Universidad Zaragoza
1542

Trabajo Fin de Grado

Título del trabajo:

Inyección de tramas Wi-Fi seguras mediante el mecanismo de agregación en un entorno GNS3

English title:

Secure Wi-Fi frames injection through the aggregation mechanism in a GNS3 environment

Autor/es

Jhosep Joel Mendoza Lazo

Director/es

Julián Fernández Navajas

ESCUELA DE INGENIERÍA Y ARQUITECTURA
2021/2022

Resumen

El aumento significativo de las conexiones inalámbricas ha llevado a que los estándares de la IEEE (Institute of Electrical and Electronics Engineers) 802.11 sufran cambios para adaptarse a todo este tráfico nuevo. Las mejoras más significativas se llevaron a cabo en conseguir un mayor alcance, rendimiento superior, estabilidad y funcionamiento óptimo trabajando en entornos multidispositivo.

Entre los estándares más relevantes tenemos 802.11n\ac\ax, los cuales utilizan el mecanismo de agregación de tramas Wi-Fi con el objetivo de alcanzar mayores eficiencias en la transmisión. Esto se consigue a través de la capa MAC[7] la cual tiene la capacidad de agregar tramas para reducir el número de tiempos de espera que requiere cada trama para ser transmitida (denominados vértices temporales), lo cual añade un gran retardo a la conexión al sumarse a lo que se denomina tiempo en el aire de cada trama. Nos centraremos en el modelo de agregación A-MPDU [1], el cual consiste en unir múltiples MPDU con un solo encabezado físico.

En base a lo comentado anteriormente se propone implementar un nuevo modelo de agregación de tramas (desarrollado en el grupo de investigación CeNIT en el que se enmarca el TFG) pero añadiendo un nivel de seguridad extra [3] y con la ventaja de poder ser aplicado en todas las normas y no sólo en aquellas que permiten agregación. Para ello se implementará el cifrado de difusión multicanal de mensajes cortos que brinda un equilibrio entre eficiencia y seguridad. Las mejoras se obtienen reduciendo el tiempo en el aire. Para lograr esta mejora se reduce los encabezados de seguridad en uno solo que se comparte entre todos los receptores, mientras la carga útil se multiplexa a través del Teorema Chino de los restos utilizando claves diferentes para cada uno de los mensajes agregados y sólo quien tenga las claves podrá descifrar su contenido. De esta manera se reduce la longitud del paquete con respecto a la aplicación clásica de la seguridad.

En este documento explicaremos en primer lugar los diferentes estándares de la IEEE 802.11 y en qué consiste la agregación de tramas Wi-Fi. Luego mostraremos cómo poder construir e inyectar tráfico real utilizando cada una de las normas Wi-Fi, además de implementar nuestro modelo de tramas agregadas seguras. Todo ello utilizando GNS3 como herramienta, que permite acceder y controlar las tarjetas Wi-Fi de red físicas de manera remota mediante un escenario virtualizado. Además, utilizaremos python3 como lenguaje de programación para la construcción de las tramas, así como para poder enviarlas y recibirlas a través de un medio inalámbrico. Finalmente utilizaremos este mismo lenguaje para poder realizar la implementación de la propuesta de nuestro grupo de investigación de tramas agregadas seguras.

ÍNDICE

1. Introducción

Estamos viendo que la tecnología está influyendo más que nunca en la vida de las personas. Esto se puede ver con el aumento significativo de dispositivos móviles y portátiles como MacBook, tabletas y teléfonos inteligentes. Lo cual nos da un claro testimonio de la importancia de la tecnología inalámbrica en las comunicaciones de la sociedad moderna. Todo esto ha llevado a que los sistemas inalámbricos sufran muchos cambios a lo largo del tiempo para adaptarse a las necesidades de los usuarios.

Para comprender la evolución de las diferentes normas de Wi-Fi, debemos tener en cuenta que siempre debe asegurar la conexión simultánea de dispositivos con diferentes normas que dependen no sólo del dispositivo sino de la ubicación de este. Un ejemplo notable de estos cambios es la evolución de la velocidad de transferencia de datos. En un principio, por ejemplo, el estándar 802.11b permite una velocidad máxima de 11Mbps en la banda de 2.4 GHz pero los dispositivos pueden situarse alejados entre sí, obteniendo mayores coberturas. En el caso del 802.11a se permite hasta los 54Mbps en la banda de los 5 GHz [2] pero con menos cobertura. Ya que en el 802.11a el coste de fabricación era excesivo, surgió el estándar 802.11g, el cual mantiene la misma tasa máxima del estándar Wi-Fi 802.11a con las diferencias de que los dispositivos son más baratos de fabricar debido a su mayor sencillez y que trabaja en la banda de 2.4 GHz que tiene mayor cobertura.

En 2009 nace el estándar 802.11n o Wi-Fi 4 y es la primera que contempla trabajar en las frecuencias de 2.4 y 5 GHz. Permite la capacidad de transmitir utilizando múltiples antenas lo cual supone una importante revolución ya que puede utilizar hasta 4 antenas entre el transmisor y receptor, lo que mejora considerablemente el rendimiento de la transmisión y la fiabilidad debido a la existencia de diversidad de caminos. Esto, sumado a la disminución de los diferentes tiempos de guarda entre tramas y el uso de nuevas modulaciones consiguió que la velocidad de transmisión se elevará hasta los 600 Mbps, lo cual es 11 veces más que el máximo permitido por el 802.11g. Sólo cuatro años después nacía el estándar 802.11ac o Wi-Fi 5; esta versión permitía la mejor utilización de la diversidad de caminos, una ampliación de ancho de banda y nuevas modulaciones, teniendo como velocidad máxima de transmisión 7 Gbps trabajando en la banda de 5 GHz.

El último estándar aprobado es el 802.11ax o Wi-Fi 6. Esta nueva implementación trae consigo una nueva mejora en robustez, reparto de recursos y velocidades. Sobre todo, esto último, ya que la velocidad máxima aumenta hasta los 10.53 Gbps. Para lograrlo emplea el nuevo sistema MIMO-OFDMA [4] que evita colisiones utilizando mecanismos de reserva de recursos. Al igual que el estándar 802.11n puede utilizar la banda de los 2.4 y 5 GHz.

Por otro lado, también hablaremos de la seguridad dentro de las tramas Wi-Fi, aspecto de vital relevancia al hablar de redes inalámbricas y por tanto de fácil acceso por parte de un posible atacante. Para aportar seguridad se implementó en el año 1999 el estándar WEP (Wired Equivalent Privacy) el cual tiene, como mecanismo de encriptación, RC4 para proteger la información confidencial [5]. Debido a que este protocolo fue roto en diferentes ocasiones, en 2004 nace 802.11i, un estándar aprobado por la IEEE para proteger las WLANs, el cual implementa varios mecanismos de seguridad. Posteriormente, surgió la norma WPA (Wi-Fi Protected Access) y sus posteriores actualizaciones, que es la que utiliza actualmente las redes inalámbricas.

1.1. Problemática

Encontrar un buen equilibrio entre la eficiencia de las comunicaciones y su seguridad (la cual se consigue añadiendo información redundante, provocando la ineficiencia entre datos útiles con respecto al total de tiempo utilizado por los paquetes de longitud pequeña) se convierte en una cuestión difícil, a menudo influenciada por los diferentes requisitos de servicio.

La posibilidad de transmitir tramas agregadas mejora considerablemente la eficiencia [1,2]. Pero las tramas Wi-Fi agregadas en la actualidad pueden sufrir ataques de seguridad, al no poderse aplicar convenientemente el protocolo WPA, por lo que es una vulnerabilidad importante debido a que, si nos ponemos a capturar el tráfico que genera un punto de acceso con los otros dispositivos conectados a él, podremos ver las cabeceras MAC[7] de los dispositivos a los que se le están enviando información dentro de la trama agregada, aunque su contenido esté correctamente cifrado, y de esta manera se abre una brecha de seguridad. Además, el protocolo de implementación de estas tramas dispone de un Block Ack request donde va la confirmación de las tramas que se ha recibido por parte del punto de acceso, por lo que se brinda aún más información de todo el proceso de comunicación [8] y permite planificar un ataque[6].

Otro aspecto a tener en cuenta es que las tramas agregadas envían información sólo a un dispositivo, por lo que si éste no genera tráfico no es posible agregar tramas. Por ello se plantea la opción de poder agregar tráfico con destino a diferentes dispositivos en una sola trama para conseguir un mayor rendimiento y uso del medio.

Finalmente, para poder inyectar tramas de diferentes estándares, poder implementar nuestro modelo de trama agregada segura y hacer los análisis oportunos es necesario tener acceso y control a las antenas Wi-Fi. Pero queremos hacerlo mediante un escenario controlado de laboratorio que nos permita una gestión única, segura y flexible. Por ello vamos a configurar un escenario de red inalámbrica con el uso de GNS3, para controlar tarjetas de red físicas mediante máquinas virtuales Debian10 capaces de compilar código python3, lenguaje necesario para la construcción, inyección y recepción de tramas.

1.2. Objetivos

Nuestro primer objetivo es la configuración de un escenario de red inalámbrica, controlado y fácilmente replicable que permita construir e inyectar tráfico de las diferentes normas IEEE 802.11.
El otro objetivo importante es implementar un nuevo esquema de seguridad capaz de ser aplicado en entornos de agregación y envío de tramas a múltiples terminales simultáneamente con el consiguiente aumento de la eficiencia de la información a transmitir [3].
Para ello, implementaremos el cifrado de difusión multicanal de mensajes cortos que combine el esquema subyacente del cifrado de radiodifusión, pero utilizando de manera más eficiente el espectro. En base a lo comentado se propone agregar los paquetes cifrados de la siguiente manera:

- Un nuevo sistema de seguridad capaz de fusionar una serie de paquetes individuales con destinos diferentes, lo que resulta en una reducción del tamaño total de la información que debe transmitirse.

Y a continuación se aplicará a las redes inalámbricas (802.11) con el fin de aumentar su eficacia mediante la reducción de la cantidad de información que debe enviarse a través del medio inalámbrico. Lo que también permite el análisis sobre el nivel de seguridad obtenido con diferentes tamaños de clave y la eficiencia (en términos de utilización del canal).

1.3. Estructura del documento

La memoria se dividirá de la siguiente manera:

- En la primera parte describiremos más detalladamente los diferentes estándares 802.11 que propone el IEEE, veremos en qué consiste el tráfico agregado y explicaremos nuestra propuesta de tráfico Wi-Fi seguro [3]. Hablaremos de qué es y cómo utilizar la herramienta de control de versiones GitHub para guardar el código implementado.

- En la siguiente parte explicaremos toda la configuración necesaria que se necesita implementar para poder controlar de manera remota las tarjetas de red reales de las antenas para poder inyectar o recibir tráfico Wi-Fi en un entorno de virtualización como GNS3.

- En la siguiente sección explicaremos cómo mediante el uso de la librería Radiotap podemos generar tramas de los diferentes estándares. Además de detallar cómo implementar nuestra propuesta de trama agregada segura. Se explicará el proceso de generación de claves, cifrado y descifrado del teorema de restos chinos.

- En la siguiente parte explicaremos cómo está distribuido el código de GitHub de todo lo que hemos implementado para que nos sirva de referencia a la hora de explicar todo el proceso, después inyectaremos el tráfico de los diferentes estándares mediante la librería de Scapy llamada Sendp. También detallaremos cómo podemos comprobar si el tráfico es el correcto mediante dos métodos. El primero mediante el uso de Tcpdump y el segundo con la librería Sniff implementada en Python3.

- En el siguiente caso inyectaremos paquete utilizando nuestro modelo de trama agregada segura en la que propondremos diferentes casuísticas para explicar de una forma más visual el proceso de cifrado y descifrado de un paquete en todo el proceso de transmisión.

- Finalmente hablaremos de las conclusiones del trabajo y brindaremos un conjunto de ideas futuras que se podrían implementar para mejorar el modelo.

2. Estándares Wi-Fi

2.1. Descripción y características de los protocolos

2.1.1. 802.11 a

Funciona en la banda de frecuencia de 5 GHz. Debido a esto, posee un área de cobertura menor y es menos efectiva de penetrar estructuras. Utiliza 52 subportadoras de multiplexación por división de frecuencias ortogonales (OFDM), ofrece velocidades de 6, 9, 12, 18, 24, 36, 48 y 54 Mb/s [9]. Los dispositivos inalámbricos tienen una única antena para transmitir y recibir señales inalámbricas.

2.1.2. 802.11 b

Tiene unas velocidades de transmisión de 1, 2, 5.5 y 11Mbps. Funciona en la banda de 2,4 GHz con mayor cobertura. Debido al espacio ocupado por el uso del protocolo CSMA/CA, en la práctica, la velocidad máxima de transmisión (para tramas grandes) es de aproximadamente 5.9 Mb/s sobre TCP y 7.1 Mbit/s sobre UDP. Introduce como novedad el método CCK (Complementary Code Keying) con el fin de poder transmitir a 11 Mb/s, que en vez de usar el Código Barker [10], usa series de secuencias complementarias que cuentan con 64 únicas palabras posibles. Este estándar tiene que afrontar problemas de interferencia debido al ruido de hornos microondas, dispositivos Bluetooth y teléfonos móviles que trabajan a esta frecuencia.

2.1.3. 802.11 g

Totalmente compatible con la norma “b” presenta un avance significativo en cuanto al ancho de banda, proporcionando un menor consumo y un mayor alcance que 802.11a porque usa la banda de 2,4 GHz.

Implementa una modulación OFDM, y permite tasas de transferencia de 6, 9, 12, 18, 24, 36, 48 y 54 Mbps. Pudiendo funcionar con CCK para 5.5 /11 Mbps y DBPSK/DQPSK+DSSS para ratios de 1 y 2 Mbps [9]. La nueva norma “g” sigue teniendo los mismos problemas de interferencias que tiene la “b”.

2.1.4. 802.11 n

El propósito de este estándar es aumentar la capacidad de transmisión. Es un estándar que acorta los tiempos de espera y utiliza múltiples antenas y mayores anchos de banda en el canal, Esto permite un aumento significativo en la máxima transferencia de datos: de 54 Mb/s a 600 Mb/s. La cantidad cada vez mayor de dispositivos inalámbricos hace que sea cada vez más difícil usar la banda de 2,4 GHz debido a la congestión inalámbrica donde todos los dispositivos conectados compiten por el ancho de banda. Esta norma puede utilizarse en las dos bandas de frecuencia: 2,4 GHz o 5 GHz dependiendo del plan de coberturas.

Implementa una nueva tecnología denominada MIMO, la cual utiliza múltiples antenas para transmitir de forma coherente mayor cantidad de información que utilizando una sola antena al conseguirse mayor potencia tanto en transmisión como en recepción. Además, utiliza la tecnología SDM (Spatial Division Multiplexing) para que varios flujos de datos espaciales independientes se transmitan simultáneamente en un mismo canal espectral y así evitar colisiones.

La transmisión de datos con una velocidad de hasta 600 Mb/s se logra mediante cuatro flujos espaciales utilizando un solo canal con un rango de ancho de banda de 40MHz. Varios tipos de modulación y codificación se han determinado por los estándares y se han guardado en el esquema de modulación y codificación (MCS – Modulation and Coding Scheme) [11]. Con el fin de obtener la máxima eficiencia de 802.11n, es aconsejable una red de 5 GHz. La banda de 5 GHz tiene un potencial considerable debido a los numerosos canales de radio que no se solapan y menores distorsiones e interferencias con respecto a la banda de 2,4 GHz.

	IEEE 802.11a	**IEEE 802.11b**	**IEEE 802.11g**	**IEEE 802.11n**
Frequency band	5.7 GHz	2.4 GHz	2.4 GHz	2.4 / 5 GHz
Average Theoretical speed	54 Mbps	11 Mbps	54 Mbps	600 Mbps
Modulation	OFDM	CCK modulated with QPSK	DSSS, CCK, OFDM	OFDM
Channel bandwidth	20 MHz	20 MHz	20 MHz	20 / 40 MHz
Coverage radius	35 m	38 m	38 m	75 m
Unlicensed spectrum	Yes (it depends on countries)	Yes	Yes	Yes (it depends on countries)
Radio Interference	Low	High	High	Low
Introduction cost	Medium-Low	Low	Low	High-medium
Device cost	Medium-Low	Low	Low	Medium
Mobility	Yes	Yes	Yes	Yes
Current use	Medium	High	High	High
Security	Medium	Medium	Medium	High

Figura 1: Comparación de detalles técnicos de los estándares a, b, g y n

En la figura 1 vemos la comparación de los detalles más importantes a la hora de utilizar un determinado estándar, esta figura será de mucha utilidad a la hora de modificar un campo específico para crear tramas de diferentes estándares.

2.1.5. 802.11 ac

El estándar 802.11ac únicamente trabaja en la banda de 5 Ghz y utiliza la tecnología Multi-User MIMO (MU-MIMO) mejorando la capacidad de compartir el ancho de banda con múltiples usuarios, mediante la tecnología estándar de conformación de haces. La principal ventaja es, en definitiva, una mayor velocidad de transferencia de las conexiones inalámbricas.

IEEE 802.11n	IEEE 802.11ac
hasta 4 flujos espaciales	hasta 8 flujos espaciales
3 flujos – máxima tasa de bits 450 Mbit/s	3 flujos – máxima tasa de bits 1300 Mbit/s
canales de 20 y 40 MHz	canales de 20, 40, 80, e incluso 160 MHz
modulación por flujo 64-QAM	modulación por flujo 256-QAM
MIMO	Multi-user MIMO
	agregación de frames extendida
	conformación de haces (beamforming)

Figura 2: Comparación de detalles técnicos.

La figura 2 representa perfectamente las principales diferencias entre el estándar 802.11n y 802.11 ac.

2.1.6. 802.11 ax

Aumenta la capacidad de rendimiento hasta cuatro veces con respecto a la de 802.11ac, utilizando las bandas de 2,4 y 5 GHz, implementa al igual que 802.11ac el sistema de múltiples antenas MU-MIMO, tiene como característica principal el uso de OFDMA que son las siglas, en inglés, de acceso múltiple por división de frecuencias ortogonales. Permite a los canales subdividirse para ofrecer paso a diferentes usuarios y dispositivos [2] evitando de esta manera la pérdida de eficiencia en el proceso de detección de la disponibilidad del canal.

