raspberry Pi modelo 3 B con la tarjeta de memoria SanDisk Ultra previamente configurada.
- Fuente de alimentación para la Raspberry.
- Cable RJ-45 para conectar la Raspberry y el encaminador.
- Un dispositivo para la configuración de de alertas, en este caso un dispositivo portátil HP Pavilion 15 Notebook PC con sistema operativo Windows 10.
- Un dispositivo móvil operando bajo un sistema operativo Android para la recepción de alertas.
- Encaminador que proveerá de conexión a Internet.
- Conexión a Internet.

NOTA: El acceso a la Raspberry se efectuara vía SSH mediante el usuario `pi`.

NOTA: es posible que en algunas imágenes mostradas a lo largo de este apartado contengan los datos ofuscados por seguridad.

3.4.1. Instalación de Telegram

La instalación de Telegram se realizará en dos entornos diferentes: en el primer entorno se obtendrá la versión de escritorio para una máquina operando bajo un sistema operativo Windows, y en el segundo entorno se obtendrá la versión para un dispositivo móvil operando bajo un sistema operativo Android.

Para realizar la instalación de escritorio en un entorno Windows se accederá al apartado `Apps` de la página web oficial de Telegram [16]. Una vez se haya accedido, se escogerá la opción `Telegram for Windows/Mac/Linux` dentro del menú `Desktop apps` como muestra la siguiente imagen (ver *Figura 19: Aplicaciones de Telegram disponibles*).

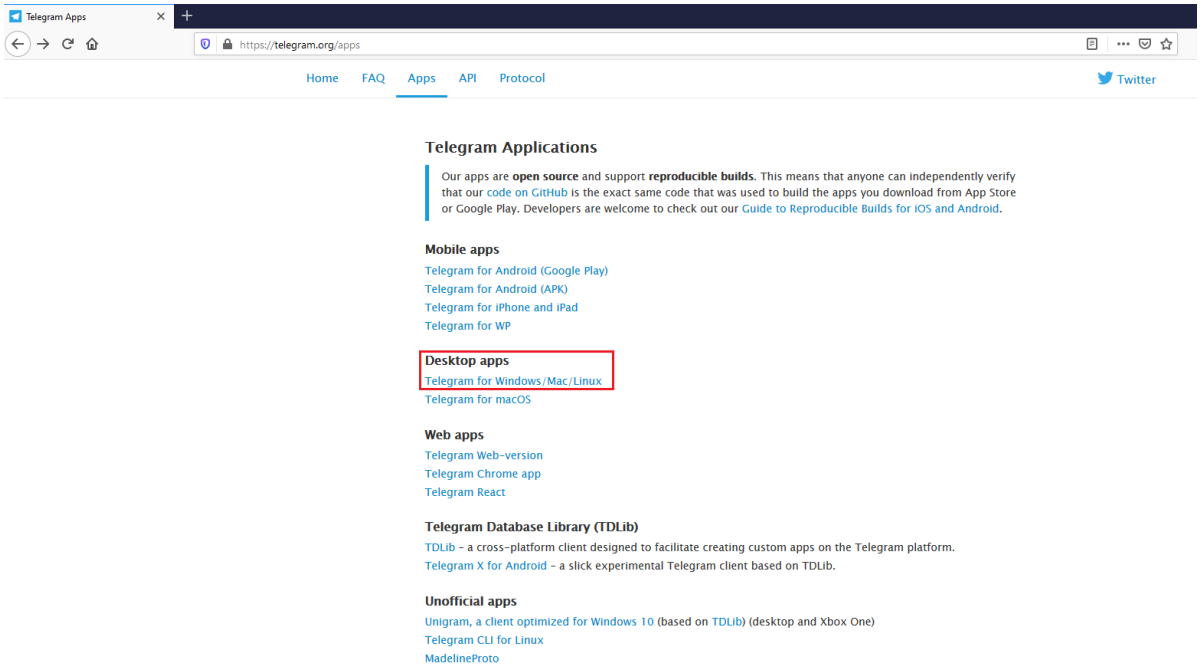


Figura 19: Aplicaciones de Telegram disponibles

Finalmente se obtendrá el instalador seleccionando `Get Telegram For Windows` mostrado en la imagen de a continuación.

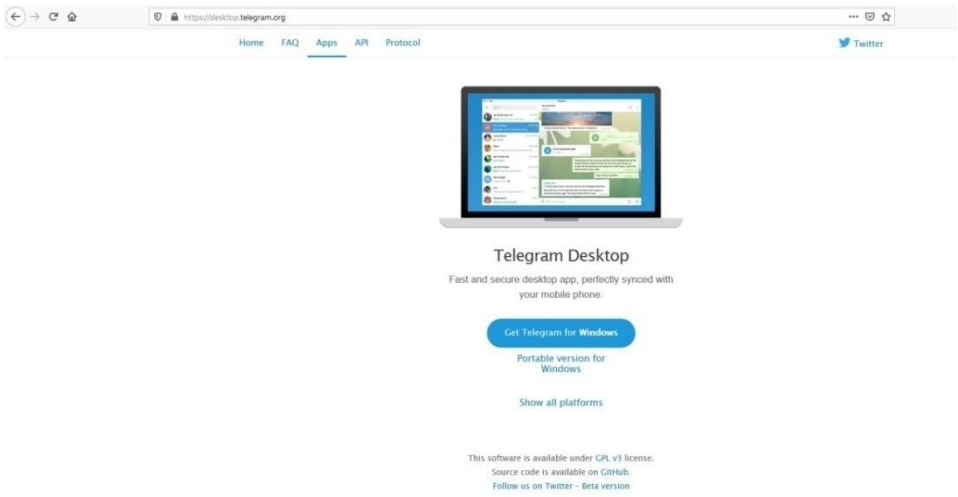


Figura 20: Obtención de la versión escritorio Telegram para Windows

Para realizar la instalación en un dispositivo Android se accederá al apartado `Apps` de la página web oficial de Telegram [16]. Una vez se haya accedido se escogerá la opción `Telegram for Android`

`(Google Play)` dentro del menú `Mobile apps`, la cual redirigirá a la aplicación Google Play del dispositivo Android.

Finalmente dentro de la aplicación Google Play se seleccionará el botón `Instalar`.

3.4.2. Preparación del entorno en Telegram

En este apartado, se preparará el entorno necesario en Telegram para que sea posible la recepción y generación de alertas. Este entorno se ha preparado mediante la aplicación de escritorio Telegram ejecutada en un sistema operativo Windows. El entorno a preparar, independientemente de la plataforma que se use, se dividirá en las siguientes fases:

- Creación del canal.
- Creación del bot.
- Integración del bot en el canal.

Creación del canal

Se creará un canal un nuevo canal, para ello se escogerá el menú de opciones de la aplicación  seguido de la opción `Nuevo canal`. Posteriormente se nombrará el canal y se escogerá la opción de hacerlo público tal y como muestra la siguiente imagen.

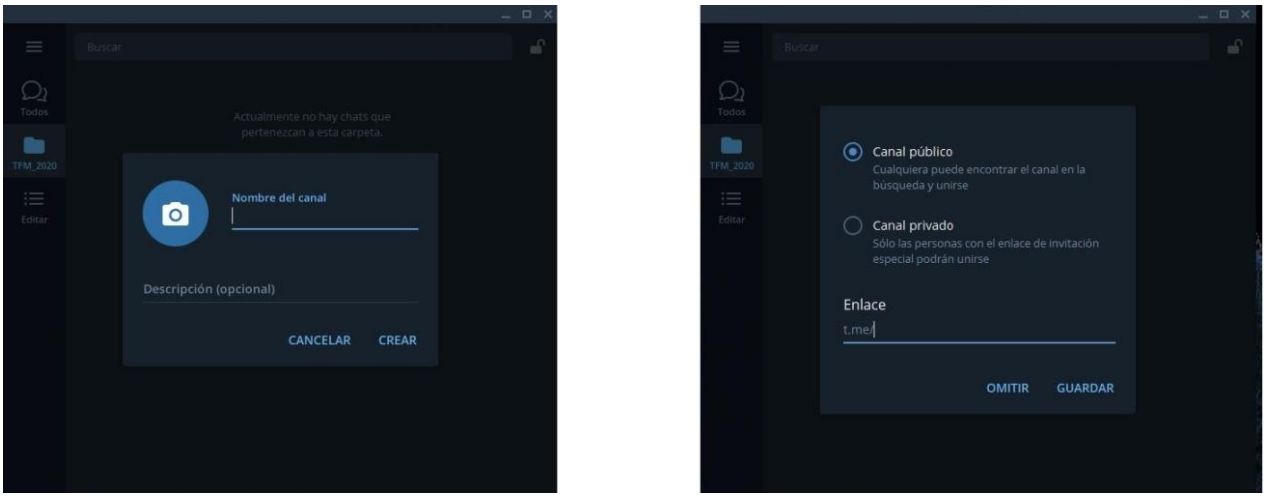


Figura 21: Creación de un canal en Telegram

Creación del bot

Para la creación de un bot en Telegram se seguirán los siguientes pasos:

1. Se contactará con el bot `Botfather` [13], el cual será el encargado de generar los bots de Telegram.

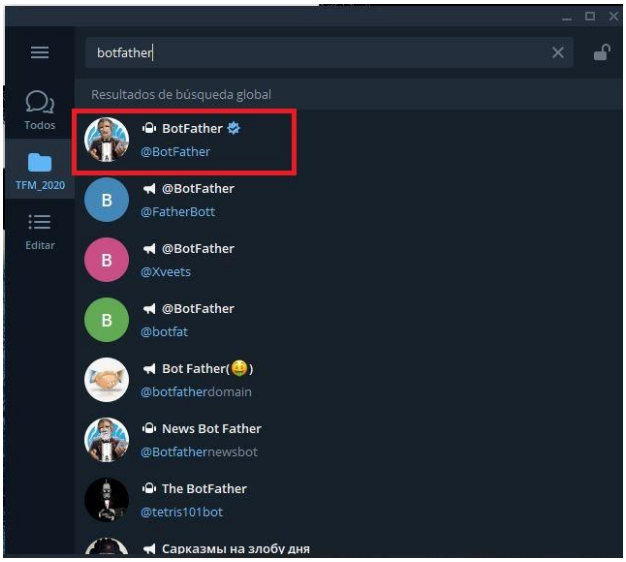


Figura 22: Contacto de Botfather

2. Se iniciará la interacción con `Botfather` pulsando `Iniciar`.

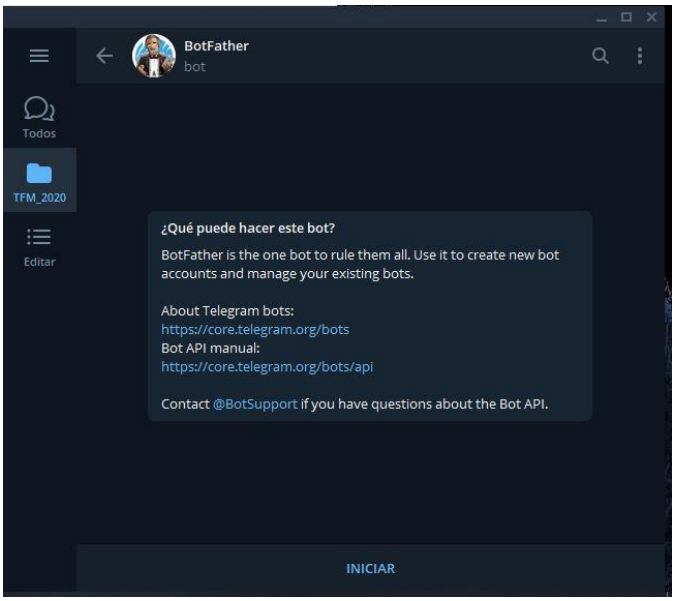


Figura 23: Iniciar Botfather

3. Se generará un nuevo bot enviando el comando `/newbot`.

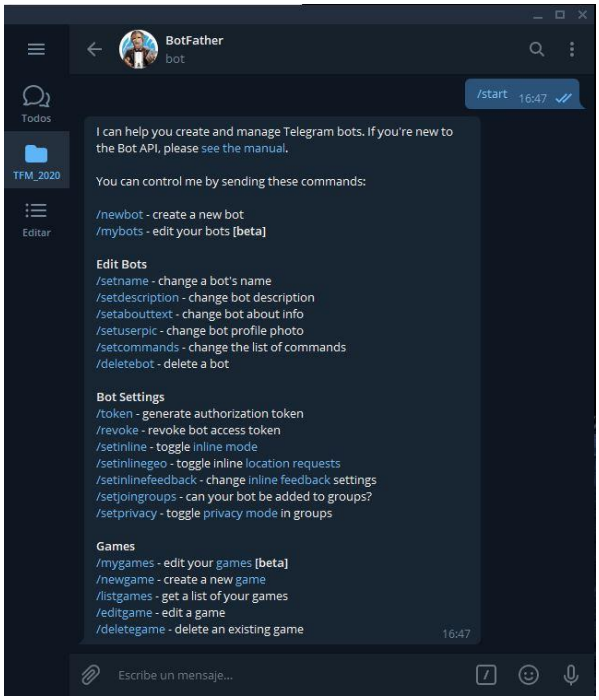


Figura 24: Creación de un nuevo bot en Telegram

4. Para finalizar con la creación del bot se le asignará un nombre de contacto (`name`) y un nombre para la búsqueda (`username`).

   IMPORTANTE: Una vez generado el bot `Botfather` devolverá un token asignado al bot creado, el cual se muestra oculto en la siguiente imagen (ver *Figura 25: Generación del bot en Telegram*). Ese token se deberá guardar para un futuro uso en la generación de alertas.

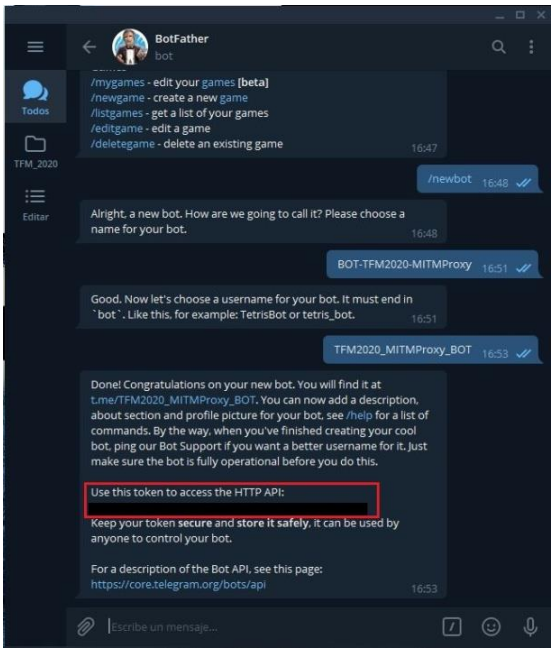


Figura 25: Generación del bot en Telegram

Una vez se hayan seguido los pasos de creación del bot, se buscará el contacto del bot en la aplicación Telegram y posteriormente se pondrá en marcha mediante el botón `Iniciar`.

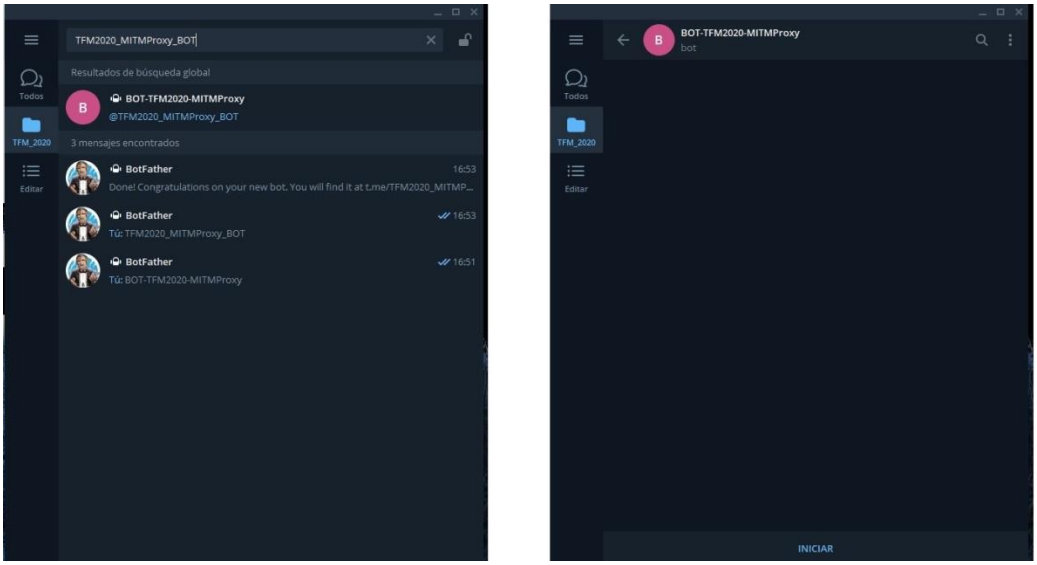


Figura 26: Búsqueda e inicio del bot en Telegram

Integración del bot en el canal

Para concluir con la preparación del entorno en Telegram se integrará el bot generado en el canal creado siguiendo los siguientes pasos:

1. Se añadirá como miembro del canal creado el bot previamente generado mediante la opción `Añadir miembros` proporcionada por el canal.