Otra novedad es la mejora de la red cuando hay muchos dispositivos conectados mediante la coloración BSS. Se trata de una técnica de reutilización especial que usa marcas o "colores" para identificar cada red. Los puntos de acceso usan estos "colores" para tomar decisiones sobre si está permitido el uso simultáneo del medio inalámbrico o no, reduciendo el problema de interferencias cuando muchas redes vecinas están usando el mismo canal [12].

2.2. Agregación de tramas (A-MPDU, A-MSDU)

2.2.1. Explicación del mecanismo de agregación

Uno de los avances de la capa MAC a partir de la norma 802.11n es la capacidad de agregar tramas para reducir los tiempos fijos por trama que perjudican significativamente el rendimiento de las redes 802.11[1]. Los estándares que implementan este mecanismo son el 802.11n, 802.11ac y 802.11ax. Existen tres formas de agregación, la unidad de datos de servicio MAC agregada (A-MSDU), la unidad de datos de protocolo MAC agregada (A-MPDU) y la agregación híbrida A-MSDU / A-MPDU (A-hybrid) [1].

Los mecanismos de agregación de tramas son útiles a nivel de capa MAC para reducir los gastos generales temporales y, en consecuencia, aumentar el rendimiento alcanzable y utilización del ancho de banda. Sin embargo, cuando la longitud de la trama MAC agregada es muy grande, es conveniente utilizar una velocidad de datos física más alta para reducir el tiempo de posibles errores y colisiones de esta trama agregada.

Además, por motivos del tráfico habitual de las redes, no siempre es posible disponer de tramas para poder ser agregadas, pero esta situación mejora con el uso de A-MPDU. El concepto de agregación A-MPDU es unir múltiples MPDU, que pueden tener como destino, diferentes equipos terminales con un solo encabezado físico (PHY) (de esta forma tendremos más tramas que pueden ser agregadas porque sumamos los tráficos a múltiples destinos. El número máximo de MPDU que puede contener es de 64, debido a que para confirmar las tramas que han llegado correctamente al receptor se utiliza un campo de mapa de bits llamado Block ACK el cual tiene una longitud máxima de 128 bytes, donde cada MPDU se mapea usando dos bytes. La estructura básica se muestra en la figura 3. Se inserta un conjunto de campos, conocido como cabecera MPDU antes de que cada MPDU y los bits de relleno variados de 0 a 3 bytes se agreguen en la cola. La operación básica del encabezado MPDU es definir la posición y longitud MPDU dentro del A-MPDU. El campo Comprobación de redundancia cíclica (CRC) del encabezado MPDU se utiliza para verificar la autenticidad de los 16 bits anteriores. Después de recibir el AMPDU, se inicia un proceso de desagregación. Primero comprueba el encabezado MPDU en busca de errores basados en el valor CRC, principal diferencia respecto al A-MSDU en cual no tiene un control de errores para cada subtrama. Si es correcto, se extrae la MPDU y continúa con la siguiente MPDU hasta que llega al final de la Unidad de Datos de Servicio PHY (PSDU). De lo contrario, comprueba cada cuatro bytes hasta que localiza un encabezado MPDU válido o el final del PSDU. El delimitador tiene un patrón único para ayudar al proceso de desagregación mientras se escanea el encabezado MPDU.

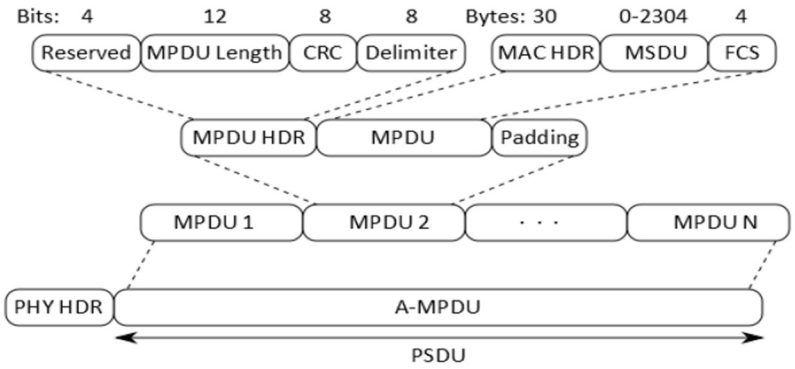

Figura 3: Estructura de trama A-MPDU

2.3. Tramas agregadas Wi-Fi seguras

Las tramas en la actualidad utilizan el modelo de cifrado de WPA2, pero esto tiene un problema con la eficiencia, debido a que supone agregar encabezados de seguridad y esta sobrecarga afecta a la transmisión de datos. Para compensar la pérdida de eficiencia por parte de la seguridad, lo ideal es utilizarla en tramas agregadas, pero, por motivos de tráfico, no siempre es posible disponer de estas tramas. Debemos tener en cuenta que sólo podemos agregar tramas que vayan a un mismo destino y esto aumenta el número de tramas individuales que deben ser transmitidas, con la consiguiente pérdida de eficiencia.

Por ello a continuación se propone un modelo que mejora la relación de seguridad y eficiencia. Se agregará varios paquetes pequeños cifrados asegurando que cada sub-paquete sólo pueda ser descifrado por aquel que conozca la clave correspondiente. De esta forma se pueden agregar, de forma segura, tramas que no tengan como destino el mismo terminal y así aumenta la disponibilidad de tramas que pueden ser agregadas de forma segura.

2.3.1. Explicación del cifrado de difusión multicanal de mensajes cortos

Tomando como ejemplo un sistema clásico de transmisión segura, se utilizan el siguiente modelo:

$$(u_1, u_2, .., u_n, Hdr, c_1, c_2 \ldots, c_n)$$

Donde u_i son una serie de encabezados individuales necesarios para construir la clave privada del usuario para el descifrado los cuales generalmente se envían juntos en una sola trama multiplexada con un campo común Hdr, para que luego se apruebe el contenido cifrado de cada usuario c_i.

La propuesta del grupo de investigación [3] se basa en el siguiente modelo:

$$(Hdr', c_1, c_2 \ldots, c_n)$$

Consiste en fusionar todas las cabeceras individuales u_i, juntamente con Hdr, para obtener una sola cabecera Hdr'. Esto puede reducir la cantidad total de información a transmitir, proporcionando ahorros reales en términos de ancho de banda. Al mismo tiempo, sólo el destinatario legítimo “i” de cada paquete podrá descifrar la información u_i, utilizando Hdr' y su propia clave privada.

Para implementarlos, mejoramos la eficiencia de las comunicaciones, reduciendo los encabezados de seguridad a uno solo, como lo hemos explicado anteriormente, que será compartido por todos los receptores, mientras que la carga útil se multiplexa a través de Chinese Remainder Theorem. De esta manera reducimos la longitud del paquete (menos encabezados) y establecemos la relación entre el texto cifrado/texto plano igual a uno, si no tenemos en cuenta el relleno y los encabezados de seguridad.

Procederemos a describir formalmente la construcción para el cifrado de difusión de mensajes cortos:

- Arreglo(λ):Se toma como entrada el parámetro de seguridad λ ,se generan los parámetros globales de la siguiente manera: primero el algoritmo aleatorio selecciona n primos p_i (uno por usuario), y aleatoriamente $x_i \epsilon\ z_{p_i}^*$,tal que $(x_i, p_i - 1) = 1$.Luego se establece $N = \Pi_{i=1}^{n} pi$ y $EK = ((p_1, x_1), (p_2, x_2), .. (p_n, x_n))$, donde cada par de este conjunto es la clave secreta de descifrado para cada usuario que les será enviado por el algoritmo de extracción.

- Cifrado:$(u_1, u_2, .., u_n, m_1, m_2, .., m_n, EK)$, se selecciona un número aleatorio $Hdr \leftarrow Z^*_{min\, p_j}$, y se define $Hdr_i = min\{g \geq Hdr\ tal\ que\ g\ es\ generado\ por\ \mathrm{Z}^*_{p_i}\}$, luego se establece $ET = (\sum_{i=1}^{n}(m_i + Hdr_i^{x_i}(mod\ p_i))\frac{N}{p_i}[(\frac{N}{p_i})^{-1}(mod\ p_i)])(mod\ N)$.dando las siguientes salidas (Hdr, ET).
- Descifrado:(Hdr, ET, i, p_i, g_i): este algoritmo calcula $m_i = (ET - Hdr_i^{x_i})(mod\ p_i)$, el resultado proviene del teorema chino de los restos.

2.3.2. Eficiencia en la utilización del canal

Una vez presentadas las soluciones propuestas en el apartado anterior, estudiamos analíticamente la compensación de eficiencia en un escenario inalámbrico: el IEEE 802.11 (Wi-Fi) estándar.

Dadas las necesidades de comunicación de paquetes pequeños en términos de seguridad y eficiencia, las restricciones implícitas por el uso del cifrado de difusión multicanal limitan principalmente el tamaño de las claves utilizadas. Nuestro principal objetivo es encontrar un equilibrio entre eficiencia y seguridad que suelen convertirse en cuestiones antagónicas.

Por lo tanto, nuestra propuesta es más eficiente para 802.11 cuando el tamaño de los paquetes del usuario tenga un límite superior. El ahorro será óptimo si el tamaño de la clave es sólo ligeramente superior al tamaño del paquete. Por lo tanto, en los servicios que envían paquetes del mismo tamaño (por ejemplo, VoIP), se puede seleccionar fácilmente una clave óptima.
En un sistema inalámbrico también es muy importante tener en cuenta el mecanismo de acceso al medio [13]. Si tenemos que esperar a que el medio esté libre, cuando tengamos que enviar muchos paquetes a través de él, acumularemos mucho tiempo de espera.

Como hemos visto, los estándares que incluyen el mecanismo de agregación son 802.11n/ac/ax para el envío de tramas multidifusión, aunque nuestra propuesta puede aplicarse a todos los estándares.

En base a esto se hizo un estudio teórico [3] de la propuesta de las tramas A-MSDU seguras respecto a las tramas A-MPDU seguras, en términos de eficiencia en el enlace descendente que se ve reflejada en la figura 4. El tamaño de cada uno de los paquetes agregados está determinado por el tamaño de la clave: por ejemplo, 128 bytes para la clave de 1024 bits. Se utilizan tres tamaños de clave diferentes (1024, 2048 y 4096 bits). Se agregan diferentes números de paquetes UDP (eje X).

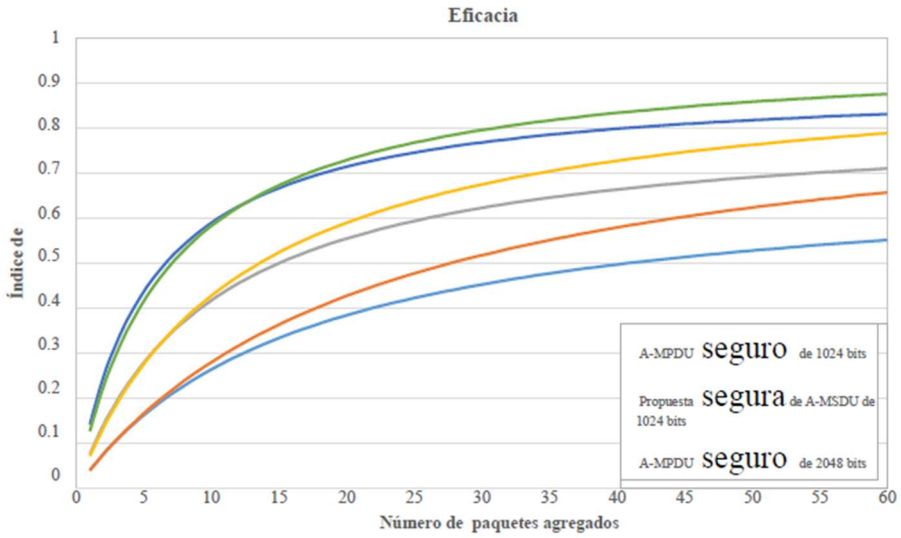

Figura 4: Relación de utilización del canal con paquetes de diferentes tamaños

Podemos ver cómo la eficiencia general crece cuando se aumenta el número de paquetes agregados o el tamaño promedio de paquetes de bits de largo. Se puede observar que, en general, la propuesta de tramas seguras basada en A-MSDU multidifusión supera a la basada en A-MPDU.

Una vez visto el estudio teórico se procederá a implementar un entorno de trabajo en el cual mediante el lenguaje de programación Python3 crearemos un paquete A-MPDU cifrado de manera normal y otro aplicando el modelo de trama agregada segura [3].

2.4. Repositorios en GitHub

GitHub es una herramienta remota de control de versiones en la cual podemos alojar proyectos de tal manera que varias personas pueden tener acceso a ellas y modificar partes de esta. Para ello se utilizan diferentes ramas de la versión principal en las cuales se pueden hacer las modificaciones antes de implementarlas a la rama principal lo que permite tener un mejor control del código y evitar fallos en la implementación.

En este trabajo añadiremos todo el código que se ha realizado en repositorios de GitHub para que puedan ser descargado fácilmente en el caso de que se quiera implementar o saber todo el funcionamiento de una forma más precisa.

Los repositorios implementados se dividirán en dos ramas. La primera consiste en todo el código en Python3 para inyectar tráfico Wi-Fi utilizando los diferentes estándares, al momento de compilar el código se podrá elegir con qué estándar y a qué velocidad se quiere transmitir. En la segunda rama se implementa el código para inyectar tramas utilizando el modelo de agregación normal o el de nuestra propuesta [3]. En el caso del modelo de agregación normal se transmitirán una serie de tramas de control antes de enviar la de datos[8]. Para nuestro modelo se proporciona el código para cifrar 5 paquetes e inyectarlas, y en el caso de recepción se tendrán 2 ficheros con los cuales se podrán capturar y descifrar paquetes. En el apartado 5 se explicará con más detalle todo lo que permite hacer el código.

3. Configuración del escenario GNS3

El uso de la herramienta GNS3 se debe a la necesidad de poder controlar varias tarjetas de red reales, utilizando un escenario virtual controlado que disponga de equipamiento para inyectar tráfico. Entre las muchas cosas que nos permite realizar GNS3, destacamos el despliegue de una infraestructura de red que permita el uso de las tecnologías de control SDN para Wi-Fi [15]. En el que se utilizan puntos de accesos virtuales que pueden ser configurados y organizados. Además, debe permitir la réplica de escenarios reales en entornos controlados lo cual nos servirá de gran ayuda a la hora de inyectar tráfico inalámbrico.

A continuación, procederemos a explicar la configuración e implementación necesarias para poder inyectar y recibir tráfico Wi-Fi.

3.1. Detalles físicos

Detallamos los equipos que utilizaremos para la inyección de tráfico Wi-Fi y aquellos donde serán virtualizados para su posterior uso con GNS3.

Antena RTL8812AU: El Realtek RTL8812AU-CG es un chip único altamente integrado que admite soluciones 802.11ac con un controlador de interfaz USB de LAN inalámbrica (WLAN). Combina una WLAN MAC, una banda base WLAN compatible con 2T2R y RF en un solo chip [14]. El RTL8812AU-CG proporciona una solución completa para un dispositivo inalámbrico integrado de alto rendimiento. El tipo de tráfico inalámbrico que se puede inyectar son:

- QFN-76 package
- 802.11ac/a/b/g/n
- 802.11ac 2x2

Figura 5: Antena RTL8812AU Realtek

En la figura 5 vemos como es la antena que utilizaremos para inyectar y recibir tráfico Wi-Fi.

Intel NUC NUC5i3RYH: Contiene un procesador Intel Core i3-5010U de quinta generación, posee 4 puertos USB3.0, un sensor infrarrojo, un conector para auriculares/micrófono. Interfaces de video Mini HDMI y mini DisplayPort. Este es el hardware donde se ubicará el servidor Host GNS3, el cual se explicará más afondo en el siguiente apartado. Dispone del sistema operativo Ubuntu Server versión 20.04.

En la figura 6 vemos una imagen del equipo Intel NUC5i3RYH.

Figura 6: Intel NUC5i3RYH

3.2. Mecanismo de virtualización

Para virtualizar un equipo es necesario un hipervisor, el cual crea máquinas virtuales mediante el reparto lógico de los recursos de la máquina anfitrión. En este trabajo utilizaremos KVM y QEMU como hipervisores. Esto permite emular por completo una CPU virtual, de esta manera el que se encarga de recibir las instrucciones de dirigidas a la vCPU (Virtual Central Processing Unit) y traducirlas para su ejecución por la CPU física es el hipervisor.

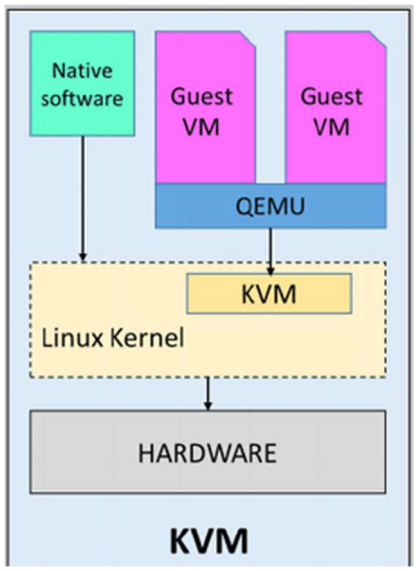

Figura 7: Arquitectura KVM y QEMU

En la figura 7 podemos ver cómo funcionan KVM/QEMU, mediante el uso de estas herramientas podremos lanzar máquinas virtuales con Debian 10 minimal, sistema operativo elegido por su poco tamaño, facilidad para instalar y ejecutar programas en Python3. El hardware que se utiliza es el Intel NUC5i3RYH.

GNS3 utiliza dos componentes para su funcionamiento, el cliente GNS3 que presenta la interfaz gráfica desde la que se diseñan y gestionan las diferentes tipologías, y varios servidores GNS3, que se ocupan de alojar y correr las diferentes instancias virtuales para la implementación.
A continuación, explicaremos los servidores que hemos utilizado:

- Servidor local: Alojado en la máquina del cliente e imprescindible para la utilización de GNS3 en el cliente, en este caso se utilizó sobre el sistema operativo Windows 10.
- GNS3 VM: GNS3 proporciona un servidor óptimo instalado sobre el sistema operativo Linux. Si utilizamos un cliente sobre Windows, se instala este servidor sobre una máquina virtual alojada en la máquina real donde se encuentra el cliente y el servidor local
- Servidor remoto: Son servidores GNS3, generalmente sobre Linux, que se ubican en equipos remotos donde estarán alojadas las máquinas virtuales con los controladores de las antenas físicas reales. En nuestro caso serán máquinas con sistema operativo Debian minimal 10. Estos servidores deberán estar alojados en máquinas con direcciones accesibles IP.