   NOTA: Cualquier bot de Telegram que se añada dentro de un canal de Telegram deberá tener el rol de Administrador, por este motivo se le limitarán los permisos.

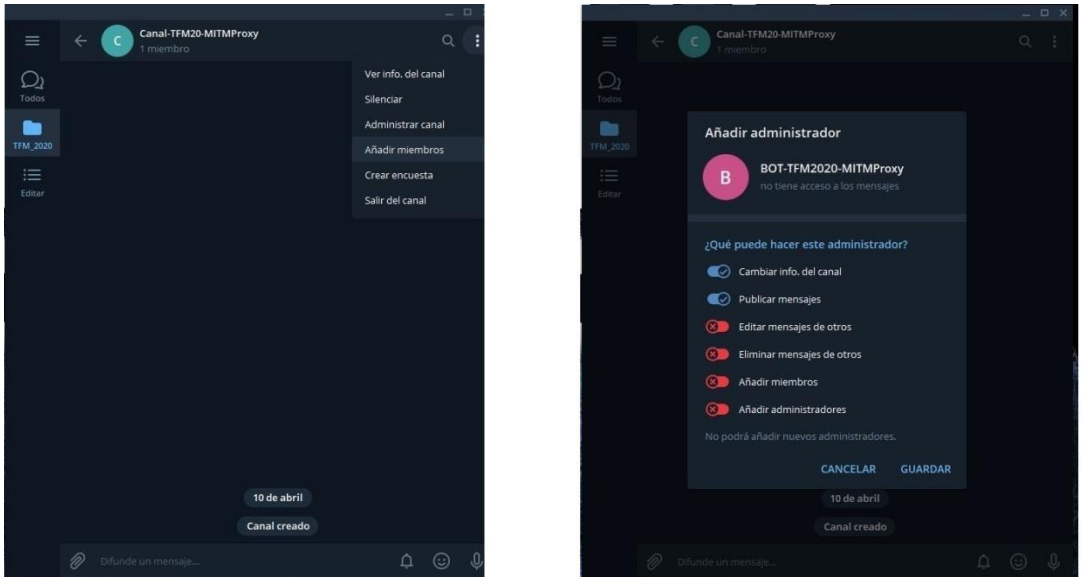


Figura 27: Añadir un bot de Telegram a un canal de Telegram

En este punto ya estaría el bot integrado en el canal. Con el fin de añadirle un extra de seguridad al desarrollo del proyecto y la obtención de las alertas, se limitará el acceso al canal haciéndolo privado. Para ello se deberán llevar a cabo las siguientes pautas:

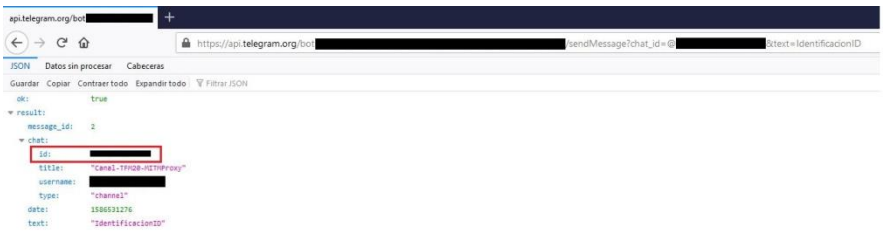
1. Se mandará una petición HTTPS vía GET (también se podrá usar el método POST) a la API `Telegram Bot API` para dos cosas:
    a. Verificar que el bot se ha unido correctamente al canal;
    b. Obtener el `id` del canal, el cual posteriormente se usará para el envío de alertas.

    La petición HTTPS que se mandará se formará siguiendo la siguiente estructura: `https://api.telegram.org/bot`*`APITokenBot/NombreMetodoYParametros`*.

    En este caso, para verificar que el bot se ha unido correctamente al canal se enviará un mensaje al mencionado canal haciendo uso del método `sendMessage` como se muestra en la siguiente petición:
    `https://api.telegram.org/bot`*`APITOKENBOT`*`/sendMessage?chat_id=@`*`NombrePublicoDelCanal`*`&text=`*`Mensaje`*

    NOTA: El parámetro *`NombrePublicoDelCanal`* es el texto añadido en el valor `Enlace` a la hora de escoger la opción `Canal público` en la generación del canal mostrado en la imagen *Figura 21: Creación de un canal en Telegram*. El parámetro *`APITOKENBOT`* tiene el valor del token obtenido en la figura *Figura 25: Generación del bot en Telegram*.

    Tras el envío del mensaje se obtendrá el id del canal y el mensaje enviado en el canal.



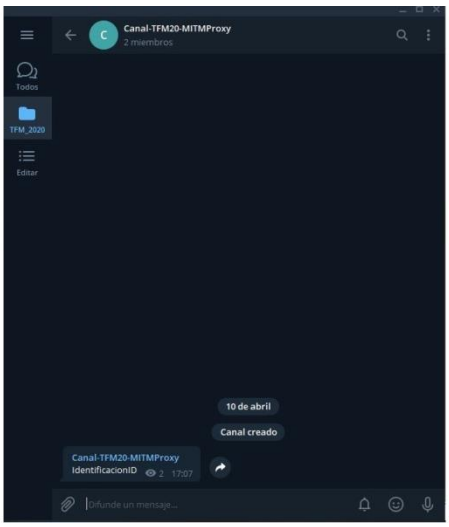


Figura 28: Mensaje bot Telegram al canal público vía HTTPS

2. Se modificará el canal haciéndolo privado. Para ello dentro de las opciones del canal se escogerá `Administrar canal` modificándolo a un canal privado mediante la elección del tipo de canal a `Canal privado`.

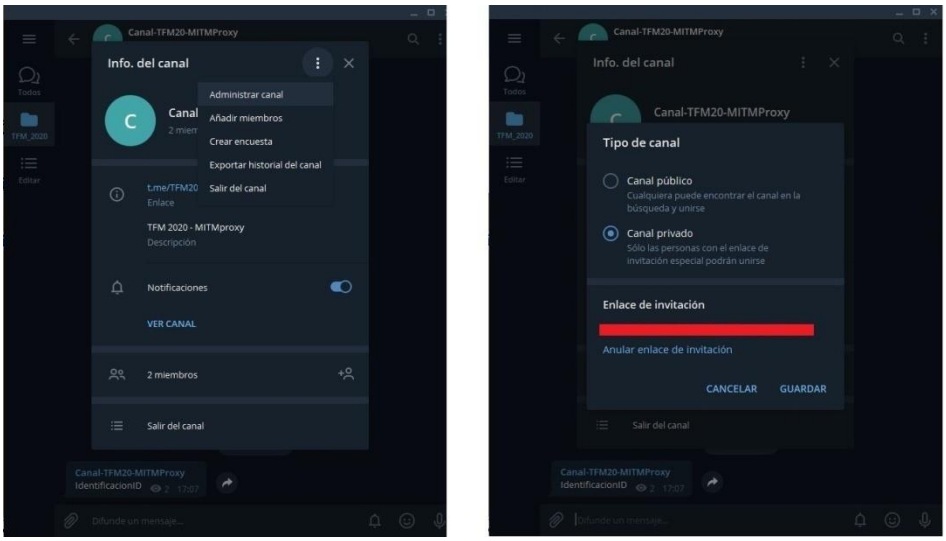


Figura 29: Modificación del tipo de canal de Telegram

3. Finalmente, se volverá a realizar la petición HTTPS a la API `Telegram Bot API` para la verificación de los cambios. En este caso el parámetro `NombrePublicoDelCanal` tomará el valor del `id` previamente obtenido.

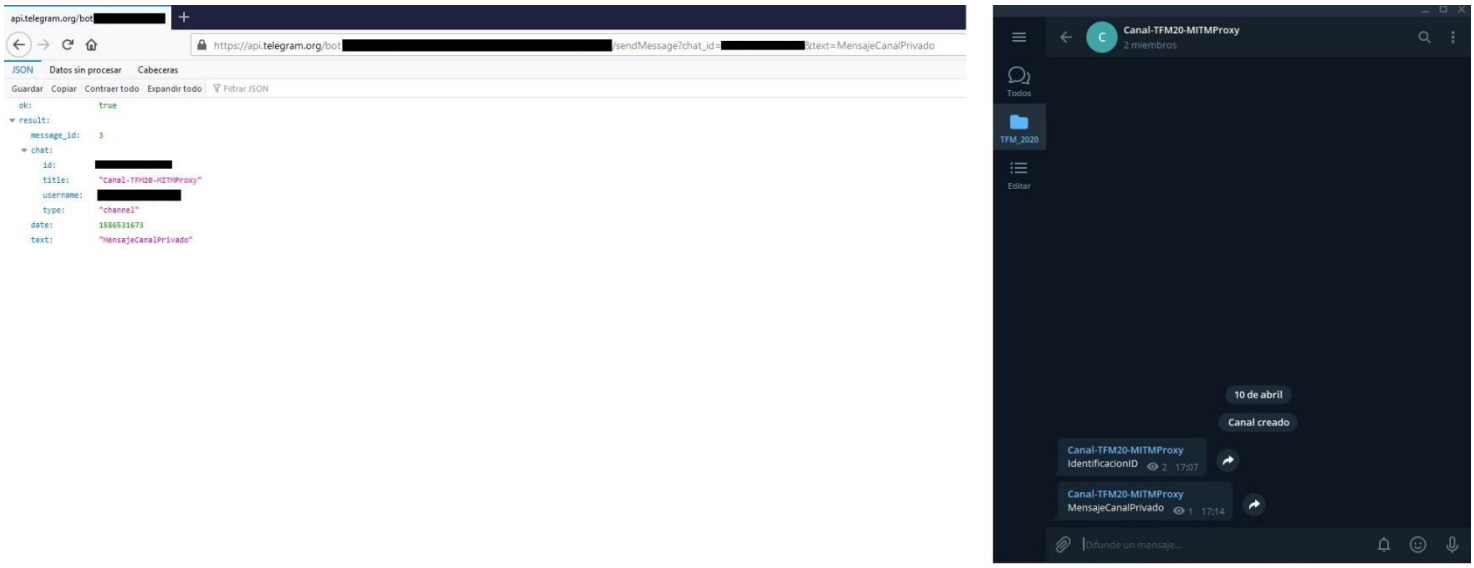


Figura 30: Mensaje bot Telegram al canal privado vía HTTPS

3.4.3. Instalación de telepot

Se procederá a la instalación de la librería de Python `telepot` en la Raspberry mediante la herramienta `pip3` previamente instalada. La instalación se realizará ejecutando el siguiente comando en la terminal:

```
$ sudo pip3 install telepot
```

3.4.4. Generación del script

En este punto se creará el script escrito en Python en la Raspberry, gracias al cual se generarán las alertas que avisarán del tráfico sospechoso.

Para ello, se abrirá un editor y se procederá con la creación del script mediante el siguiente comando:

```
$ nano NombreScript.py
```

Posteriormente se comenzará con la generación del script, el cual seguirá el diseño previamente definido en el apartado *2.4.6.Diseño del script*. Este diseño constará de los siguientes tres puntos principales:

1. La definición de los módulos Python necesarios que se usarán a lo largo del script.
2. La definición de las variables globales que se usarán a lo largo del script.
3. La definición de las funciones que se usarán para la generación de alertas.

En la primera parte del script, como previamente se ha mencionado, se definirán los módulos necesarios que se usarán. Aparte de los módulos referenciados en el apartado del diseño del script, *2.4.6.Diseño del script*, también se usarán los siguientes módulos:

- El módulo `threading`, el cual habilitará procesos simultáneos tras la ejecución del script. Se usará para mantener actualizadas las listas de reputación cada cierto periodo de tiempo.
- El módulo `re`, que habilitará el uso de expresiones regulares, gracias al cual se podrán obtener únicamente las IPs/URLs de las listas de reputación de terceros.
- Cabe mencionar que no se ha hecho uso del módulo `urllib.request` por completo como se preveía, bastará con el uso del submódulo `urlopen`.

Posteriormente se definirán las variables globales que se usarán a lo largo del script. Dentro de las cuales se identificarán:

- El bot de Telegram previamente definido.
- Las lista que contiene las IPs sospechosas.
- Las lista que contiene las URLs sospechosas.
- Las URLs de donde se obtendrán las listas de reputación de terceros.

NOTA: En este proyecto a modo de prueba únicamente se obtendrá una lista de reputación tipo *blacklist*, la cual contiene las IPs relacionadas con la botnet Kraken [14].

Para mostrar que se ha fijado la posibilidad de generar listas propias el desarrollador ha inicializado una IP que considerará como sospechosa (aunque realmente no lo sea) dentro de la lista que contiene las IPs sospechosas.

El mismo procedimiento lo ha llevado a cabo dentro de la lista que contiene las URLs sospechosas definiendo una URL que se considerará como sospechosa (aunque realmente no lo sea).

Finalmente se definirán las funciones que se usarán para la generación de alertas. Se definirán las siguientes tres funciones:

- La función `request`, a la cual se la llamará una vez ocurra el evento de leer una solicitud de tráfico por parte de la herramienta mitmproxy. En esta función se obtendrá la información respectiva a las solicitudes provenientes del tráfico analizado, para posteriormente realizar el tratamiento de las mismas y generar la alerta mediante el bot de Telegram integrado (si esta fuese necesaria). La alerta se generará en caso de que el destino de la solicitud coincida con algún valor de las IPs y URLs consideradas como sospechosas.
- La función `actualizarListas`, la cual se ejecutará mediante un proceso simultaneo cada cierto periodo de tiempo controlado a base de un temporizador. Esta función se encargará de ir actualizando las listas de reputación obtenidas de terceros.
- La función `obtenerContenidoListas`, gracias a la cual se obtendrán las listas de reputación de terceros. Esta función consta de dos principales puntos, en el primero, se obtendrá la información de las listas de reputación de terceros; en el segundo, se adaptará la información obtenida.

El script completo se mostrará en *Anexo A: Script usado en la generación de alertas* ofuscando por seguridad algunos de los valores reales usados.

3.4.5. Integración del script

Se procederá a integrar el script generado en la herramienta mitmproxy. Para ello se hará uso de la funcionalidad que proporciona dicha herramienta una vez se ponga en marcha mediante el argumento `-s`.

Así pues, se pondrá en marcha el proxy en modo transparente con el script ejecutándose mediante el siguiente comando:

```
$ sudo mitmdump --mode transparent -s NombreScript.py
```

4. Fase de demostración

En este capítulo se mostrará una guía de usuario del proyecto desarrollado. En esta fase todos los elementos/herramientas estarán previamente configurados o creados tal y como se ha mostrado a lo largo del documento. Por su parte, la Raspberry ya se encuentra conectada al encaminador mediante un cable de red.

Para ello se han hecho uso de los siguientes materiales:

- La Raspberry Pi modelo 3 B con la tarjeta de memoria SanDisk Ultra previamente configurada.
- Fuente de alimentación para la Raspberry.
- Cable RJ-45 para conectar la Raspberry y el encaminador.
- Un cliente para realizar la demostración, en este caso un dispositivo portátil HP Pavilion 15 Notebook PC con sistema operativo Windows 10 y un navegador Firefox.
- Un dispositivo con Telegram instalado, en este caso un dispositivo móvil operando bajo un sistema operativo Android.
- Encaminador que proveerá de conexión a Internet.
- Conexión a Internet.

NOTA: Es posible que dependiendo de los recursos que el cliente use haya que realizar adaptaciones. Por ejemplo, si alguien accede mediante un cliente el cual hace uso del sistema operativo iOS la forma de realizar la instalación del certificado para el proxy (ver *3.3.4.Instalación del certificado*) se deberá adaptar.

4.1. Conexión de los dispositivos

Este apartado recogerá los pasos que habrá que seguir para que un cliente pueda conectarse a la red generada a consecuencia de la creación del punto de acceso. Como previamente se ha mostrado dicho cliente hará uso de un portátil HP Pavilion 15 Notebook PC con sistema operativo Windows 10 y navegador Firefox.

Se seguirán los siguientes pasos:

1. Se abrirá algún visor que muestra las redes disponibles.

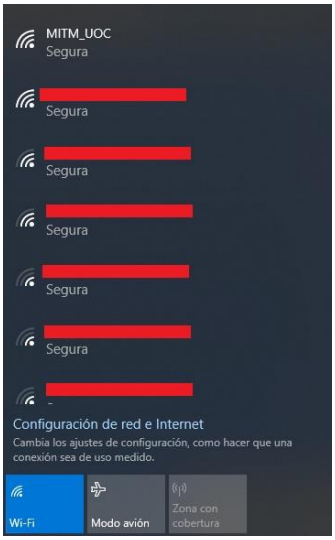


Figura 31: Redes del cliente disponibles

2. Se conectará a la red generada.

NOTA: La contraseña será la asignada en el punto *3.2.4.Configuración de Hostapd*.