Una vez configurados los diferentes servidores [15], procedemos a crear templates, los cuales nos permiten vincular un archivo de imagen virtual que en nuestro caso es el sistema operativo Debian10 minimal con determinados parámetros de arranque para ser cargados en cualquiera de los servidores anteriormente descritos.

Para poder acceder a los servidores remotos necesitamos tener acceso a la red del laboratorio de telemática. Debido a que es ahí donde se encuentran las antenas por lo que en el Anexo 1.1 se explicará cómo se ha realizado la conexión.

Otra configuración importante es la de poder controlar, desde GNS3, las interfaces de red reales de las antenas y para ello necesitamos agregar en opciones de configuración avanzada de las máquinas virtuales Debian10 que controla cada antena, definidas en GNS3, el siguiente comando:

-device ich9-usb-ehci1, id= usb, bus=pci.0 -device usb-host, hostbus=Nbus, hostaddr=Ndevice, vendorid=0xAAAA, productid=0xBBBB, id=hostdev0, bus=usb.0

En la figura 8 podemos ver, al ejecutar el comando “lsusb” en la máquina real, que tenemos 3 antenas del mismo tipo conectadas al Host GNS3 [15], para identificar cada una utilizaremos los atributos bus, device y el ID. Los cambios que haremos respecto al comando anterior son: El número de bus se ingresará dentro del campo hostbus, el device en el hostaddr, el vendorid son los primeros 4 dígitos del ID y finalmente el productid son los otros 4.

```
proyecto@proyecto-NUC1:~$ lsusb
Bus 001 Device 002: ID 8087:8001 Intel Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 003 Device 005: ID 0bda:8812 Realtek Semiconductor Corp. RTL8812AU 802.11a/b/g/n/ac 2T2R DB WLAN Adapter
Bus 003 Device 010: ID 0bda:8812 Realtek Semiconductor Corp. RTL8812AU 802.11a/b/g/n/ac 2T2R DB WLAN Adapter
Bus 003 Device 003: ID 0bda:8812 Realtek Semiconductor Corp. RTL8812AU 802.11a/b/g/n/ac 2T2R DB WLAN Adapter
Bus 003 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 002 Device 005: ID 8087:0a2a Intel Corp.
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

Figura 8: Lista de dispositivos USB conectados

En la figura 9 podemos observar la configuración final de un template. En el Anexo 1.2 se explica detalladamente el proceso de configuración de un template.

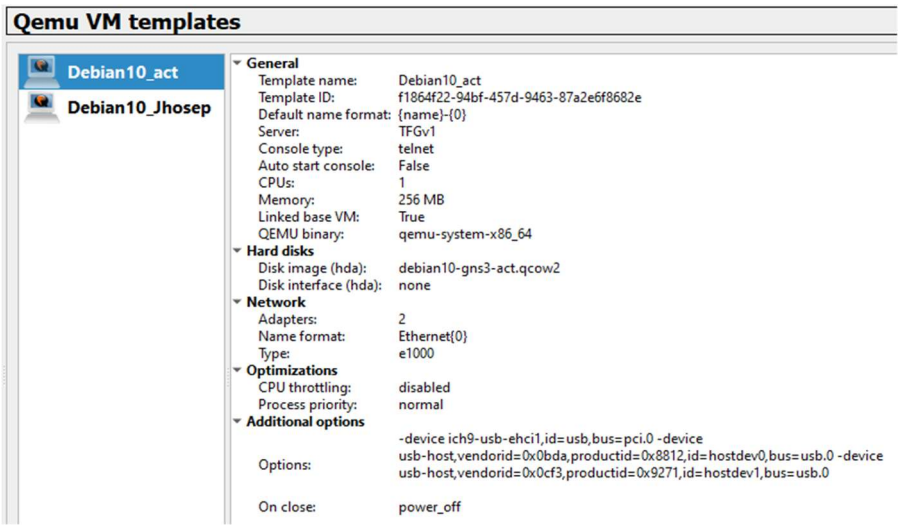

Figura 9: Resumen de detalles de template

3.3. Configuración interna de las máquinas virtuales

3.3.1. Configuración de red

Se requiere salida al exterior para descargar los paquetes y repositorios necesarios. Por lo que necesitamos configurar la red de las máquinas virtuales Debian10, en la figura 10 vemos un ejemplo de cómo podemos configurarla.

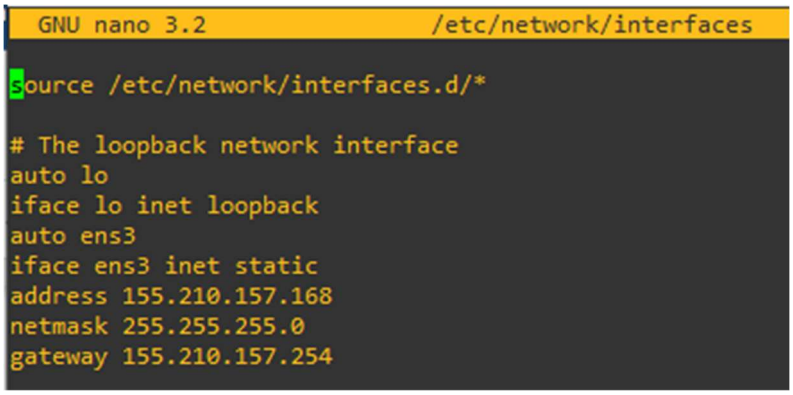

Figura 10: configuración de interfaz

Para validar el cambio es necesario reiniciar la interfaz mediante el comando:

```
/etc/init.d/networking restart
```

3.3.2. Configuración de dirección MAC

Esta configuración es muy importante ya que nos permite asignar una dirección MAC a una interfaz que no fuera la de la tarjeta de red física. Lo que nos permitirá contestar a otros ACK (confirmación de mensajes recibidos) que van destinados a otros dispositivos.

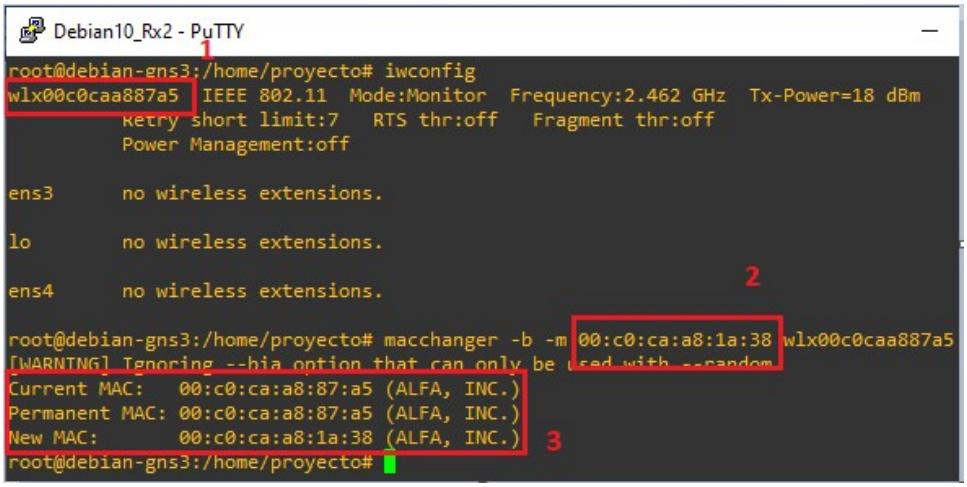

Figura 11: cambio de dirección MAC

En la figura 11.1 vemos nombre de la tarjeta de red donde la dirección MAC es 00:c0:ca:a8:87:a5, en la figura 11.2 ingresamos el comando para cambiar la dirección por una nueva y finalmente en la figura 11.3 se produce el cambio de direcciones.

3.3.3. instalación de Python3

Una parte del trabajo es la utilización de Python3 como herramienta de programación para elaborar el código que sea capaz de inyectar y recibir tráfico Wi-Fi. Para ello, en el Anexo 1.3, se explicará con más detalle cómo instalarlo y los diferentes repositorios que se utilizarán dentro del sistema operativo.

3.3.4. Configuración de las interfaces Wi-Fi

Selección de modo monitor:

El modo monitor también es conocido como modo de escucha o modo promiscuo, en este modo de funcionamiento la tarjeta Wi-Fi se encargará de escuchar todos y cada uno de los paquetes que hay en el «aire» y tendremos la posibilidad de capturarlos con diferentes programas. Por estas razones pondremos todas las antenas en este modo.

Selección de canal:

Hemos visto que los diferentes estándares transmiten a 2,4 o 5 GHz. Dentro de la primera frecuencia están los canales de 1 a 13 que son permitidos para todo el mundo excepto para norte américa, para la segunda frecuencia están los canales 32,34,36,38, etc. [16]. Esta selección se llevará a cabo dependiendo del estándar en el que queramos transmitir.

Antes de poder ejecutar el código para inyectar o escuchar el tráfico inalámbrico es necesario configurar la interfaz mediante los siguientes comandos:

- iwconfig: (Visualizar los nombres interfaces inalámbricas)
- ifconfig nombreInterfaz up: Habilitar la interfaz
- iwconfig nombreInterfaz mode monitor: Cambiar a modo monitor
- iwconfig nombreInterfaz channel n: Seleccionar el canal donde transmitir.

Esta configuración se realizará en cada una de las interfaces inalámbricas que formen parte de la comunicación.

3.4. Escenario final de pruebas

Hemos planteado nuestro escenario final de pruebas de la siguiente manera:

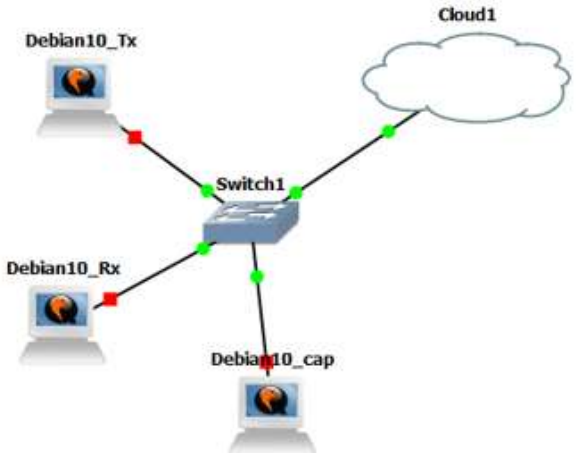

Figura 12: Escenario en GNS3

Además de los beneficios mencionados con anterioridad a la hora de utilizar GNS3 como herramienta de virtualización, otro detalle muy relevante es la capacidad de poder utilizar diferentes ubicaciones lo cual nos brinda mucha flexibilidad a la hora de asignar direcciones con las cuales trabajar. Sin embargo, para una mejor comprensión del escenario se han asignado las siguientes direcciones fijas:

- Debian10_Tx (AP- Access Point): Lo utilizaremos como punto de acceso.
  - Interfaz IPv4 :155.210.157.168
  - Dirección MAC:00:c0:ca:a4:73:7c
  - Interfaz inalámbrica: wlx00c0caa4737c

- Debian10_Rx (STA - Station): Lo utilizaremos como estación.
  - Interfaz IPv4 :155.210.157.167
  - Dirección MAC:00:c0:ca:a4:73:7b
  - Interfaz inalámbrica: wlx00c0caa4737b

- Debian10 Cap (Capturar tráfico – STA 2): Para poder capturar todo el tráfico que se envían el AP y STA. También será utilizada como segunda estación.
  - Interfaz IPv4 :155.210.157.166
  - Dirección MAC:00:c0:ca:a8:87:a5
  - Interfaz inalámbrica: wlx00c0caa887a5

- Cloud 1: Permite la salida al exterior.

4. Uso de Scapy para transmitir

Scapy es una completa librería interactiva de manipulación de paquetes construida para Python 3. Es capaz de construir o decodificar paquetes de una gran cantidad de protocolos, enviarlos por cable, capturarlos, emparejar solicitudes y respuestas, y mucho más. Puede manejar fácilmente la mayoría de las tareas clásicas como escaneo, trazado de ruta, sondeo, pruebas unitarias, ataques o descubrimiento de redes [18].

4.1. Explicación del funcionamiento de Radiotap

Radiotap es un estándar para la inyección y recepción de tramas en 802.11 que funciona para distintos sistemas operativos [17]. La cabecera Radiotap permite obtener información adicional acerca de las tramas recibidas (canal, potencia, etc. que se extrae de la tarjeta wifi y de la cabecera física) y especificar los parámetros de las tramas a enviar. Esta información de Radiotap se puede construir mediante programas desarrollados en Python 3.

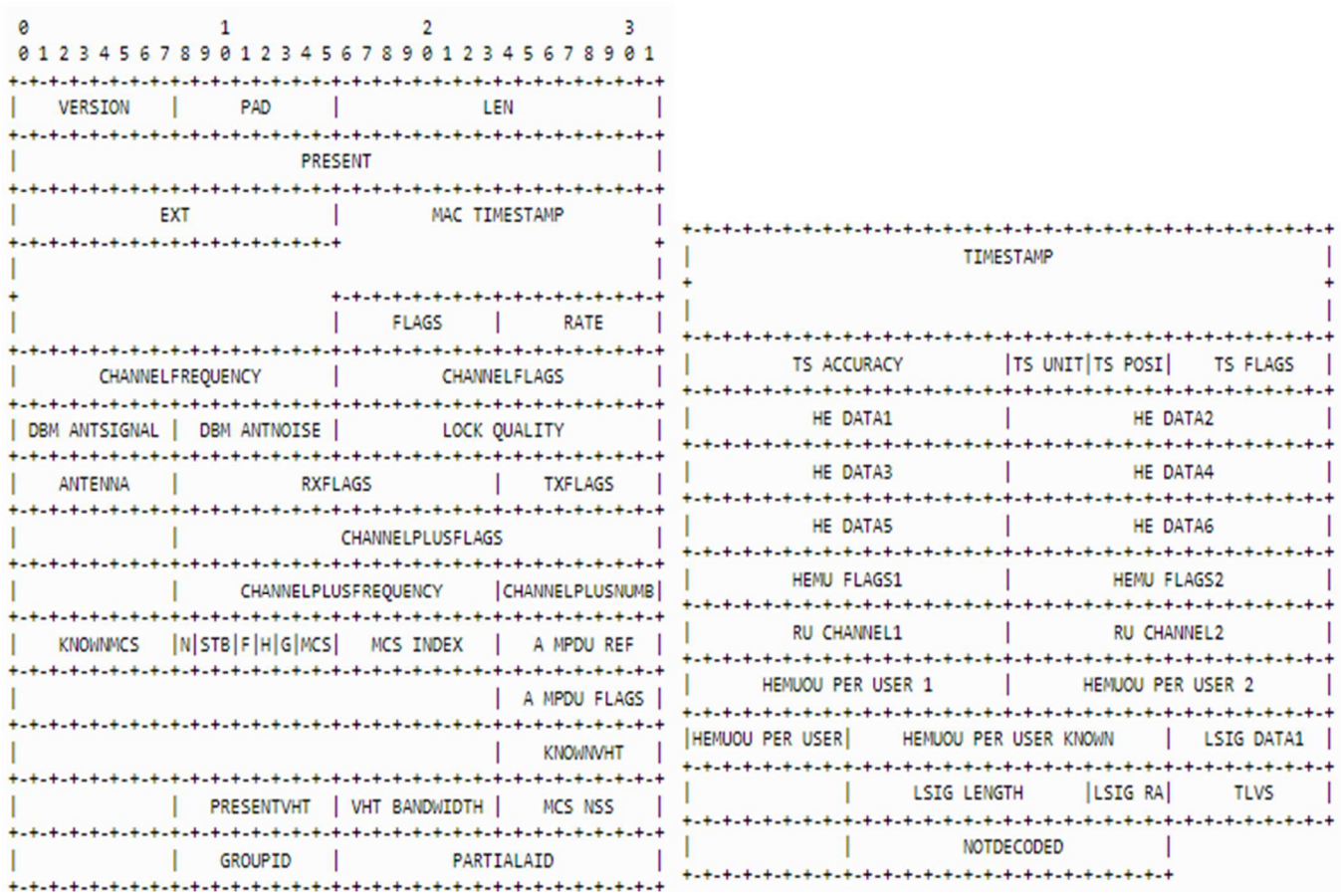

Figura 13: estructura de la cabecera Radiotap

En la figura 13 vemos todos los campos que podemos modificar utilizando Radiotap, cada estándar tiene reservado un grupo de campos en específico.

4.2. Creación de tramas para los diferentes protocolos

La cabecera Radiotap, dependiendo de la frecuencia en la que se esté, tiene definidas una serie de campos establecidos por defecto. Por ejemplo, si se está en la frecuencia de los 5Ghz el tráfico que se envíe, si no se modifica nada de la cabecera, utilizará el estándar 802.11a. Si se está en la frecuencia de 2,4Ghz se envían tramas de 802.11b con la velocidad de transmisión mínima. En los siguientes apartados sólo se modificarán los campos necesarios para inyectar tramas de un determinado estándar teniendo en cuenta que la mayoría de los campos están definidos por defecto.

Otro detalle importante es que Radiotap está programado de tal manera que cuando se crea una cabecera con estructura incorrecta es posible que la trama se envíe, pero usando los parámetros por defecto y no los deseados. Por lo que hay que asegurarse de que los valores que asignamos a los campos sean los correctos para ese estándar.

En los siguientes apartados explicaremos cómo se deben construir las cabeceras Radiotap para inyectar tráfico de los diferentes estándares y cómo podemos modificar valores como la modulación y la velocidad de transmisión.

4.2.1. 802.11a

Las características más relevantes de este estándar es que trabaja en la frecuencia de los 5Ghz y tiene como modulación OFDM.

Para modificar esos valores necesitamos poner en el atributo “present” el nombre de los campos que deseo modificar.

El campo “Channel Flag” nos permite modificar los parámetros que tiene relación con la frecuencia de trabajo y la modulación con la que se quiere transmitir, algunos de los valores para este campo son los siguientes ejemplos:

- CCK (Complementary Code Keying)
- OFDM (Orthogonal Frequency-Division Multiplexing)
- 2 GHz
- 5 GHz
- Dynamic CCK-OFDM
- GFSK (Gaussian Frequency Shift Keying)

El campo “rate” indica cuál será la velocidad de transmisión de la trama a inyectar. En este estándar los valores que puede tomar este campo son los siguientes: 6,9,12,18,24,36,48 y 54 Mb/s.