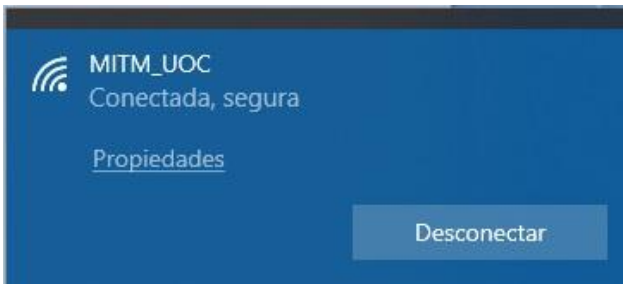


Figura 32: Red relativa al punto de acceso

3. Para finalizar con este punto y comprobar que la conexión es correcta se abrirá el navegador Firefox y se realizará una petición HTTP o HTTPS.

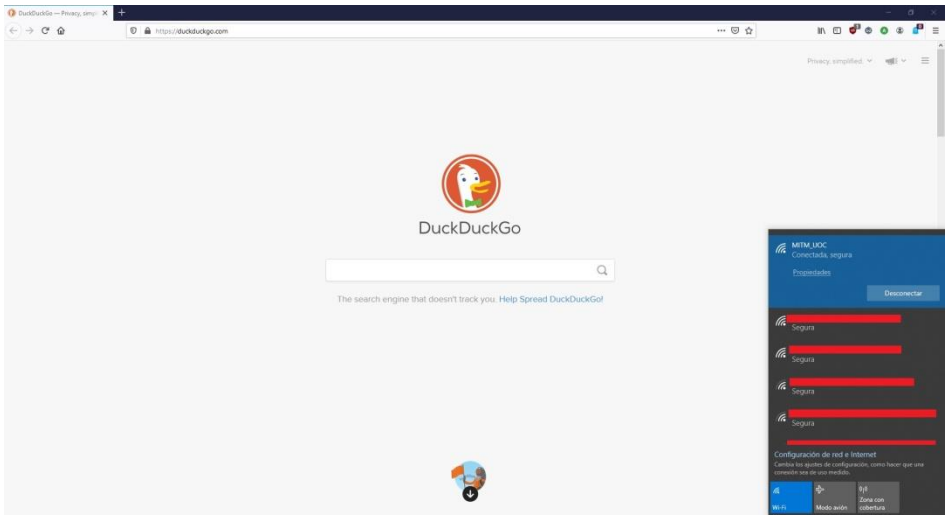


Figura 33: Petición con el navegador Firefox y la red de conexión

NOTA: En caso de que algo no funcione como es debido se deberá consultar los puntos *3.1.Preparación de la Raspberry* o *3.2.Implantación del punto de acceso*.

4.2. Análisis del tráfico

En lo que al cliente respecta, dentro de la guía de usuario del proyecto no deberá efectuar ninguna acción más que la de realización de consultas, como por ejemplo la efectuada en la siguiente imagen.

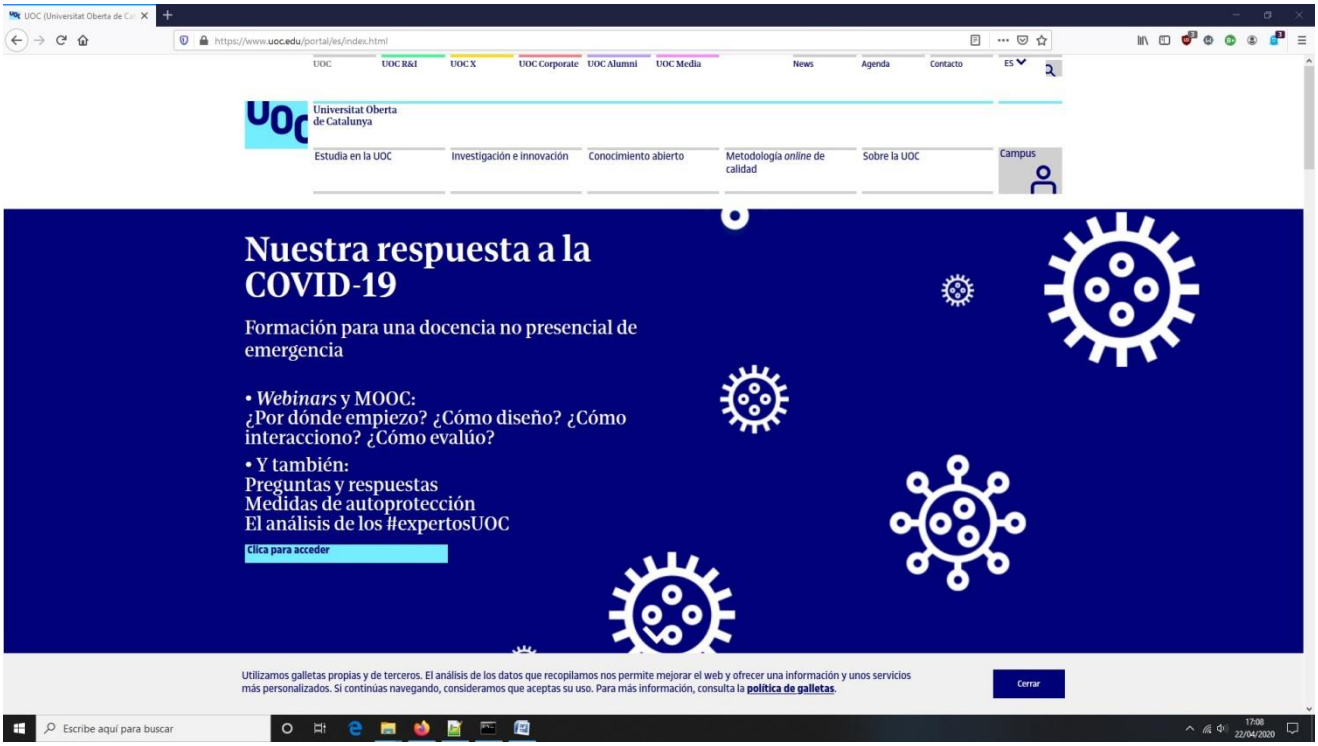


Figura 34: Consulta realizada para el análisis de tráfico

NOTA: El cliente en este proyecto deberá tener el certificado instalado tal y como se especifica en el punto *3.3.4.Instalación del certificado*.

Para este apartado sí que se detallará que el responsable o administrador del proyecto desde la Raspberry podrá observar cómo se está realizando el análisis de tráfico. Si todo va bien, verá una pantalla en la que se muestra un análisis del tráfico parecido a la imagen *Figura 35: Análisis de tráfico mediante mitmproxy (mitmdump)* mostrada a continuación.

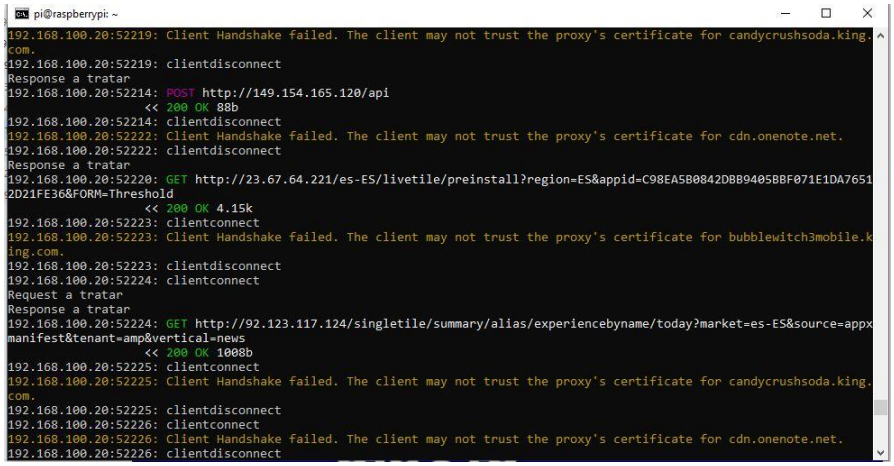


Figura 35: Análisis de tráfico mediante mitmproxy (mitmdump)

NOTA: En caso de que algo no funcione como es debido se deberá consultar los puntos *3.1.Preparación de la Raspberry* o *3.3.Implantación del proxy a modo de MITM*.

4.3. Tratamiento del tráfico

Para este punto se deberá disponer de un dispositivo que tenga Telegram instalado con acceso al canal privado en donde se generarán las alertas. Previamente se ha mencionado que se hará uso de un dispositivo móvil operando bajo un sistema operativo Android.