```
(RadioTap(present="Rate+Channel",Rate=36,ChannelFlags="OFDM+5GHz")
```

Figura 14: cabecera Radiotap 802.11a

En la figura 14 podemos ver un ejemplo de lo que podría ser una cabecera de este estándar.

4.2.2. 802.11b

Este estándar trabaja en la banda de los 2,4Ghz con una modulación CCK por ello modificaremos estos atributos. A diferencia del estándar anterior los valores de la velocidad de transmisión son diferentes en este estándar son los siguientes: 1,2,5.5 y 11 Mb/s.

En la figura 15 podemos ver una cabecera del estándar “b” con una velocidad de 5.5 Mbps.

```
RadioTap(present="Rate+Channel", Rate=5.5, ChannelFlags="CCK+2GHz"
```

Figura 15: cabecera Radiotap 802.11b

4.2.3. 802.11g

El estándar “g” tiene una configuración muy parecida al estándar “a” con la diferencia más significativa que trabaja en la frecuencia de los 2,4 GHz, en la figura 16 vemos un ejemplo de cabecera.

```
RadioTap(present="Rate+Channel", Rate=54, ChannelFlags="OFDM+2GHz")
```

Figura 16: cabecera Radiotap 802.11g

4.2.4. 802.11n

En el caso de una trama “n”, aparte de modificar la modulación agregaremos el campo MCS (Modulation and Coding Scheme). En la figura 17 podemos ver todos los valores que se le pueden asignar.

Table 1. Data rates (Mbps) of 802.11n/ac (802.11n MCS are in gray) – case of 1 spatial stream (Nss = 1)

MCS	Modulation	Coding Rate	20 MHz		40 MHz		80 MHz		160 MHz	
			GI 0.8µs	GI 0.4µs	GI 0.8µs	GI 0.4µs	GI 0.8µs	GI 0.4µs	GI 0.8µs	GI 0.4µs
0	BPSK	1/2	6.5	7.2	13.5	15	29.3	32.5	58.5	65
1	QPSK	1/2	13	14.4	27	30	58.5	65	117	130
2	QPSK	3/4	19.5	21.7	40.5	45	87.8	97.5	175.5	195
3	16-QAM	1/2	26	28.9	54	60	117	130	234	260
4	16-QAM	3/4	39	43.3	81	90	175.5	195	351	390
5	64-QAM	2/3	52	57.8	108	120	234	260	468	520
6	64-QAM	3/4	58.5	65	121.5	135	263.3	292.5	526.5	585
7	64-QAM	5/6	65	72.2	135	150	292.5	325	585	650
8	256-QAM	3/4	78	86.7	162	180	351	390	702	780
9	256-QAM	5/6	-	-	180	200	390	433.3	780	866.7

Figura 17: Tabla de velocidad de transmisión y ancho de banda

En el atributo “knownMCS” indicaremos los campos del MCS que queremos modificar, los cuáles pueden ser:

- MCS bandwidth
- MCS index
- guard interval
- FEC type
- STBC streams

En el caso del atributo “MCS bandwidth” podemos asignar los siguientes valores:

- 0 = 20 Mhz
- 1 = 40 Mhz
- 2 = 40 Mhz-
- 3 = 40 Mhz+

Para el atributo “guard_interval” se permiten 2 valores:

- 1 = 400 ns (corto)
- 0 = 800 ns (largo)

El campo “STBC_streams” indica el número de secuencias espaciales [19].

```
RadioTap(present="Channel+MCS",ChannelFlags="2GHz+Dynamic_CCK_OFDM",
         knownMCS='MCS_bandwidth+MCS_index+guard_interval+STBC_streams',
         MCS_bandwidth=0, MCS_index=2, guard_interval=1, STBC_streams=0)
```

Figura 18: cabecera Radiotap 802.11n

En la figura 18 podemos ver un ejemplo de cabecera 802.11n, vemos que esta es más compleja en cuanto a que se tiene que agregar más campos y es la primera que trabaja en la frecuencia de 2 y 5 GHz.

4.2.5.802.11ac

Para inyectar tramas de este estándar es necesario que haya un intercambio de tramas concreto [8]. Una vez realizado el intercambio de tramas se procede a la construcción de la cabecera Radiotap que es muy parecida a la del estándar “n” con el añadido del campo “A-MPDU” por lo que agregamos dentro de “present” lo que nos permitirá modificar el atributo “A_MPDU_ref” y “A_MPDU_flags”.

El campo “A-MPDU_Ref” lo genera el dispositivo de captura y es el mismo en cada subtrama de una A-MPDU [8], en el caso de los “A_MPDU_flags” que nos permite configurar Radiotap son:

- Report0Subfram = El controlador informa subtramas de longitud 0
- Is0Subframe = La subtrama es de longitud 0
- Known Last Subframe = Se conoce la última subtrama
- LastSubframe = Es la última subtrama
- CRCerror = Delimitador CRC

```
RadioTap(present="Channel+MCS+A_MPDU", ChannelFlags="2GHz+Dynamic_CCK_OFDM",
         knownMCS='MCS_bandwidth+MCS_index+guard_interval+STBC_streams', MCS_bandwidth=0
         , MCS_index=2, guard_interval=1, STBC_streams=0, A_MPDU_ref=1298091,
         A_MPDU_flags='LastSubframe')
```

Figura 19: cabecera Radiotap 802.11ac

En la figura 19 vemos lo que podría ser una la cabecera Radiotap de una trama del estándar "ac".

4.3. Implementación del teorema chino de los restos

Hemos visto que una parte importante a la hora de utilizar el teorema chino de los restos es la generación de las claves, las cuales tienen que ser mayor que el paquete que se quiere cifrar. Para ello en el Anexo 2.1 se explicará el código de cómo se han generado, al finalizar ese proceso tendremos un vector de claves XS, otro vector del mismo tamaño con claves PS y un clave Hdr. Estas claves serán colocadas en el transmisor el cual las utilizará para cifrar todos los paquetes e inyectar la trama cifrada y en los receptores se tendrán sólo las claves de los paquetes que estén destinados hacia ellos.

El transmisor (Access Point) utiliza 3 claves para cifrar un paquete para un dispositivo. Dos claves son privadas (XS, PS) ya que solo las tienen los que van a recibir el paquete en específico. La otra clave (Hdr) es compartida por lo que todos los dispositivos que esperan recibir algún paquete de la trama agregada cifrada tendrán acceso a ella. Esto significa que si se quieren enviar N paquetes dentro de una misma trama agregada se necesitarán N claves privadas y una pública, independientemente de que haya varios paquetes destinados al mismo dispositivo, ya que el modelo no permite reutilizar claves para cifrar más de un paquete que vaya a ser enviado en la misma trama. Pero si se quiere enviar otra trama agregada se puede utilizar claves repetidas siempre respetando lo mencionado anteriormente.

Él código que se utilizó para implementar el cifrado del teorema chino de los restos se encuentra en el Anexo 2.2, el resultado de aplicar el teorema es la compactación de los paquetes que se han cifrado de manera individual en una trama de mayor longitud lista para la transmisión.

Para el descifrado, los receptores (STA) deben tener las claves privadas XS y PS de los paquetes deseados, además de la clave pública Hdr la cual no tiene ninguna relación con las otras dos por lo que es casi imposible llegar a las claves privadas a partir de ella. Para este trabajo asumimos que el intercambio de claves ya se ha llevado a cabo en el protocolo de autenticación en el que la STA y AP intercambian todas las claves que van a utilizar en el proceso de cifrado de la comunicación [20]. Por lo que cuando le llegue el paquete inyectado que enviar el AP sólo tiene que aplicar la función de descifrado juntos con sus claves para poder ver todos los paquetes que se le han enviado. El código de descifrado se encuentra en el Anexo 2.3.

En base a lo comentado anteriormente mostraremos en el transmisor los paquetes antes de cifrarlos de tal manera que será posible ver las direcciones MAC de los dispositivos, luego las cifraremos y mostraremos el resultado de la trama antes de ser inyectada. Para el caso de recepción mostraremos el paquete que hemos capturado y comprobaremos que es el mismo que ha enviado el transmisor. Luego aplicaremos el descifrado para ver en claro el paquete destinado a ese dispositivo.

5. Implementación del escenario

En el primer apartado explicaremos de qué manera se pueden descargar y ejecutar el código subido a GitHub, donde se encuentra implementado todo lo que se había propuesto en el apartado 2.4. En los siguientes apartados se hará uso de ese código para ir explicando todos los resultados obtenidos.

5.1. Configuración y ejecución de repositorios en GitHub

En el anexo 5.1 se explica cómo está estructurada el repositorio y los comandos para poder descargar los ficheros[21].

Como configuración inicial las antenas que vayan a interactuar en el proceso de transmisión deben estar configuradas como indica el apartado 3. Por lo que partiremos de la figura 12 para ir explicando la manera en la que se tiene que ir compilando el código para cada caso.

5.1.1. Código de inyección con los estándares

El código implementa una serie de cabeceras construidas según los diferentes estándares de tal manera que permite al usuario a elegir con qué estándar transmitir y a qué velocidad. Para la comprobación de esta se brinda otro fichero que permite capturar la trama y mostrarla.

Se procede a ejecutar el siguiente comando dentro del terminal del Debian10_Tx:

- python3 inject_tx.py n_estandar vel

Valores de “n_estandar”

1 = cabecera 802.11a
2 = cabecera 802.11b
3 = cabecera 802.11g
4 = cabecera 802.11n

Campo “vel” indica la velocidad de transmisión en Mbps con la que se quiere transmitir, siempre respetando las características del estándar.

Para poder capturar el paquete transmitido y ver su estructura se utiliza el siguiente comando en el dispositivo Debian10_Rx.

- Python3 inject_rx.py

El cual utiliza la herramienta Sniff para mostrar el paquete que ha capturado.

5.1.2. Código de trama agregada normal y segura

Para esta parte se ha implementado los dos modelos de tramas agregadas, para implementar el modelo de agregación normal primero se tiene ejecutar el receptor Debian10_RX después el transmisor Debian10_tx ya que estas tramas envían entre ellas una serie de tramas de control antes de enviar la de datos.

En el caso de las tramas agregadas seguras se implementa el código para enviar 5 paquetes desde un transmisor (Debian10_Tx), en el receptor Debian10_Rx se tendrán las claves suficientes para descifrar 3 paquetes y en el Debian10_Rx2 las claves para descifrar 1 paquete.

Se irán mostrando en forma hexadecimal los datos enviados, cifrados y descifrados para una mejor comprobación a la hora de ver si todo el proceso es correcto.

En AP (Debian10_tx):

- Python3 Tx_process.py rate formaCifrado

En STA (Debian10_Rx):

- Python3 Rx_process.py rate formaCifrado

En STA2 (Debian10_Rx2)

- Python3 Rx_process_Ant2.py rate formaCifrado

Campo “rate” es la velocidad de transmisión en Mbps.
Valores de forma de cifrado:

1= Cifrado normal A-MPDU
2= Cifrado con el teorema chino de los restos

5.2. Inyección de tramas con los diferentes estándares

Para inyectar tramas utilizaremos la función de Scapy llamada “sendp” el cual tiene como parámetro de entrada el paquete que quieres transmitir con el formato adecuado y el nombre del interfaz, en la figura 20 vemos como es todo el proceso.

```
IFACE = 'wlx00c0caa4737b' 1          #Interfaz de la targeta de red del transmisor
AP_MAC = ' 00:c0:ca:a4:73:7b'        #Direccion mac del transmisor
AP_MAC_2= '00:c0:ca:a4:73:7c'        #Direccion mac del receptor()
packet = []                                                          2
packet.append(RadioTap(present="Rate+Channel",Rate=12,ChannelFlags="OFDM+5GHz") /
                       Dot11(type=2, subtype=8, addr1=AP_MAC_2, addr2=AP_MAC,addr3=AP_MAC))
sendp(packet, iface=IFACE, verbose=False)  3
```

Figura 20: Ejemplo de inyección de trama

En la figura 20.1 vemos el interfaz por el cual inyectaremos el tráfico, en la 20.2 el paquete que vamos a inyectar y finalmente en la figura 20.3 el comando donde se indica por donde y que enviar al medio inalámbrico.

5.2.1. Métodos de comprobar el tráfico inyectado

Para inyectar las tramas necesitamos tener la configuración explicada en el apartado 3. Hemos visto cómo construir la cabecera Radiotap, pero para transmitir información hace falta agregar otros datos como las cabeceras Radiotap y Mac. Por lo que en el Anexo 3.1 se explicará cómo se ha construido la trama.

A Continuación, veremos las 2 maneras que tenemos para confirmar que el tráfico se ha enviado correctamente, dependiendo del caso utilizaremos uno u otro para visualizar los resultados.

5.2.1.1. Mediante la librería de Scapy Sniff

Esta manera de ver las tramas inyectadas es la más fiable y consiste en configurar el STA (Debian10_Rx) para que esté escuchando siempre. Para ello utilizaremos la función Sniff de la librería Scapy que tiene como parámetro de entrada el interfaz en el que va a estar escuchando, una función en la que se va a recibir el paquete y el valor del tiempo de espera. El código para implementar el receptor se encuentra en el Anexo 4.1. Esta opción será la utilizada cuando queramos comprobar si las tramas agregadas seguras que hemos recibido son las correctas ya que nos permite obtener todo el paquete transmitido por el AP para luego poder descifrar y ver si los datos son los correctos.

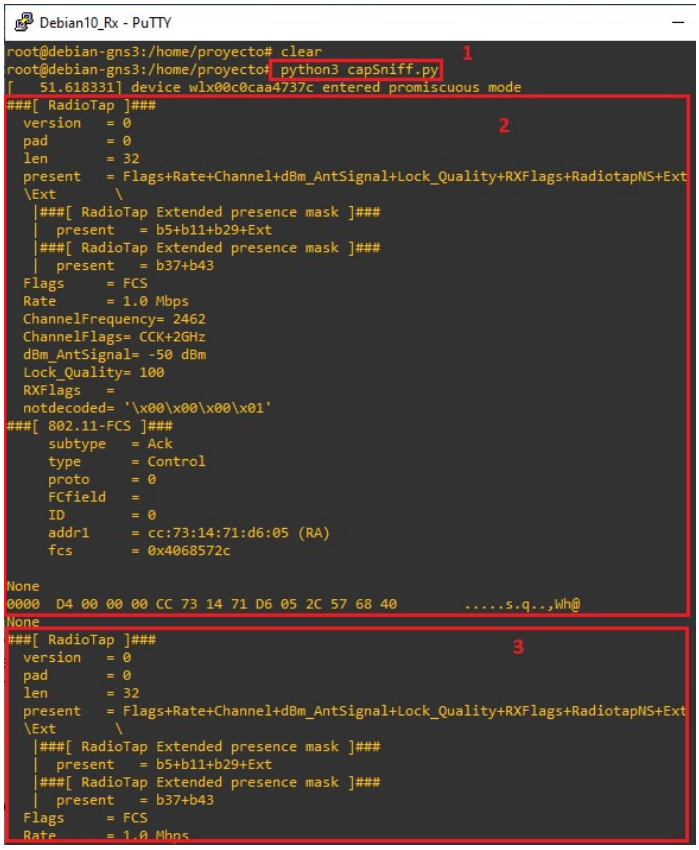

Figura 21: Captura mediante Sniff

En la figura 21.1 vemos como ejecutando el archivo de python3 podemos ver en las figuras 21.2 y 21.3 algunos de los paquetes capturados por esa interfaz.

5.2.1.2. Mediante la herramienta Tcpdump

En esta opción configuraremos la antena (Debian 10 Cap) para que pueda capturar todo el tráfico mediante su interfaz inalámbrica puesta en modo monitor. Para ello utilizaremos la herramienta Tcpdump, la cual creará una captura con todo el tráfico que ha recopilado, luego mediante Wireshark podremos comprobar si la estructura de la trama es correcta. Cabe resaltar que este modelo de captura es menos fiable que el mencionado anteriormente ya que en las diferentes pruebas realizadas en este trabajo ha habido tramas que se han visto mediante la herramienta Sniff mientras que Tcpdump no ha podido capturarlas.

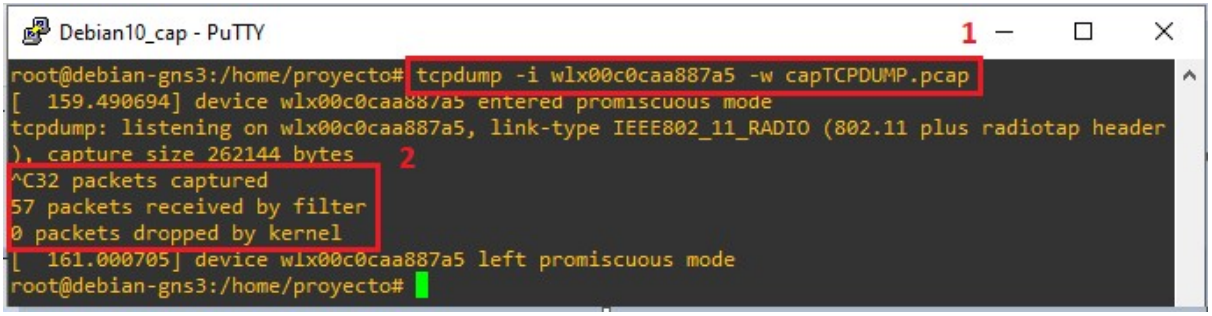

Figura 22: Captura mediante Tcpdump

En la figura 21.1 vemos el comando para capturar el tráfico mediante Tcpdump el cual se guardará en el ficho capTCPDUMP.pcap que contiene 32 paquetes tal como vemos en la figura 22.2.

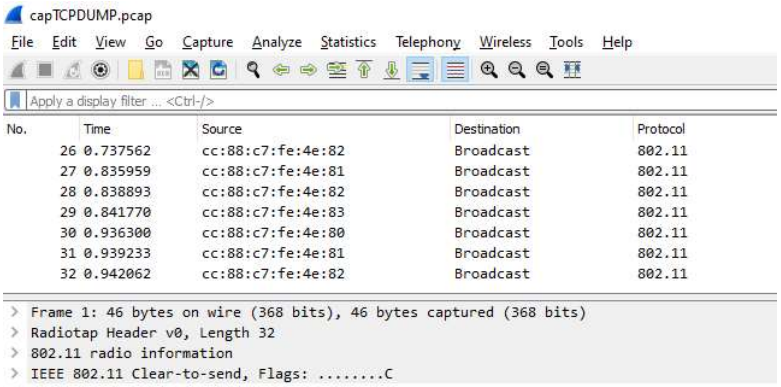

Figura 23: Vista del fichero con Wireshark

En la figura 23 vemos los 32 paquetes capturados mediante la ayuda de Wireshark.

5.2.2. Verificación de tramas

No podemos mostrar el correcto funcionamiento de la inyección de tráfico para todos los tipos de tramas, pero sí lo haremos para las más significativas. Para verificar las tramas sólo nos centraremos en la cabecera Radiotap vista en la trama recibida, ya que en ésta se encuentra los cambios que hemos realizado para los diferentes estándares. Mostraremos la trama inyectada y la trama capturada mediante Tcpdump para poder confirmar que la hemos transmitido correctamente.

A continuación, mostraremos el resultado de inyectar todas las tramas que hemos explicado en el apartado 4.2.

5.2.2.1. 802.11a

En la figura 24 vemos la trama que hemos inyectado con la modificación de la cabecera Radiotap. La cual tiene las características de modulación y frecuencia del estándar “a” con la velocidad de transmisión modificada de 36 Mbps.

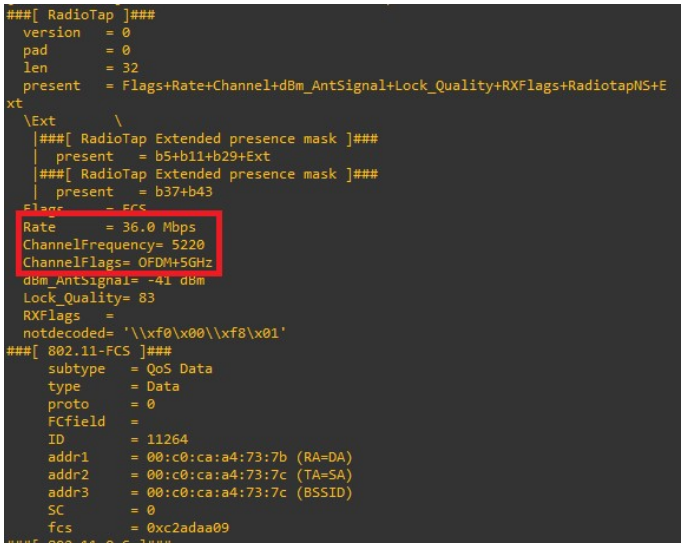

Figura 24: Trama inyectada 802.11a

En la figura 25 tenemos la captura en Wireshark que hemos realizado de la trama mediante Tcpdump. Vemos que la estructura de la cabecera Radiotap se ha construido correctamente ya que Wireshark reconoce que es una trama 802.11a transmitida con la velocidad de 36 Mbps.

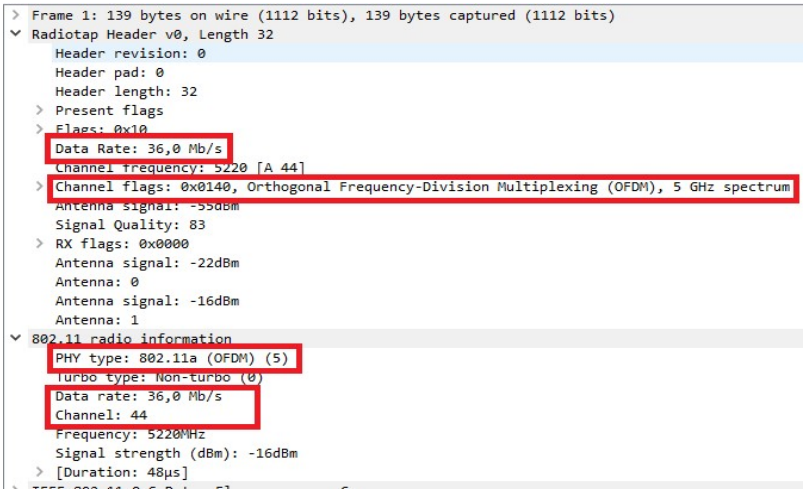

Figura 25: Captura de Wireshark de trama 802.11a

5.2.2.2. 802.11b

En la figura 26 vemos la trama inyectada con los atributos modificados según el estándar “b”, en esta ocasión hemos asignado una velocidad de transmisión de 5.5 Mbps. Estamos utilizando una frecuencia más baja ya que este estándar trabaja en la banda de los 2.4 GHz.

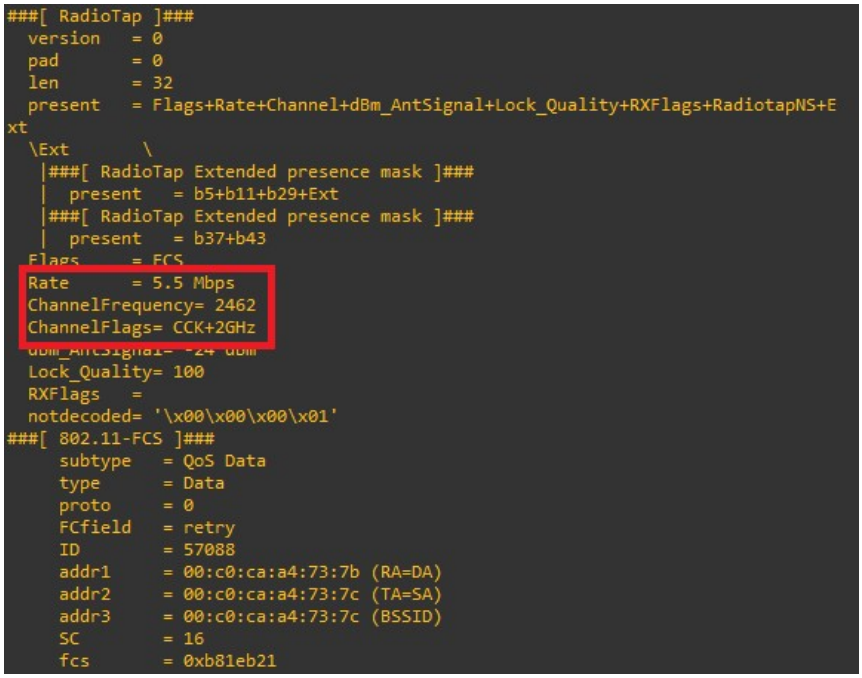

Figura 26: Trama inyectada 802.11b

En la figura 27 podemos ver que las modificaciones realizadas en la cabecera Radiotap se han realizado correctamente ya que hemos podido capturar la trama a la velocidad indicada con las características del estándar "b".

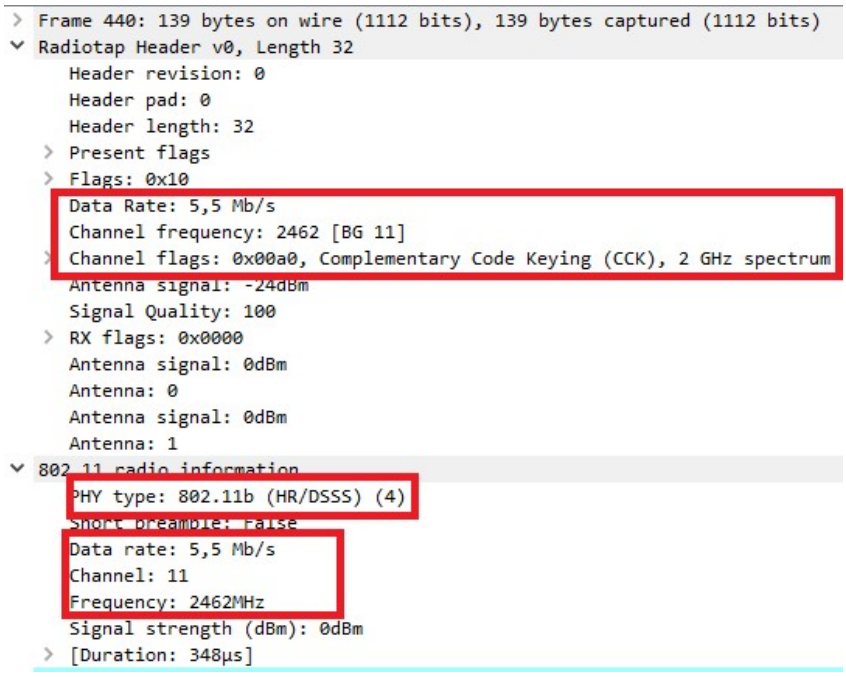

Figura 27: Captura de Wireshark de trama 802.11b

5.2.2.3. 802.11g

En la figura 28 inyectamos una trama en la banda de los 2 GHz transmitiendo a una velocidad de 54 Mb/s con la modulación necesaria del estándar “g”.

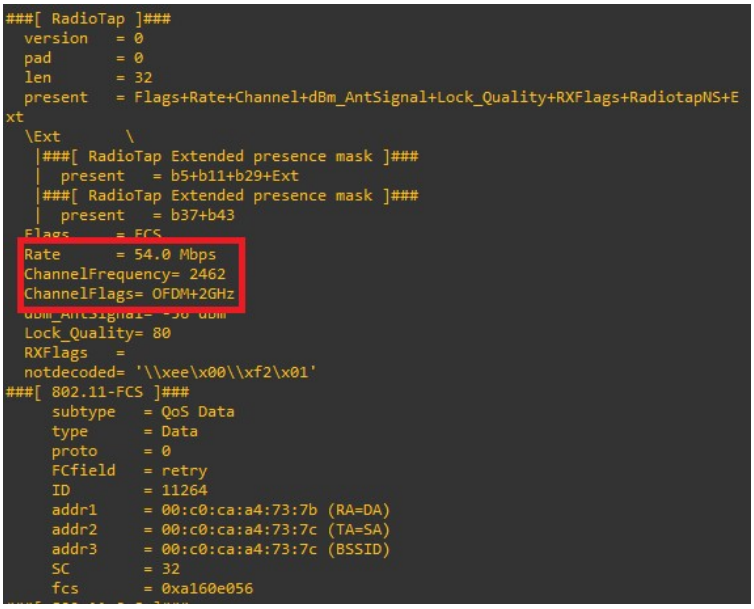

Figura 28: Trama inyectada 802.11g

En la figura 29 podemos ver que hemos capturado la trama con los cambios realizados correctamente y con la estructura del estándar “g”.

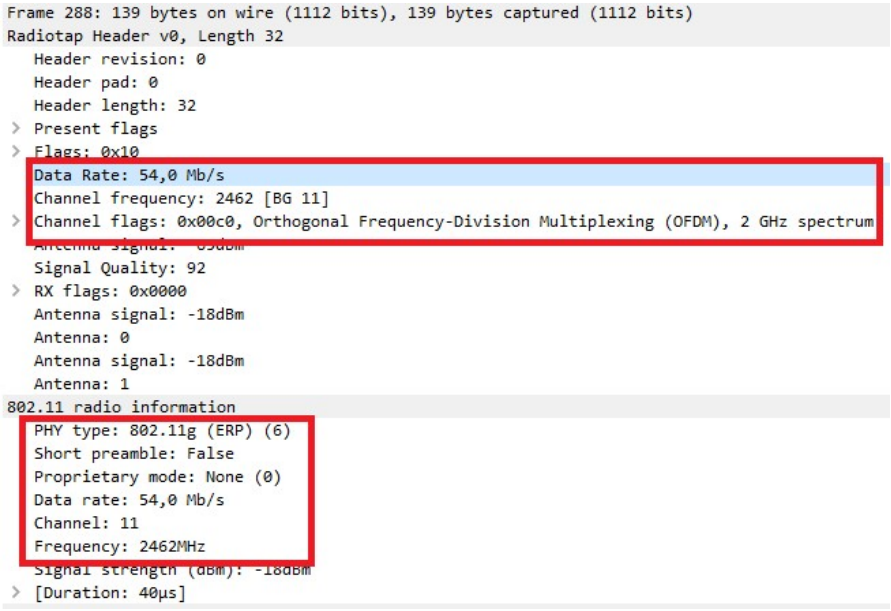

Figura 29: Captura de Wireshark de trama 802.11g

5.2.2.4. 802.11n

En la figura 30 vemos la trama que estamos inyectando y señalamos las partes de la cabecera que hemos modificado. Podemos ver que el campo MCS se ha construido correctamente.

```
###[ RadioTap ]###
  version   = 0
  pad       = 0
  len       = 35
  present   = Flags+Channel+dBm_AntSignal+Lock_Quality+RXFlags+MCS+RadiotapNS+Ex
t
  \Ext      \
   |###[ RadioTap Extended presence mask ]###
   |  present   = b5+b11+b29+Ext
   |###[ RadioTap Extended presence mask ]###
   |  present   = b37+b43
  Flags     = FCS+ShortGI
  ChannelFrequency= 2462
  ChannelFlags= 2GHz+Dynamic_CCK_OFDM
  dBm_AntSignal= -42 dBm
  Lock_Quality= 48
  RXFlags   =
  knownMCS  = MCS_bandwidth+MCS_index+guard_interval+STBC_streams
  Nss= 1 FEC= 0
  STBC_streams= 0
  FEC_type  = BCC
  HT_format = mixed
  guard_interval= Short_GI
  MCS_bandwidth= 20MHz
  MCS_index = 2
  notdecoded= '\\xf2\x00\\xf2\x01'
###[ 802.11-FCS ]###
     subtype   = QoS Data
     type      = Data
     proto     = 0
     FCfield   =
     ID        = 12288
     addr1     = 00:c0:ca:a4:73:7b (RA=DA)
     addr2     = 00:c0:ca:a4:73:7c (TA=SA)
     addr3     = 00:c0:ca:a4:73:7c (BSSID)
     SC        = 48
     fcs       = 0x70bc6cee
```

Figura 30: Trama inyectada 802.11n

En la figura 31 podemos confirmar que la cabecera Radiotap se ha construido correctamente ya que hemos recibido la trama con la estructura y estándar que queríamos.

```
Frame 131: 142 bytes on wire (1136 bits), 142 bytes captured (1136 bits)
Radiotap Header v0, Length 35
   Header revision: 0
   Header pad: 0
   Header length: 35
 > Present flags
 > Flags: 0x90
   Channel frequency: 2462 [BG 11]
 > Channel flags: 0x0480, 2 GHz spectrum, Dynamic CCK-OFDM
   Antenna signal: -40dBm
   Signal Quality: 54
 > RX flags: 0x0000
 v MCS information
   > Known MCS information: 0x27, Bandwidth, MCS index, Guard interval, STBC streams
       .... ..00 = Bandwidth: 20 MHz (0)
       .... .1.. = Guard interval: short (1)
       .00. .... = STBC streams: 0
     MCS index: 2
   [Data Rate: 21,7 Mb/s]
   Antenna signal: -14dBm
   Antenna: 0
   Antenna signal: -14dBm
   Antenna: 1
802.11 radio information
   PHY type: 802.11n (HT) (7)
   MCS index: 2
   Bandwidth: 20 MHz (0)
   Short GI: True
   Number of STBC streams: 0
   Data rate: 21,7 Mb/s
   Channel: 11
   Frequency: 2462MHz
   Signal strength (dBm): -14dBm
 > [Duration: 79us]
```

Figura 31: captura de Wireshark de trama 802.11n

5.3. Inyección de tramas seguras

Para las siguientes pruebas, como ya se había mencionado anteriormente, partimos sabiendo que las claves ya se han compartido de tal manera que los dispositivos que recibirán paquetes tendrán solamente las claves de los paquetes destinados hacia él.

En el transmisor se ejecuta la función cifrarCRT que tiene como parámetros de entrada las claves que se utilizarán en el proceso de cifrado. Esta función crea MSDU de determinado tamaño para luego convertirlos en MPDU con la función MPDU_gen, estos paquetes son cifrados con el teorema de los restos chinos en la función MPDUs_enc el cual se encarga de devolver la trama cifrada.

Como hemos visto anteriormente una parte fundamental a la hora de construir una trama Wi-Fi es la de cabecera MAC en la cual van las direcciones de origen y destino del paquete, en este caso asignaremos la dirección destino STA (Debian10_Rx) para poder inyectar y confirmar el modelo de forma más directa. Si bien la mejor opción sería que cada dispositivo sea agregado a un grupo *multicast* al momento de conectarse al AP (Debian10_tx) por lo que este enviará la trama agregada a la dirección MAC del grupo, así no se podría saber la dirección MAC de ningún dispositivo.