Lo único que se deberá hacer en este punto es esperar que se reciba una alerta a consecuencia de una conexión llevada a cabo por el cliente. Una vez se reciba, esta se consultará en el canal privado generado a lo largo del desarrollo en el punto *3.4.2.Preparación del entorno en Telegram*. En la imagen *Figura 36: Alerta recibida en Telegram desde Python* mostrada a continuación se muestra el ejemplo de una alerta, en la cual se podrá observar:

- La fecha (horario incluido) en la que la que la alerta se ha generado.
- La dirección IP o URL a la que se intentará acceder, la cual ha hecho generar la alerta.

Figura 36: Alerta recibida en Telegram desde Python

Para ser más exactos la alerta mostrada en la imagen anterior, *Figura 36: Alerta recibida en Telegram desde Python*, pertenece a la petición de la imagen *Figura 34: Consulta realizada para el análisis de tráfico*. Esto se podrá comprobar observando: por una parte la hora en la que la petición se ha realizado y la hora en la que la alerta se ha generado, ambas coinciden; y por otra parte, gracias a la obtención de las características de la petición mediante el uso del inspector del navegador, mostrado en la siguiente imagen (ver *Figura 37: Inspección de la consulta realizada*) en la que se recalcará:

- La dirección IP que generará la alerta, en este es una IP relacionada con la plataforma Vimeo.
- La dirección URL que generará la alerta, en este caso la plataforma Youtube.

NOTA: El porqué de dichas alertas se podrá observar en *Anexo A: Script usado en la generación de alertas*, dado que estás IPs/URLs están consideradas en el script como sospechosas por parte del desarrollador.

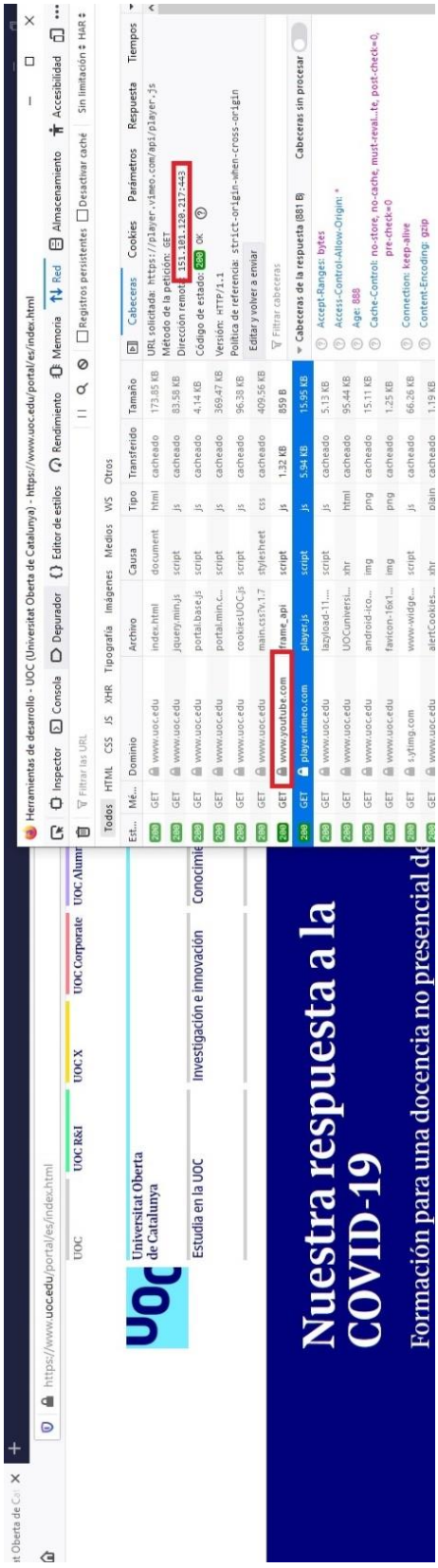


Figura 37: Inspección de la consulta realizada

5. Conclusiones

En este apartado se mostrarán las conclusiones obtenidas una vez se ha realizado el desarrollo del producto, y casi habiendo finalizado el cierre del proyecto.

Las conclusiones obtenidas han sido satisfactorias, por una parte, a la hora de obtener el producto desarrollado para este proyecto, y por otra parte, en lo que a la experiencia obtenida en la realización del proyecto se refiere.

En primer lugar y resumidamente, puesto que se tratará con mayor detalle en el apartado *5.1.Revisión de los objetivos del proyecto*, se ha obtenido un producto de calidad por el principal motivo de que se han cumplido los objetivos definidos para este proyecto, sin olvidar que hay margen de mejora en lo que a ciertos aspectos del producto se refiere (ver *6.Líneas de futuro*).

Además, han sido diferentes las enseñanzas adquiridas a lo largo del desarrollo del proyecto, enriqueciendo de esta manera el conocimiento del desarrollador. Esto se tratará con más detalle en el apartado *5.2.Lecciones aprendidas*.

Para finalizar con las conclusiones, se hará mención de los aspectos más recalcables del control y seguimiento realizado, llevando a cabo un análisis crítico de lo efectuado a lo largo del desarrollo del proyecto en comparación con lo planificado (ver *5.3.Seguimiento y control*).

5.1. Revisión de los objetivos del proyecto

En este puto se presentará un análisis de los objetivos definidos en este proyecto determinando así cuáles se han cumplido y cómo.

✓ **Objetivo 1: Desarrollar un sistema MITM para realizar un análisis de tráfico.**
Se ha cumplido este objetivo establecido haciendo uso de la herramienta mitmproxy en la Raspberry Pi (ver *3.3.Implantación del proxy a modo de MITM*). Como se recoge en el documento la herramienta se ha implantado en modo trasparente pudiendo así realizar un análisis del tráfico mediante el uso de técnicas *man-in-the-middle*.

✓ **Objetivo 2: Generar un punto de acceso con salida a Internet mediante una Raspberry Pi.**
Se ha cumplido este objetivo establecido mediante el uso de los materiales necesarios y siguiendo las pautas establecidas por el fabricante de la propia Raspberry Pi [12], el principal componente de los materiales usados (ver *3.2.Implantación del punto de acceso*).

✓ **Objetivo 3: Buscar una solución a implantar que ayude a tratar el tráfico generado por dispositivos digitales.**

Se ha cumplido este objetivo establecido haciendo uso de la herramienta mitmproxy, la cual se ha valido de la ejecución de un script escrito en Python (ver *3.4.Tratamiento del tráfico*). El script se encargará, entre otras cosas, de tratar las solicitudes provenientes del tráfico generado por los dispositivos digitales, mientras que mediante mitmproxy se obtendrá el mencionado tráfico.

✓ **Objetivo 4: Generar un sistema de alertas que permita obtener información sobre el tráfico sospechoso generado.**

Se ha cumplido este objetivo establecido por medio del script previamente definido, que se encargará de tratar el tráfico generando las alertas en caso de que fuesen necesarias. Mediante la aplicación Telegram, se permitirá visualizar las alertas generadas (ver *3.4.Tratamiento del tráfico*). Dichas alertas se generarán en caso de que el destino de la solicitud coincida con algún valor de las IPs y URLs consideradas como sospechosas, gracias al uso de listas de reputación.

✓ **Objetivo 5: Implantar un escenario donde se haga uso de todos los anteriores objetivos presentados.**

Se ha cumplido este objetivo establecido. El producto desarrollado es muestra de ello, en donde en una Raspberry mediante diferentes implantaciones, recoge un punto de acceso generado, un sistema MITM que realiza un análisis de tráfico y su posterior tratamiento, y finalmente, un sistema de generación de alertas las cuales se podrán visualizar mediante la aplicación Telegram.

✓ **Objetivo 6: Elaborar un informe que recoja el trabajo desarrollado.**

Se ha cumplido este objetivo establecido. Este documento es la prueba de ello, en donde se muestra el trabajo desarrollado a lo largo del proyecto.

5.2. Lecciones aprendidas

Han sido varias las lecciones aprendidas a lo largo del desarrollo del trabajo las cuales se listarán a continuación.

- El conocimiento del autor del proyecto ha aumentado en los temas abordados, siendo estos: la generación de un punto de acceso, la implantación de un analizador de tráfico, la realización de un sistema de tratamiento de tráfico, la generación de un sistema de alertas, y la configuración y uso de diferentes herramientas, programas, técnicas, etc. Gracias a esto, se ha cumplido uno de los objetivos secundarios (ver *1.3.2.Objetivos secundarios*).

- Se ha podido comprobar el bajo coste que supone un sistema de estas características, el cual podrá ser de gran ayuda a la hora de mitigar riesgos en las comunicaciones de los dispositivos con acceso a Internet.