```
packet = []
qoscontrol = b'\x00\x00'                    1
mpduCi_Hex, mpduCi = cifrarCRT(ps,xs,Hdr)                    2
packet.append(RadioTap() / Dot11(type=2, subtype=8, addr1=pkt.addr2, addr2=AP_MAC, addr3=AP_MAC)
            3/ qoscontrol / mpduCi_Hex)  4
print("Paquete cifrado enviado")
print(hexdump(mpduCi_Hex))    5
sendp(packet, iface=IFACE, verbose=False)
```

Figura 32: Código de creación y envío de trama agregada segura

En la figura 32.1 podemos ver la función comentada anteriormente que devuelve la trama cifrada en formato binario y numérico. Las figuras 32.2 y 32.3 son partes necesarias para inyectar una trama. La primera es la cabecera Radiotap por defecto, lo que indica que se transmitirá utilizando el estándar 802.11b a velocidad de 6 Mbps acompañada del tipo de trama[22] que en este caso es el de datos y la cabecera MAC. La segunda indica el campo QoScontrol. La figura 32.4 es la trama agregada cifrada en formato binario para hacer posible su transmisión. Finalmente vemos en la figura 32.5 cómo se inyecta todo el paquete al medio inalámbrico.

El receptor captura el tráfico mediante la función Sniff, a este paquete se le quita toda la cabecera Radiotap y MAC, además del campo QoS de tal manera que nos quedamos solamente con los datos cifrados que veíamos en la figura 31.4. Para descifrar la trama utilizamos la función MPDUs_dec el cual tiene como parámetro de entrada las claves que se le han asignado de los paquetes que espera recibir y la trama cifrada. El resultado es la obtención de los paquetes descifrados destinados a ese dispositivo.

Las modificaciones más relevantes en cada caso serán la creación de los paquetes con una determinada longitud y el reparto de claves tanto en el transmisor como en los diferentes receptores que estén esperando algún paquete, dependiendo a esto se irán mostrando los paquetes que serán cifrados enviando y capturados por el transmisor y receptor.

5.3.1. Casuísticas

Para visualizar mejor los datos cifrados y descifrados mostraremos las capturas en el receptor con la función de Scapy Sniff. Veremos los paquetes en formato hexadecimal tanto al inyectarla como al capturarla debido a que nos permite ver mejor las diferencias a la hora de cifrar y descifrar un paquete. La propuesta [3] es muy libre a la hora de seleccionar un estándar en específico con el cual debe ser transmitido por lo que para las siguientes pruebas se ha utilizado el 802.11b.

Una parte muy importante de todo el proceso es la creación de claves, las cuales se han generado mediante el código que se encuentra en el Anexo 2.1. Para estos casos se han creado 5 pares de claves privada y pública con una longitud de 128 bytes. Dependiendo del número de paquetes que se deseen cifrar se ha ido colocando las claves suficientes para cifrar los paquetes en el transmisor e iremos poniendo en los distintos receptores algunas claves para que puedan descifrar los paquetes específicos, así pondremos a prueba nuestro modelo de trama agregada segura.

- **AP (Debian Tx) envía agregados 2 paquetes de la misma longitud - STA (Debian Rx) solo puede descifrar 1 paquete**

Para esta prueba generamos 2 paquetes de la misma longitud con una estructura MPDU normal. En la figura 33.1 y 33.2 vemos los dos paquetes sin cifrar con direcciones MAC destino 15:ff:54:74:b3:78 y 00:c0:ca:a4:73:7c (Debian10_rx) respectivamente, ambos con dirección MAC origen 00:c0:ca:a4:73:7b (Debian10_Tx). Si aplicamos la función de cifrado del teorema chino de los restos explicado en el Anexo 2.2 sobre los dos paquetes obtenemos un solo paquete cifrado tal como vemos en la figura 33.3 donde todas las cabeceras MAC de los diferentes dispositivos, mencionados anteriormente, ya no son visibles. Esta trama es la que inyectaremos al medio inalámbrico; si bien todos los dispositivos que estén cerca podrán ver todo el paquete sólo los que tengan las claves de los paquetes que se le han asignado podrán descifrar los paquetes.

Figura 33: Vista de los 2 paquetes sin cifrar y trama cifrada que se inyectara con Debian10_tx

En la figura 34.1 vemos que hemos recibido todo el paquete cifrado correctamente debido a que no se puede apreciar ninguna cabecera MAC de ningún dispositivo ni el número de paquetes que contiene. Aplicando la función de descifrado, explicado en el Anexo 2.3 y las claves que tiene asignadas el dispositivo, se obtiene sólo el paquete descifrado, tal como se puede ver en la figura 34.2 el cual es idéntico al enviado en la figura 33.2.

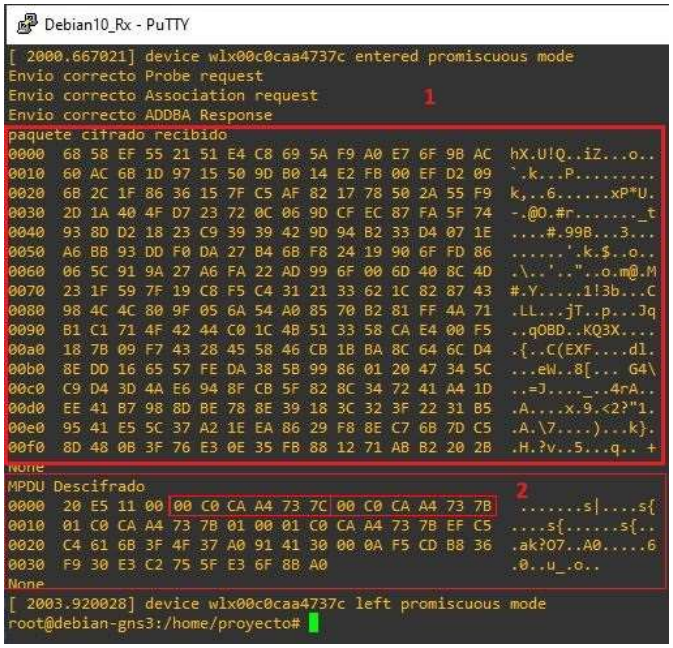

Figura 34: Vista de datos recibidos y el paquete descifrado en Debian10_Rx

- **AP (Debian Tx) envía agregados 3 paquetes de diferente longitud - STA (Debian Rx) tiene destinado el paquete más grande y STA2 (Debian Rx2) el más pequeño**

Para esta prueba generamos 3 paquetes de diferente longitud. En la figura 35.1 vemos el primer paquete que tiene una MAC destino de 00:c0:ca:a8:87:a5 (Debian10 Rx2), en la 35.2 el segundo paquete que va dirigido a otro dispositivo con dirección c2:ff:52:2f:51:db y en la 35.3 el último paquete que va dirigido a 00:c0:ca:a4:73:7c (Debian10 Rx). Todos ellos con la dirección MAC origen 00:c0:ca:a4:73:7b (Debian10 Tx) que es el terminal desde dónde se enviará la trama cifrada. Aplicamos el modelo de trama segura en los 3 paquetes obteniendo el único paquete de la figura 35.4 donde no se aprecia ninguna dirección MAC de ningún dispositivo. Esta trama es la inyectada por el medio inalámbrico y las diferentes estaciones utilizarán las claves de los paquetes que se le han asignado para obtener el paquete descifrado.

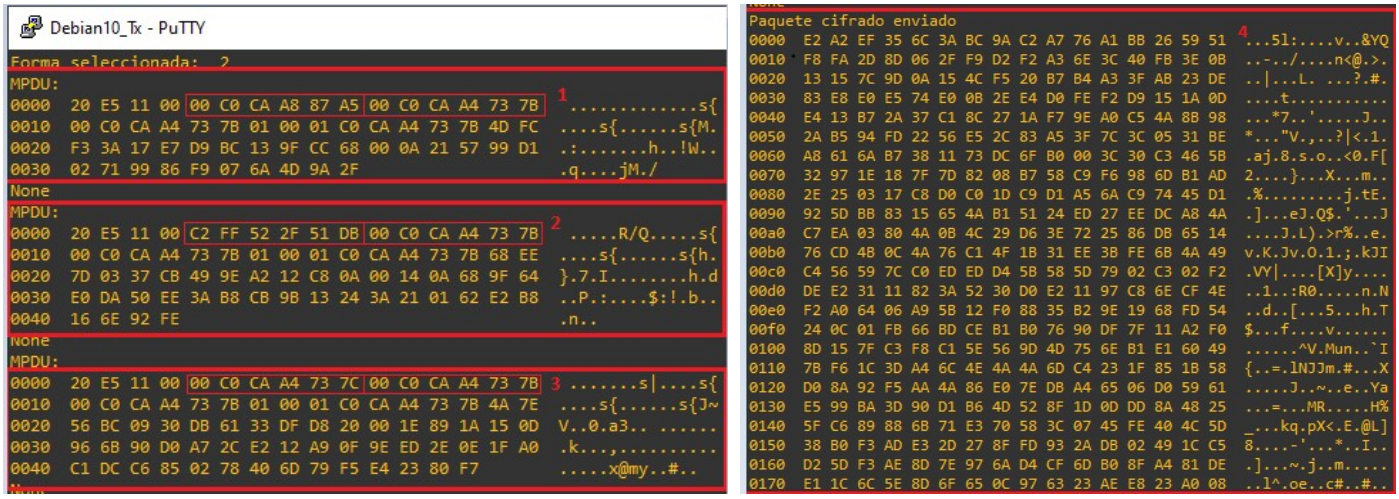

Figura 35: Vista los 3 paquetes sin cifrar y trama cifrada que se inyectara con Debian10_tx

En la figura 36.1 vemos que hemos recibido el paquete cifrado correctamente en el equipo Debian10 Rx, de tal manera que este dispositivo no sabe cuántos paquetes hay dentro de este ni las direcciones de los demás paquetes, que es lo que se busca con el modelo de tramas agregadas seguras. Si aplicamos el descifrado con la clave del paquete que le corresponde obtenemos solo ese paquete tal como vemos en la figura 36.2 el cual es idéntico al de la figura 35.3.

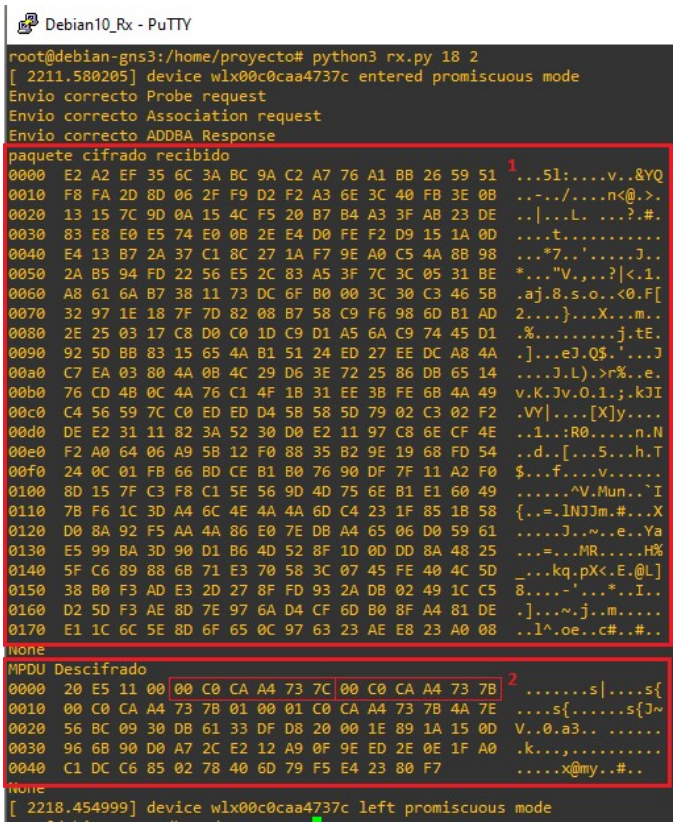

Figura 36: Vista de datos recibidos y paquete descifrado en Debian10_Rx

En la figura 37.1 capturamos el paquete cifrado con el terminal Debian10 Rx2 (Debian10 Cap) y aplicando la clave del paquete que le corresponde obtenemos el paquete de la figura 37.2 que es idéntico al de la 35.1.

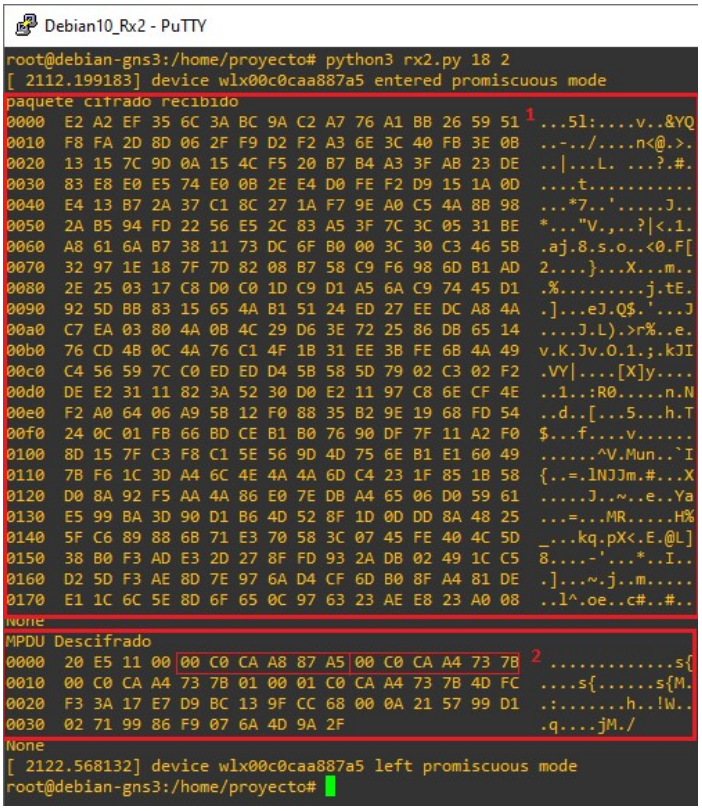

Figura 37: Vista de datos recibidos y paquete descifrado en Debian10_Rx2

- **AP (Debian Tx) envía 5 paquetes de diferente longitud - STA (Debian Rx) tiene destinados 3 paquetes y STA2 (Debian Rx2) 1 paquete**

En este caso generamos 5 paquetes de diferente longitud de las cuales el primer paquete que vemos en la figura 38.1 será para el terminal Debian10 Rx2, otro al equipo con dirección MAC 4f:10:ed:7c:51:38 y los otros 3 paquetes para el Debian10 Rx, tal como se representan en las de las figuras 38.3,38.4 y 38.5.

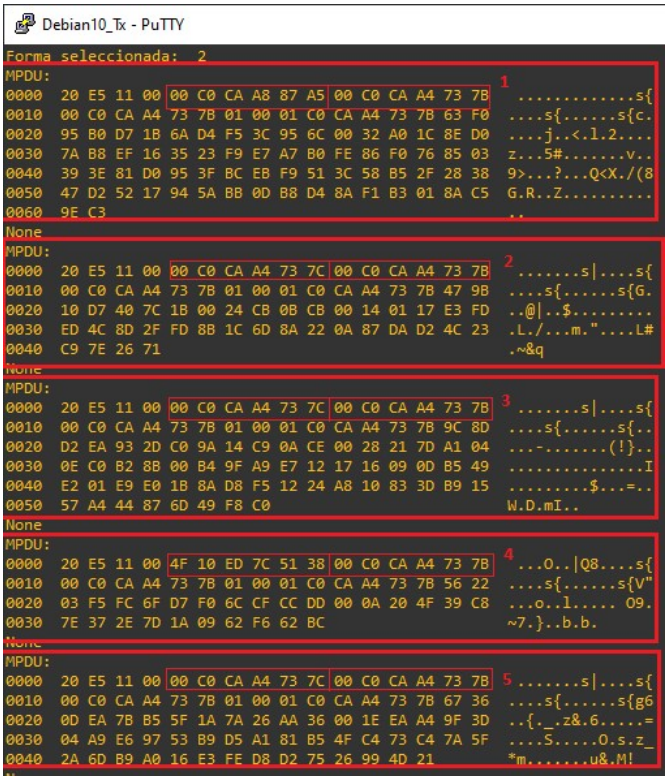

Figura 38: Vista de los 5 paquetes sin cifrar

En la figura 39.6 vemos el paquete cifrado resultado de aplicar el modelo de trama segura que será enviado por el medio inalámbrico.

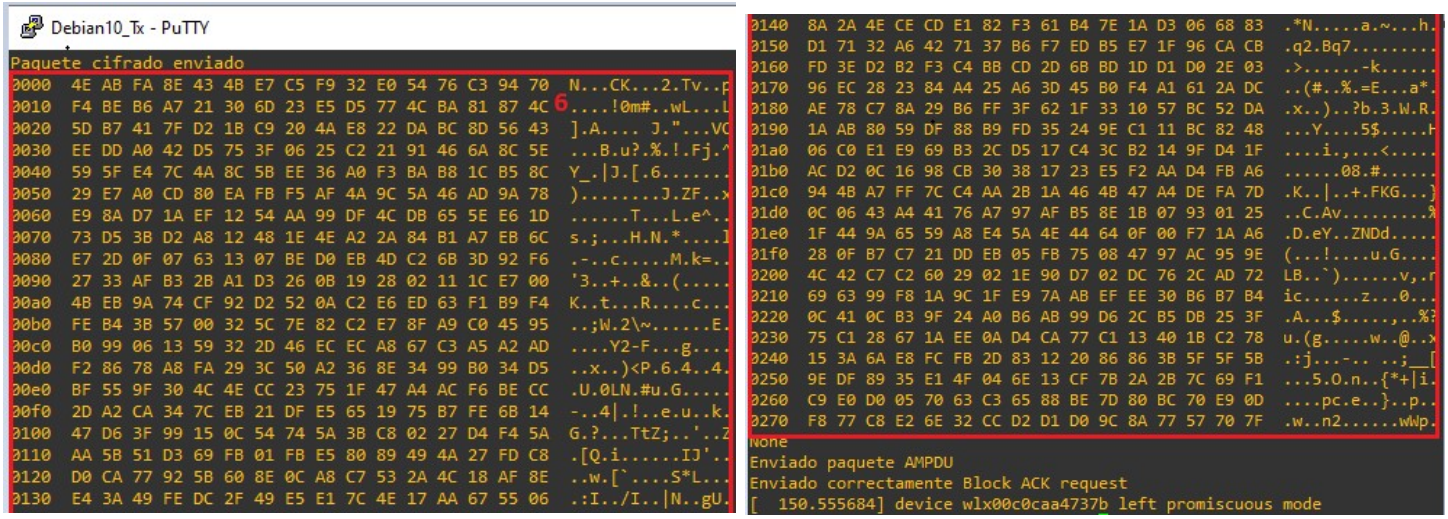

Figura 39: Vista de la trama cifrada que se inyectara con Debian10_tx

En la figura 40.1 vemos el paquete que ha capturado el terminal Debian10 Rx.

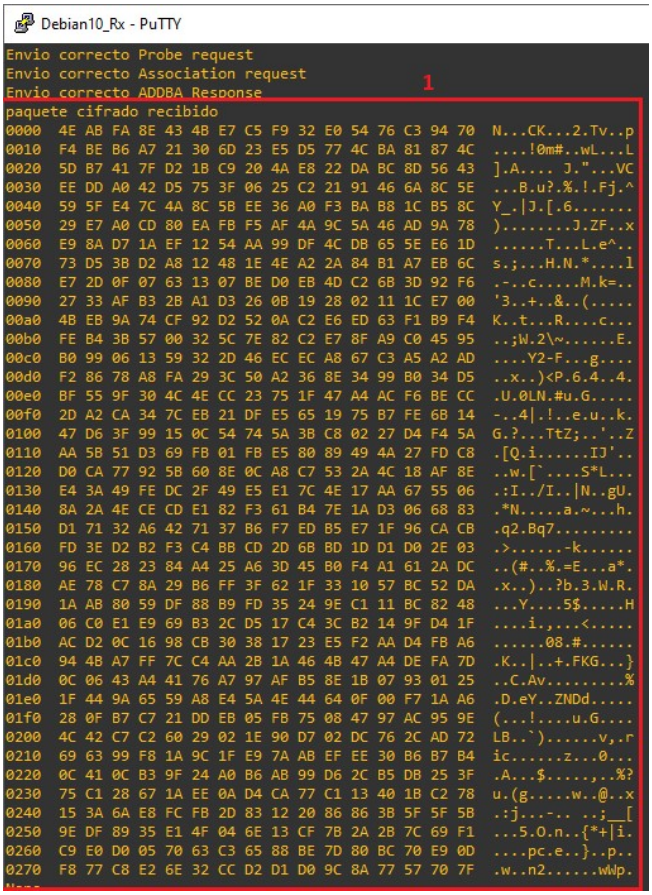

Figura 40: Vista de datos recibidos Debian10_Rx para el caso de 5 paquetes

Luego de aplicar el descifrado del paquete capturado con las claves que se le han enviado anteriormente obtenemos los paquetes de las figuras 41.1, 41.2 y 41.3.

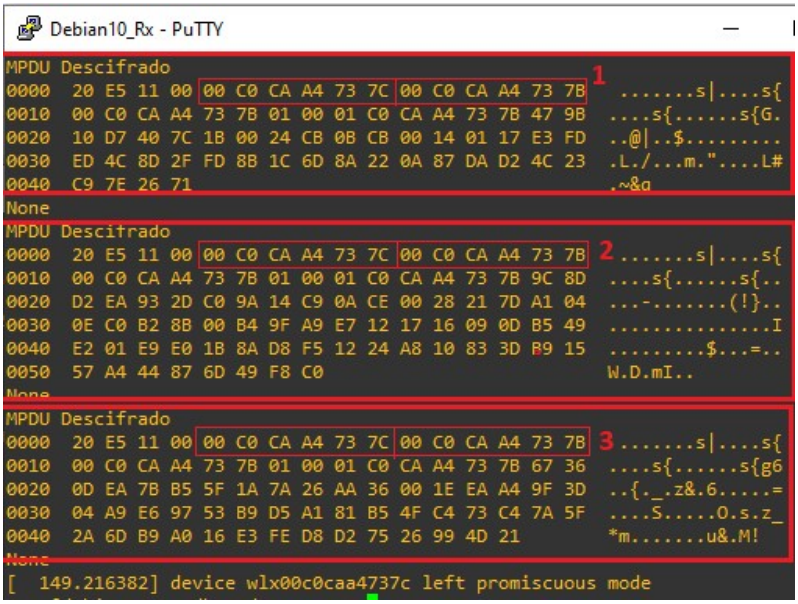

Figura 41: Vista de los 3 paquetes descifrado en Debian10_Rx

En la figura 42.1 vemos la trama que ha capturado el terminal Debian10 Rx2, la cual es misma que vemos en la 40.1 y 39.6.

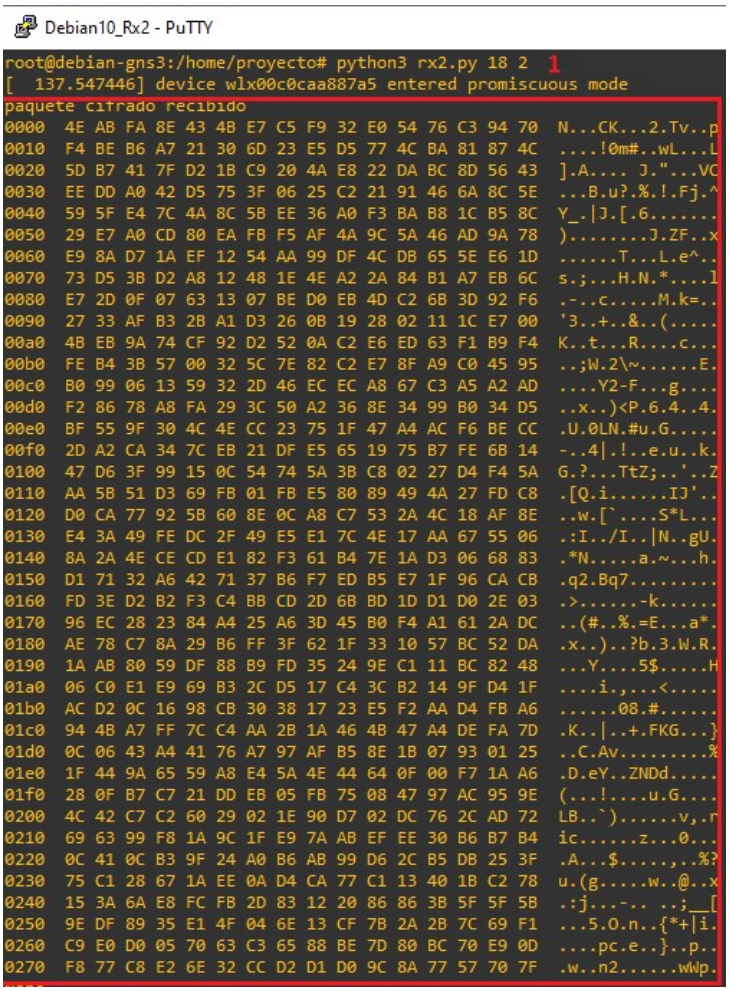

Figura 42: Vista de datos recibidos en Debian10_Rx2 para el caso de 5 paquetes

Luego de aplicar el descifrado del paquete 42.1 obtenemos la trama que corresponde al dispositivo Debian10_Rx2 tal como vemos en la imagen 43.1.

Figura 43: Vista del paquete descifrado en Debian10_Rx2

6. Conclusiones y líneas futuras

6.1. Conclusiones

Hemos visto que utilizando GNS3 como herramienta de gestión y virtualización se pueden controlar equipos reales de manera remota. Si bien la configuración pueda parecer compleja, como se explica en el apartado 3, una vez realizada tenemos control remoto total del equipo, lo cual nos ha brindado una mayor facilidad a la hora de poder crear escenarios controlados de laboratorio donde inyectar y capturar tráfico Wi-Fi.

Se ha hecho un estudio de los diferentes estándares que propone la IEEE para la transmisión inalámbrica de datos donde hemos explicado las características más importantes de cada uno de ellos. Luego mediante el uso de la herramienta Radiotap, que está desarrollada en Python3, hemos podido modificar la cabecera para crear tramas que envíen información sobre diferentes estándares. Además, podemos modificar algunos valores relevantes como la velocidad de transmisión, tal como lo explicamos en el apartado 4.2.

Por otro lado, hemos podido capturar el tráfico Wi-Fi real que generan los diferentes dispositivos integrantes de nuestro escenario virtual mediante el uso de herramientas como Tcpdump y la librería de Scapy Sniff. Esto nos ha permitido ver si el tráfico que hemos inyectado tiene la estructura correcta o si ha habido algún fallo, lo que nos permite disponer de una herramienta de análisis de tráfico real muy potente desde una ubicación remota. Estas herramientas se pueden utilizar para un monitoreo de redes inalámbricas más fiable mediante la librería Sniff que, además, permite analizar las tramas.

Se ha implementado el modelo de cifrado de difusión de mensajes cortos propuesto de forma teórica por el grupo de investigación [3], hemos comprobado que somos capaces utilizar una nueva manera de cifrar que nos permite ocultar las cabeceras MAC de los diferentes dispositivos que reciben paquetes confidenciales en una trama agregada haciendo que la transmisión sea más segura y eficiente. Para ello se ha propuesto una serie de casos donde se explica la manera en la que se cifra, envía y descifra la información utilizando un escenario de comunicación normal entre AP y STA.

6.2. Líneas futuras

Como línea futura hemos planteado la posibilidad de inyectar tramas utilizando el estándar 802.11 ac/ax para poder hacer comparaciones más precisas con el modelo propuesto [3] utilizando las normas de última generación.

Por otro lado, el reparto de claves entre el punto de acceso y los diferentes dispositivos que quieren recibir información es otro aspecto importante a tener en cuenta, ya que el modelo de trama agregada segura se basa en el cifrado de paquetes utilizando diferentes claves a lo largo de la comunicación. Para ello se propone él envió de una clave que sea como una semilla generadora de claves en los diferentes dispositivos, de tal manera que el punto de acceso también tenga esa semilla que mediante la sincronización con los otros equipos se generen claves dependiendo del número de paquetes que se quieran enviar y su tamaño. Pero esto es algo que deberá estudiarse con suficiente profundidad

Otra línea futura sería la creación de un protocolo de confirmación para el modelo de tramas agregadas seguras, ya que se tiene que implementar la manera de confirmar las tramas que se han recibido y en caso de que haya habido algún error con alguna de ellas decirle al transmisor que la retransmita.

7. Bibliografía

[1] Yazid, M., & A¨ıssani, D. A. (n.d.). *Performance Study of Frame Aggregation Mechanisms in the New Generation WiFi*.

[2] Khorov, E., Kiryanov, A., Lyakhov, A., & Bianchi, G. (2019). A tutorial on IEEE 802.11ax high efficiency WLANs. *IEEE Communications Surveys and Tutorials*, *21*(1). https://doi.org/10.1109/COMST.2018.2871099

[3] Salazar, J. L., Saldana, J., Fernández-Navajas, J., Ruiz-Mas, J., & Azuara, G. (n.d.). *Short Message Multichannel Broadcast Encryption*.

[4] Khorov, E., Kiryanov, A., & Lyakhov, A. (2016). IEEE 802.1 lax: How to build high efficiency WLANs. *Proceedings - 2nd International Conference on Engineering and Telecommunication, En and T 2015*. https://doi.org/10.1109/EnT.2015.23

[5] Prasithsangaree, P., & Krishnamurthy, P. (2003). Analysis of Energy Consumption of RC4 and AES Algorithms in Wireless LANs. *GLOBECOM - IEEE Global Telecommunications Conference*, *3*. https://doi.org/10.1109/glocom.2003.1258477

[6] Rachedi, A., & Benslimane, A. (2009). Impacts and solutions of control packets vulnerabilities with IEEE 802.11 MAC. *Wireless Communications and Mobile Computing*, *9*(4). https://doi.org/10.1002/wcm.690

[7] IEEE Computer Society. (2012). IEEE Standard 802.11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications IEEE Computer Society. In *Control* (Vol. 2012, Issue March).

[8] Hernández Fernández, C., María, J., Medina, S., José, P., & Más, R. (2015). *Trabajo Fin de Grado - Wi-Fi Frame grouping: Improving network efficiency by using aggregation mechanisms*.

[9] *Different Wi-Fi Protocols and Data Rates*. (n.d.). Retrieved September 18, 2022, from https://www.intel.com/content/www/us/en/support/articles/000005725/wireless/legacy-intel-wireless-products.html#legacy

[10] Golomb, S. W., & Scholtz, R. A. (1965). Generalized Barker Sequences. In *IEEE Transactions on Information Theory* (Vol. 11, Issue 4). https://doi.org/10.1109/TIT.1965.1053828

[11] *MCS Index and 7MCS™ Wi-Fi Experience Score*. (n.d.). Retrieved September 18, 2022, from https://www.7signal.com/info/mcs

[12] Coleman, D. (2021, March 1). *BSS Coloring – Definition and Role in Optimizing 802.11 ax*. https://www.extremenetworks.com/extreme-networks-blog/how-does-bss-coloring-work-in-802-11ax/

[13] *CSMA/CA: protocolo de acceso al medio para redes inalámbricas - IONOS*. (n.d.). Retrieved September 18, 2022, from https://www.ionos.es/digitalguide/servidores/know-how/csmaca-protocolo-de-acceso-al-medio-para-redes-inalambricas/

[14] *RTL8812AU-CG SINGLE-CHIP IEEE 802.11ac 2T2R WLAN CONTROLLER WITH USB 2.0/3.0 INTERFACE DATASHEET (CONFIDENTIAL: Development Partners Only)*. (2012). www.realtek.com

[15] Garcia Roca, D. (2021). *Trabajo Fin de Grado - Management and analysis of a virtual control scenario for a WiFi-based wireless network*.

[16] minetad. (2010). *CNAF 2010: NOTAS UN*.

[17] *Radiotap - Introduction*. (n.d.). Retrieved September 18, 2022, from https://www.radiotap.org/

[18] *Introduction — Scapy 2.5.0 documentation*. (n.d.). Retrieved September 18, 2022, from https://scapy.readthedocs.io/en/latest/introduction.html

[19] *02-WLAN Configuration Guides-02-Radio management configuration- 新华三集团-H3C*. (n.d.). Retrieved September 18, 2022, from http://www.h3c.com/en/Support/Resource_Center/Technical_Documents/Home/Switches/00-Public/Configure/Configuration_Guides/H3C_Unified_Wired_Wireless_(R5417P03)-6W103/02/201907/1211821_294551_0.html

[20] Benton, K. (2010). The Evolution of 802 . 11 Wireless Security. *Journal, UNLV Informatics-Spring*, *INF 795*.

[21] *GitHub - joserpl21/INYECCION_TRAMA_WIFI: Trabajo de fin de grado (Teleco)*. (n.d.). Retrieved September 18, 2022, from https://github.com/joserpl21/INYECCION_TRAMA_WIFI

[22] Nicolas, D. (2010, October 25). *802.11 frames : A starter guide to learn wireless sniffer traces - Cisco Community*. https://community.cisco.com/t5/wireless-mobility-knowledge-base/802-11-frames-a-starter-guide-to-learn-wireless-sniffer-traces/ta-p/3110019

ANEXO 1 Configuración escenario

1.1 Conexión remota a las antenas

Primero se necesita estar dentro de la red Wi-Fi de la universidad para ello tenemos dos opciones de conexión:

1. WIUZ
2. EDURAM

Las dos se pueden acceder con el NIP y la contraseña administrativa que proporciona la UNIZAR.

El siguiente paso es utilizar una VPN para acceder al laboratorio para ello hemos creado una conexión llamada TFG cuyas propiedades de conexión son las siguientes:

Nombre: TFG
Nombre de servidor o dirección: 155.210.158.30
Tipo de información de inicio de sesión: Nombre de usuario y contraseña

Para que se pueda acceder a internet mediante el túnel VPN tenemos que desactivar la opción de crear Gateway por defecto de la conexión, como en la siguiente imagen.

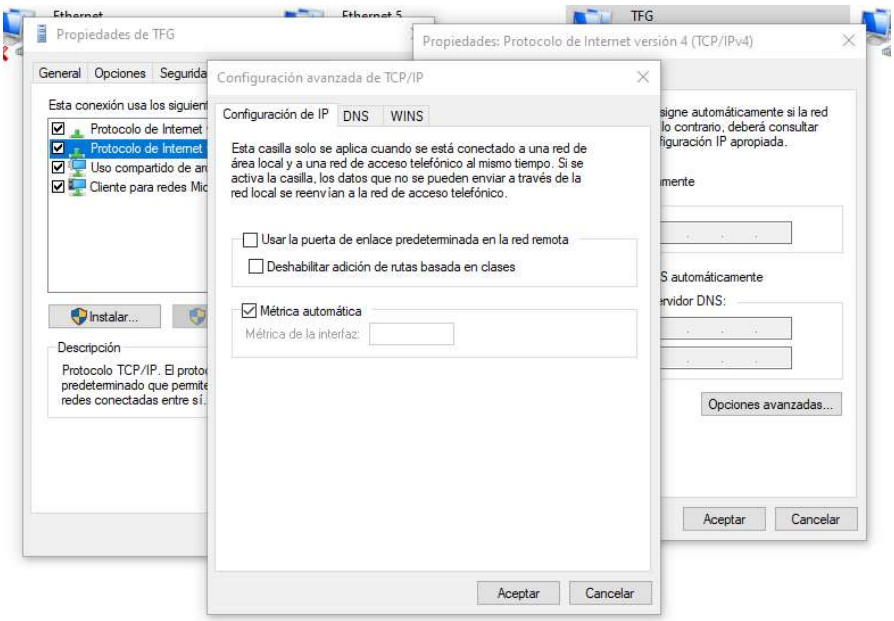

Figura 44: configuración avanzada del VPN TFG

Una vez dentro de la red del laboratorio se nos ha asignado la dirección 155.210157.160:3082 para conectarnos al equipo que tendrá las conexiones de las antenas mediante los puertos USB.

1.2 Configuración completa de template

1. Estar en la red del laboratorio.
2. Descargamos la imagen del sistema operativo que tiene el servidor GNS3 que en este caso es el Debian 10 minimal, para ello podemos conectarnos mediante SSH.
3. En GNS3 añadir una conexión a un servidor remoto como en la siguiente imagen:

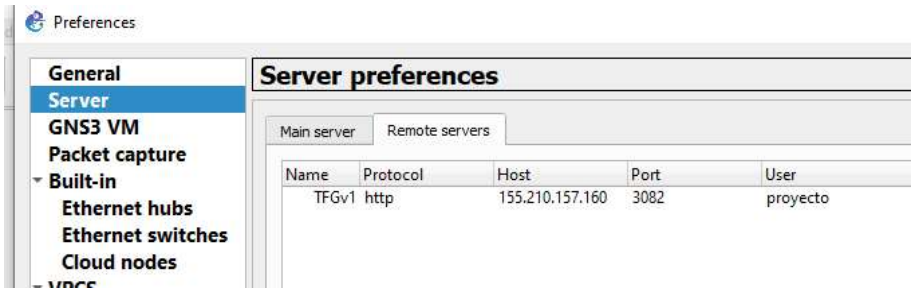

Figura 45: configuración de servidor remoto en GNS3

4. Creamos un template QEMU con la opción que se pueda ejecutar en un servidor remoto, seleccionamos el servidor remoto creado anteriormente.

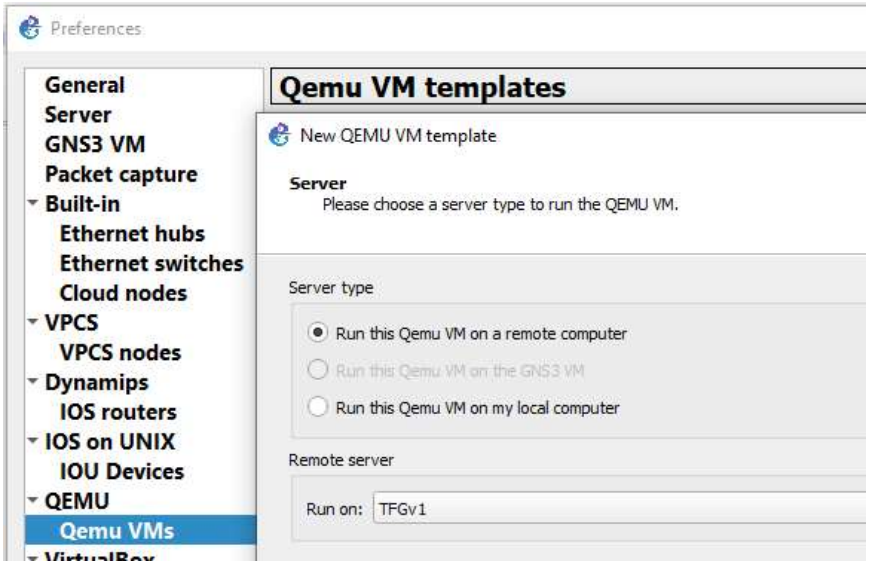

Figura 46: configuración de QEMY VM

5. Seleccionamos la versión x86_64(v4.2.1) de QEMU, le asignamos un tamaño de memoria RAM y elegimos la opción de consola telnet.
6. Selección de imagen de disco, hay dos opciones: utilizar la imagen descargada anteriormente del equipo físico o utilizar las imágenes que se encuentran dentro del equipo remoto, nosotros escogeremos la primera opción.
7. Agregamos el comando para controlar los puertos USB del equipo.

1.3 Instalación repositorios

Comandos para instalar python3 en un sistema operativo Linux.