- Se ha podido verificar la flexibilidad que ofrece la implantación de un sistema de estas características, pudiendo instaurarlo en diferentes entornos tales como espacios domésticos, empresas, etc.

- Se han interiorizado que conceptos asociados por el autor a malas prácticas enfocados de otra manera pueden ser de ayuda contra ataques, riesgos o vulnerabilidades. Para ser más exactos y en lo que a este proyecto se refiere, el uso de técnicas *man-in-the-middle*.

5.3. Seguimiento y control

Se mostrarán los aspectos más relevantes obtenidos tras la realización de un seguimiento y control aplicado a lo largo del proyecto.

Como punto de partida, a este proyecto se le han dedicado 278 horas, mientras que la dedicación prevista era de 252 horas (ver *1.5.Planificación*), obteniendo así 26 horas de diferencia.

NOTA: Cabe mencionar que a las horas dedicadas se le han sumado 13 horas que todavía no se han realizado, las cuales estaban estimadas para las tareas Presentación y Defensa (ver *Tabla 2: Horas del proyecto*).

Dada la diferencia entre lo estipulado y lo dedicado, se podrá afirmar que se han obtenido desviaciones en las diferentes tareas del proyecto (ver *Tabla 2: Horas del proyecto*), las cuales se recogerán a lo largo de este punto. Estas desviaciones han sido de dos tipos:

- Desviaciones negativas: Son las desviaciones obtenidas en las tareas en las que se ha necesitado dedicar menor tiempo que el estipulado.
- Desviaciones positivas: Son las desviaciones obtenidas en las tareas en las que se ha necesitado dedicar mayor tiempo que el estipulado.

A continuación se presentarán las desviaciones obtenidas.

Fase 1: Planificación (+3)

Las tareas recogidas en la planificación principal eran muy genéricas, y estas se han ido especificando en mayor grado una vez empezado el desarrollo del proyecto. La desviación surge a consecuencia de la especificación de las mismas, puesto que estas únicamente no se han tenido que documentar en el apartado de tareas (ver *1.5.1.Tareas*), en el diagrama de Gantt (ver *Figura 2: Diagrama de Gantt*) y en la estructura de descomposición del trabajo (ver *Figura 1: Estructura de Descomposición del Trabajo*) también se han tenido que hacer las respectivas modificaciones.

Fase 2: Punto de acceso, elección de la alternativa (-2)

A medida que se iba haciendo el estudio de diferentes alternativas prácticamente ya se estaba haciendo la elección de la misma, puesto que el grado de satisfacción obtenido por diferentes usuarios hacia una de las alternativas era un patrón que se mostraba constantemente, además que el propio fabricante en donde se iba a implantar el punto de acceso hacia recomendación de dicha alternativa, para ser más exactos, la alternativa usada en el desarrollo del proyecto.

Fase 2: Desarrollo, implantación del analizador del tráfico (+3)

La desviación ha surgido en la instalación de los certificados, precisamente, cuando se ha tenido que importar el certificado dentro del almacén de certificados Firefox (ver *Figura 17: Problema de seguridad Firefox, certificado no localizado en el almacén Mozilla CA*), puesto que no se tenía constancia de que Firefox únicamente usase el almacén de Mozilla CA para verificar que la conexión fuese segura, se pensaba que también admitía los certificados proporcionados por el sistema operativo del usuario.

Fase 2: Desarrollo, implantación del punto de acceso (+2)

Al principio, en el desarrollo del proyecto se llego a implantar el punto de acceso en la Raspberry siguiendo el manual del modo puente o *bridged*, con lo que al hacer las pruebas se pudo identificar que realmente no se estaba generado una nueva red local, sino que se estaba operando sobre la misma red con acceso a Internet. La desviación surge a la hora de volver a reconfigurar la Raspberry, implantando esta vez el modo del punto de acceso mostrado en el documento (ver *3.2.Implantación del punto de acceso*).

Fase 2: Desarrollo, implantación del tratamiento del tráfico (+3)

Se tenía nociones de programar en lenguaje Python, pero se desconocía la manera de implementar ciertas funcionalidades usadas en el script escrito para este proyecto, tales como el multihilo, teniendo así que documentarse previamente. La desviación ha surgido por la documentación realizada y la metodología prueba y error aplicada antes de añadir las funcionalidades en el script usado en este proyecto.

Fase 3: Redactar la memoria (+12)

La desviación se debe a que el desarrollador ha necesitado más tiempo del previsto para aclarar los conceptos recogidos en la memoria del TFM.

Fase 3: Revisión enriquecimiento y corrección de la memoria (+5)

La desviación surge prácticamente por lo mismo que se ha mencionado en la desviación anterior, el desarrollador ha necesitado más tiempo del previsto para aclarar mejor los conceptos recogidos en la memoria, incluso en algunos casos corrigiendo dichos conceptos en más de una ocasión.

6. Líneas de futuro

Una vez desarrollado el producto, en este apartado se mostrarán sugerencias y mejoras para futuras versiones de este proyecto o para futuros proyectos de estas características. Estas ideas no se han incluido en el desarrollo de este producto por el principal motivo de la falta de tiempo.

Configuración de la Raspberry

La configuración de la Raspberry llevada a cabo en este proyecto (ver *3.1.Preparación de la Raspberry*) cumple con los requisitos mínimos necesarios para que el proyecto sea exitoso. No obstante, convendría asegurar mejor [15] el propio dispositivo, debido a que en muchos aspectos se ha dejado la configuración por defecto, lo que supondría que el sistema generado podría ser más vulnerable a ataques informáticos.

Herramienta Hostapd

Al hilo de la sugerencia anterior, convendría buscar una alternativa en la configuración de la herramienta Hostapd (ver *3.2.4.Configuración de Hostapd*), puesto que los valores del archivo de configuración de la misma se guardan en texto plano, incluyendo la contraseña de la red que se creará. Si alguien tuviese acceso a dicho archivo podría conectarse a la red sin permiso previo, esto podría suponer dos cosas:

- Que parte del sistema implantado sea accesible para quien no debiera tener acceso, pudiendo así realizar diferente tipos de acciones, como escaneos, ataques informáticos, etc.
- Que existan problemas legales (ver *1.7.Requisitos legales*), puesto que el tráfico generado por el dispositivo que no debiera tener acceso a la red se analizará y se tratará, a priori sin consentimiento, llegando así a incumplir el RGPD.

Certificado del cliente

Tal y como se ha presentado en el apartado *3.3.4.Instalación del certificado* lo ideal será que no haya que hacer ningún tipo de configuración en el lado del cliente, evitando llevar a cabo la instalación del certificado a fin de hacer más transparente el sistema implantado.

Reglas del tráfico

En este proyecto únicamente se trata el tráfico dirigido a los puertos 80 y 443 relacionados con los protocolos HTTP y HTTPS. Puede darse el caso de que un dispositivo de dentro del sistema implantado genere tráfico de red dirigido a otros puertos. Por ejemplo, si un dispositivo conectado a la red generada hace uso de `WhatsApp Business API` proporcionado por la empresa Whatsapp, también podrá generar tráfico en dirección al puerto 5222 [16], tal y como está el proyecto desarrollado dicho tráfico no se trataría, evadiendo así el sistema implantado. Lo lógico será poder tratar todo el tráfico de red que se genere en el sistema implantado.

Generación de alertas

En este proyecto las alertas únicamente se generan para visualizarlas vía Telegram, lo cual tiene sus múltiples ventajas pero una gran desventaja: En caso de que se generen una cantidad considerable de alertas en un breve periodo de tiempo dificultaría la legibilidad de las mismas.

Tal y como se ha presentado en el apartado *2.4.2.Alternativas para la visualización de alertas*, lo ideal será disponer de diferentes alternativas para la generación y posteriormente visualización de alertas. Por ejemplo, y teniendo en cuenta la problemática de Telegram, será conveniente implementar un sistema de logs para posteriormente poder visualizarlos gráficamente mediante un *dashboard*, ya que esta opción facilitaría la legibilidad y visualización de las alertas generadas a gran escala.

Script usado

El script escrito en Python usado en este proyecto cumple con las funcionalidades mínimas exigidas, aun así, cabrá mencionar que existe margen de mejora en los apartados que se presentan a continuación:

- En lo que a la programación del script se refiere, existen dos puntos principales los cuales el autor cree que se deberían tratar.
    - Por una parte el script no dispone de un tratamiento de excepciones. En caso de ocurrir un error inesperado en la ejecución del script, habilitando esta acción, se permitirá tener cierto control sobre el mencionado error.
    - Por otra parte el script no dispone de ningún generador de trazas. Habilitando esta acción, se podrá disponer de información relativa al comportamiento del script.

- Optimizar el script e intentar hacerlo lo más genérico posible. Por ejemplo, las listas de reputación (en este caso por falta de tiempo únicamente se hace uso de una única lista de reputación, aunque lo ideal sería hacer uso de más de una) se obtienen de diferentes maneras, ya sea vía web, desde un archivo, etc. La idea consistiría en recuperar las listas de reputación lo más genéricamente posible a fin de tener el menor código posible cumpliendo una misma funcionalidad.
- Para este proyecto en concreto, a la hora de hacer el tratamiento del tráfico, cuando se percibe que el tráfico analizado podrá dar lugar a algún futuro riesgo únicamente se generará una alerta, posteriormente dejando que el tráfico siga su curso. Lo ideal sería que una vez genere la alerta haga una redirección del tráfico evitando así hacer conexiones, entrantes como salientes, que podrían dar lugar a futuros riesgos.
- En caso de desarrollar más código en el script no estará de más modularlo, clasificando partes del código en diferentes módulos, a fin de hacerlo más legible.

Bibliografía

[1] Clement, J. (7 de Enero de 2020). *Number of internet users worldwide from 2005 to 2019.* Obtenido de https://www.statista.com/statistics/273018/number-of-internet-users-worldwide

[2] United Nations. (2019). *World Population Prospects 2019.* Obtenido de https://population.un.org/wpp/Publications/Files/WPP2019_Highlights.pdf

[3] BOE. (4 de Mayo de 2016). *REGLAMENTO (UE) 2016/679 DEL PARLAMENTO EUROPEO Y DEL CONSEJO de 27 de abril de 2016.* Obtenido de https://www.boe.es/doue/2016/119/L00001-00088.pdf

[4] BOE. (6 de Diciembre de 2018). *Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales.* Obtenido de https://www.boe.es/buscar/act.php?id=BOE-A-2018-16673

[5] *Wireshark*. Recuperado el 5 de Marzo de 2020, de https://www.wireshark.org

[6] Jacobson, V., McCanne, S., & Leres, C. (1987). *Tcpdump.* Recuperado el 7 de Marzo de 2020, de http://daedalus.cs.berkeley.edu/software/pub/tcpsack/bsdi-2.1/tcpdump/tcpdump.c

[7] *Snort*. Recuperado el 10 de Marzo de 2020, de https://www.snort.org/

[8] *Suricata*. Recuperado el 12 de Marzo de 2020, de https://suricata-ids.org/

[9] *mitmproxy*. Recuperado el 13 de Marzo de 2020, de https://mitmproxy.org

[10] Cortesi, A. (14 de Febrero de 2010). *Introducing mitmproxy: an interactive man-in-the-middle proxy.* Recuperado el 22 de Marzo de 2020, de https://corte.si/posts/code/mitmproxy/announce0_1/index.html

[11] *Portswigger Burp Proxy*. Recuperado el 15 de Marzo de 2020, de https://portswigger.net/burp/documentation/desktop/tools/proxy

[12] *Setting up a Raspberry Pi as a Wireless Access Point*. Recuperado el 21 de Marzo de 2020, de https://www.raspberrypi.org/documentation/configuration/wireless/access-point.md

[13] Malinen, J. *hostapd*. Recuperado el 23 de Marzo de 2020, de https://w1.fi/hostapd/

[14] *Dnsmasq*. Recuperado el 23 de Marzo de 2020, de http://www.thekelleys.org.uk/dnsmasq/doc.html

[15] *WhatsApp*. Recuperado el 28 de Marzo de 2020, de https://www.whatsapp.com/

[16] *Telegram*. Recuperado el 28 de Marzo de 2020, de https://telegram.org/

[17] *Telepot*. Recuperado el 28 de Marzo de 2020, de https://telepot.readthedocs.io/en/latest/

[18] *Raspbian*. Recuperado el 1 de Abril de 2020, de https://www.raspbian.org/

[19] *mitmproxy docs*. Recuperado el 9 de Abril de 2020, de https://docs.mitmproxy.org/stable/

[20] *mitmproxy docs API*. Recuperado el 19 de Abril de 2020, de https://mitmproxy.readthedocs.io/_/downloads/en/v2.0.2/pdf/

Referencias externas a modo de ayuda

plantillas [1] : https://ayudaleyprotecciondatos.es/modelos-plantillas/

la página web de las preguntas frecuentes sobre Telegram [2] : https://telegram.org/faq

listas de reputación [3] : http://osint.bambenekconsulting.com/feeds/

El archivo [4] : https://reputation.alienvault.com/reputation.data

lista de reputación [5]: http://rules.emergingthreats.net/fwrules/emerging-Block-IPs.txt

una API [6] : https://www.badips.com/documentation

los eventos [7] : https://docs.mitmproxy.org/stable/addons-events/#http-events

página web oficial de Raspberry [8] : https://www.raspberrypi.org/downloads/raspbian/

Win 32 Disk Imager [9] : https://sourceforge.net/projects/win32diskimager/

El canal [10] : https://en.wikipedia.org/wiki/List_of_WLAN_channels#2.4_GHz_(802.11b/g/n/ax)

proxy en modo transparente [11] : https://docs.mitmproxy.org/stable/howto-transparent/

argumento [12] : https://docs.mitmproxy.org/stable/concepts-options/

`Botfather` [13] : https://core.telegram.org/bots#6-botfather

Kraken [14] : https://www.fireeye.com/blog/threat-research/2008/04/kraken-botnet-1.html

asegurar mejor [15] : https://www.raspberrypi.org/documentation/configuration/security.md

puerto 5222 [16] : https://developers.facebook.com/docs/whatsapp/guides/network-requirements

Mención especial

- Artículo en donde se muestra un proyecto que hace uso de una Raspberry Pi con la herrmienta mitmproxy para generar un proxy MITM.

  Fizzotti, D. (9 de Enero de 2019). *Running a man-in-the-middle proxy on a Raspberry Pi 3*, de https://www.dinofizzotti.com/blog/2019-01-09-running-a-man-in-the-middle-proxy-on-a-raspberry-pi-3/

Anexos

A continuación se mostrarán los anexos referenciados a lo largo del documento.

Anexo A: Script usado en la generación de alertas

```
######## SCRIPT MITM PYTHON ######################
###############
# 1. Modulos #
###############
from mitmproxy import http
import telepot
import threading
from urllib.request import urlopen
import re

#########################
# 2. Variables globales #
#########################
#Se deberá sustituir el valor APIKEY bot Telegram por el valor real
bot = telepot.Bot("APIKEY bot Telegram")

listaIPSospechosas = ["151.101.120.217"]
listaURLSospechosas = ["www.youtube.com"]

# Links listas para las listas de reputacion #

#Feed Provided By: John Bambenek of Bambenek Consulting con licencia CC BY-NC-SA
4.0#
linkBambenek_kraken =
"https://osint.bambenekconsulting.com/feeds/kraken-iplist.txt"

###############
# 3.Funciones #
###############

# Funcion para obtener las listas #
def obtenerContenidoListas():
```

```
 #Leer contenido de la lista de reputacion
 listaReputacionKraken = urlopen(linkBambenek_kraken)
 archivoListaReputacionKraken = listaReputacionKraken.read()

 ## Adaptacion de la informacion obtenida ##
 listaIPBambenek_kraken =
re.findall(r"\b\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}\b",
archivoListaReputacionKraken.decode("utf-8"))

 #Sumar las IPs no repetidas a la lista principal de IPs
 listaIPSospechosas.extend(x for x in listaIPBambenek_kraken if x not in
listaIPSospechosas)


# Funcion que actualizara listas de reputacion consultadas #

def actualizarListas():
#Se obtendran las listas de reputacion
 obtenerContenidoListas()
#Se volverá a llamar a la funcion recursivamente
 threading.Timer(600, actualizarListas).start()


########## FIN #############

# Inicializar por primera vez la funcion de actualizar las listas
actualizarListas()




# Tratar los eventos #
def request(flow: http.HTTPFlow):

 print("Request a tratar")

 for x in range(0,len(listaIPSospechosas)):
      if listaIPSospechosas[x] == flow.request.host:
```

```
            bot.sendMessage(id canal, "WARNING! Request to IP: " +
listaIPSospechosas[x].replace(".",". ") + " done")

#Se deberá sustituir todos los valores id canal por el valor real

 for y in range(0,len(listaURLSospechosas)):
      if listaURLSospechosas[y] == flow.request.pretty_host:
            bot.sendMessage(id canal, "WARNING! Request to URL: " +
listaURLSospechosas[y] + " done")
```