```
sudo apt-get update
sudo apt-get install python3
sudo apt-get install python3-pip
sudo apt-get install python3-scapy
```

Comandos para instalar todas las librerías de python3 necesarias para implementar el modelo de agregación de tramas seguras.

```
pip3 install random2
pip3 install pickle
pip3 install python-secrets
pip3 install pycryptodome
pip3 install pycopy-binascii
pip3 install zlib-state
sudo apt-get update -y
sudo apt-get install -y python3-gmpy2
```

ANEXO 2 Implementación del teorema chino de los restos

2.1 Cálculo de claves

Una de las principales partes de nuestro modelo de agregación de tramas seguras es la creación y ubicación de estas, por ello a continuación explicaremos todo el proceso.

Para la creación de las claves se utiliza la función calc_primos ubicada dentro del fichero datos.py de la rama ***encryptado*** en los repositorios GitHub. Esta función tiene como parámetros de entrada el número de los paquetes que quiere cifrar y la longitud de las claves privadas. En el caso de la clave pública Hdr tendrá el máximo tamaño posible ya que esto nos permitirá cifrar cualquier paquete sin problema. La función devolverá las claves privadas en 2 vectores la primera de claves PS y la segunda de claves XS, además, de una clave pública Hdr.

```
def calc_primos(num, long):
    psnew = []
    xsnew = []
    Hdr = random.getrandbits(2047)
    m = 2 ** (long * 8)
    for i in list(range(num)):
        pnew = gmpy2.nextprime(m + 3)
        m = pnew
        psnew.append(pnew)
        xsnew.append(getrandbits(pnew.bit_length() - 1))
    return (psnew, xsnew, Hdr)
```

Debido a que las claves son muy extensas propondremos un ejemplo con valores pequeños. En este caso queremos enviar 2 paquetes y que las claves tengan una longitud de 5 por lo tendremos que ejecutar el siguiente comando:

[psnew, xsnew, Hdr]=calc_primos(2, 5)

```
Claves PS [mpz(263), mpz(269)]
Claves XS [83, 188]
Clave publica 27727257597068125567972504413588556955
```

Figura 47: Claves para cifrar 2 paquetes de longitud 5

En la figura 47 vemos las claves que se generan aleatoriamente al compilar la función anterior. Estas claves deben ser ubicadas dentro del código del transmisor (Tx_process.py) y en la de los receptores que estén esperando un paquete.

```
import sys
import scapy
from scapy.layers.dot11 import *
from scapy.layers.dot11 import Dot11, RadioTap, Dot11Beacon, Dot11Elt, Dot11ProbeResp

from datos_new import *

ap_list = []  # Lista de indices para controlar que paquetes nos han llegado
IFACE = 'wlx00c0caa4737b'  # Interfaz de la targeta de red del transmisor
AP_MAC = '00:c0:ca:a4:73:7b'  # Direccion mac del transmisor
AP_MAC_2 = '00:c0:ca:a4:73:7c'  # Direccion mac del receptor()
rates = 0  # Velocidad de transmision
forma = 0

#Clave cifrado normal
key = b'\x01\x0a\x03\x0a\x03\x0a\x03\x0a\x03\x0a\x03\x0a\x03\x0a\x03\x0a'
#Clave cifrado CRT
ps = [263,269]
xs = [83,188]
Hdr = 27727257597068125567972504413588556955269194931754944627328691356390550562943152
def PacketHandler(pkt):
    # Capturamos el Assoc request
    if pkt.subtype == 0:
        if pkt.subtype not in ap_list:
            infoPacket = pkt.getlayer(Dot11Elt)
            if infoPacket.info.decode() == "ap0":
```

Figura 48: Ubicación de claves en Tx_process.py – transmisor

En la figura 48 vemos la ubicación de las claves en el transmisor estos serán utilizados para cifrar todos los paquetes que serán enviadas dentro de una trama agregada segura. En la figura 49 vemos en qué parte del código son utilizadas.

```
if int(forma) == 2:
    packet = []
    qoscontrol = b'\x00\x00'
    mpduCi_Hex, mpduCi = cifrarCRT(ps,xs,Hdr)
    packet.append(RadioTap() / Dot11(type=2, subtype=8, addr1=pkt.addr2, addr2=AP_MAC, addr3=AP_MAC)
                  / qoscontrol / mpduCi_Hex)
    print("Paquete cifrado enviado")
    print(hexdump(mpduCi_Hex))
    sendp(packet, iface=IFACE, verbose=False)
```

Figura 49: Uso de las claves en el transmisor

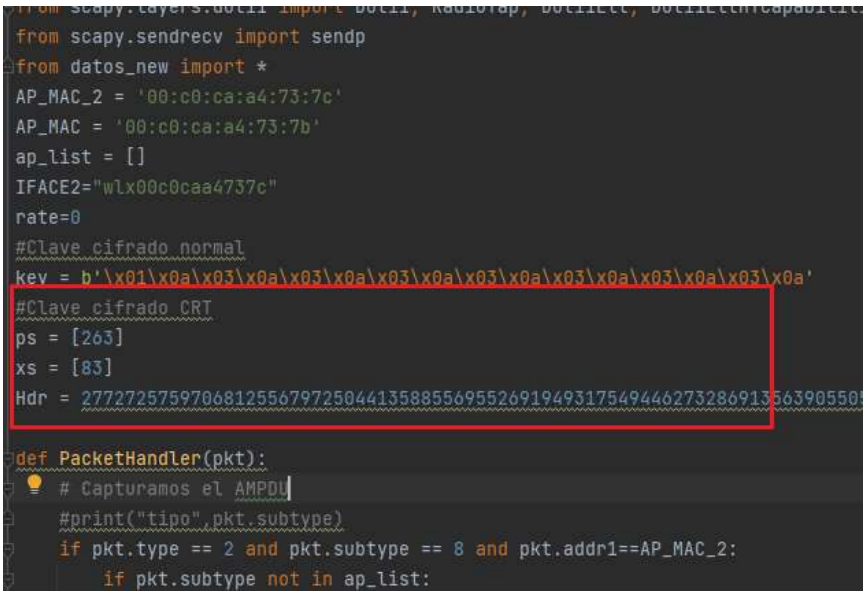

Figura 50: Ubicación de claves en Rx_process.py – Receptor 1

En la figura 50 vemos la ubicación de las claves en el receptor las cuales se utilizarán como vemos en la figura 51 para descifrar los paquetes que han sido destinados hacia él, en este caso podemos ver que tiene las claves necesarias para descifrar solo 1 paquete.

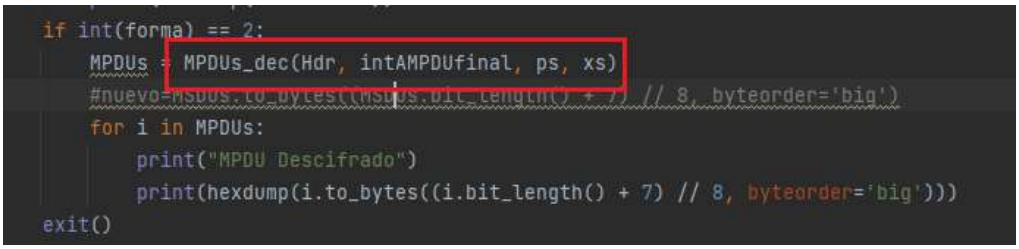

Figura 51: Uso de las claves en el receptor

2.2 Cifrado

El código se encuentra dentro del repositorio Criptotramas.py en la rama encryptado en los repositorios GitHub. La siguiente función permite cifrar un conjunto de paquetes utilizando el teorema de los restos chinos explicado en el apartado 2.3.1, para ello se ingresa el vector de paquetes de datos, el vector de claves PS, XS y la clave compartida Hdr. Donde es necesario que haya suficientes claves para cifrar todos los paquetes que se deseen enviar, además que todas las claves privadas tienen que ser distintas.

El resultado del cifrado es un solo paquete listo para ser transmitido de manera segura.

```
def MPDUs_enc(MPDUs,ps,xs,Hdr):
    a = []  # Máscaras aleatorias
    n = 0
    for i in MPDUs:
        a.append((i + pow(Hdr, xs[n], ps[n])) % ps[n])
        n = n + 1
    return chinese_remainder(ps, a)
```

Función que implementa el teorema chino de los restos para cifrar un conjunto de paquetes.

```
def chinese_remainder(n, a):
    sum = 0
    prod = functools.reduce(lambda a, b: a * b, n)
    for n_i, a_i in zip(n, a):
        p = prod // n_i
        sum += a_i * mul_inv(p, n_i) * p
    return sum % prod
```

Función para calcular el inverso multiplicativo modular.

```
def mul_inv(a, b):
    b0 = b
    x0, x1 = 0, 1
    if b == 1: return 1
    while a > 1:
        q = a // b
        a, b = b, a % b
        x0, x1 = x1 - q * x0, x0
    if x1 < 0:
            x1 += b0
    return x1
```

2.3 Descifrado

El código se encuentra dentro del repositorio Criptotramas.py en la rama encryptado en los repositorios GitHub. La función permite descifrar un paquete cifrado con CRT utilizando las claves PS, XS y Hdr. En este caso a diferencia de la función de cifrado, no es necesario que haya suficientes claves para descifrar todos los paquetes. Para su implementación tiene que haber como mínimo claves suficientes para descifrar al menos un paquete.

```
def MPDUs_dec(Hdr, MPDU, ps, xs):
    MPDUs = []
    for i in range(len(ps)):
        MPDUs.append((MPDU - pow(Hdr, xs[i], ps[i])) % ps[i])
    return (MPDUs)
```

ANEXO 3 Inyección de tramas

3.1 Creación de trama completa

Para enviar una trama aparte de la cabecera Radiotap necesitamos una cabecera MAC y finalmente toda la información asociada al tipo de trama que sea[22]. En la figura 52 vemos la estructura básica del paquete.

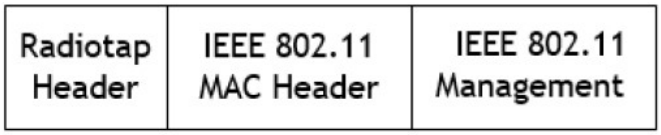

Figura 52: Estructura de una trama Wi-Fi

A continuación, explicaremos en qué consisten las partes mencionadas anteriormente mediante el código para la inyección de un Beacon. En la figura 53.2 vemos la cabecera Radiotap que tiene todos los parámetros por defecto, en la figura 53.3 es donde se indica el tipo de trama que es [23]. En este caso indicamos que se trata de un Beacon por lo cual se irá transmitiendo de manera constante por el AP para que los diferentes dispositivos puedan conectarse a él. En la imagen 53.4 vemos las direcciones MAC. La addr1 que indica la dirección destino, el addr2 y addr3 es la dirección de origen ya que estamos transmitiendo desde el AP. El siguiente campo sería el de QoS necesario para añadir información de calidad y servicio del paquete, pero en este caso no lo pondremos al tratarse de un Beacon. Finalmente vemos en la figura 53.5 toda la información que quiere enviar la trama. Al tratarse de un Beacon enviamos información del SSID cuyo nombre es “ap0”.

```
ap_list = []  # Lista de indices para controlar que paquetes nos han llegado
IFACE = 'wlx00c0caa4737b'  # Interfaz de la targeta de red del transmisor
AP_MAC = '00:c0:ca:a4:73:7b'  # Direccion mac del transmisor
AP_MAC_2 = '00:c0:ca:a4:73:7c'  # Direccion mac del receptor()
rates = 0  # Velocidad de transmision
forma = 0
packet = []
packet.append(RadioTap() / Dot11(subtype=8, addr1='ff:ff:ff:ff:ff:ff', addr2=AP_MAC, addr3=AP_MAC)
              / Dot11Beacon(cap="ESS+CFP") / Dot11Elt(ID='SSID', info="ap0") )
# Enviamos el Beacon
sendp(packet, iface=IFACE, verbose=False)
```

Figura 53: Creación y envío de una trama (Beacon)

Antes de inyectar el paquete es necesario que la antena esté configurada como se indica en el apartado 3.3.4. En la figura 53.6 vemos que enviamos el paquete mediante la interfaz de la figura 53.1.

ANEXO 4 Recepción de paquetes

4.1 Captura de trama

Es necesario que la antena que transmite el paquete esté en el mismo canal que la antena de recepción. Tal como explicamos en el apartado 3.3.4.

En la figura 54.1 vemos el nombre de la interfaz inalámbrica por donde capturaremos el tráfico Wi-Fi, en la figura 54.4 utilizamos la función Sniff para capturar el tráfico en esa interfaz con un timeout elevado para capturar diferentes paquetes. Cuando la interfaz capture algún paquete se ejecutará la función que vemos en la figura 54.2 y todo el paquete se guardará en la variable pkt. De este paquete podemos comprobar diferentes parámetros como el tipo y subtipo de trama[22] tal como vemos en la figura 54.3.

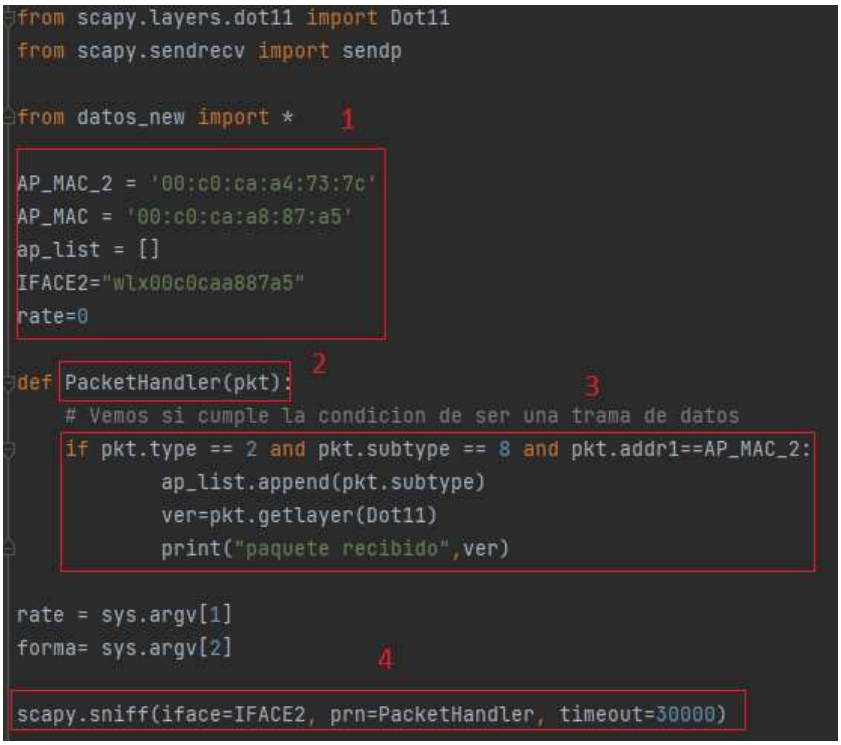

Figura 54: Ejemplo de código de recepción de trama

4.2 Estructura de trama agregada segura

En la figura 55.2 vemos las claves suficientes para cifrar 5 paquetes. Si se quisiera cifrar más o menos paquetes se tendrían que colocar todas las claves en esa ubicación tal como se explica en el anexo 2.1. En la figura 55.5 construimos la trama que inyectaremos, debido a la flexibilidad a la hora de utilizar un estándar de transmisión en nuestro modelo de trama agregada segura utilizamos una cabecera Radiotap por defecto. Si bien podemos modificar esta misma para inyectarla en otro estándar tal como explicamos en el apartado 4.2.

En la cabecera MAC de la figura 55.5 indicamos que se trata de una trama de datos [23]. Luego asignamos las direcciones MAC de origen y destino. En este caso al ser una trama de datos ponemos un valor de QoS. Después agregamos los datos cifrados obtenidos de la función cifrarCRT. Finalmente, en la figura 55.6 vemos como inyectamos la trama mediante la interfaz de la figura 55.1.

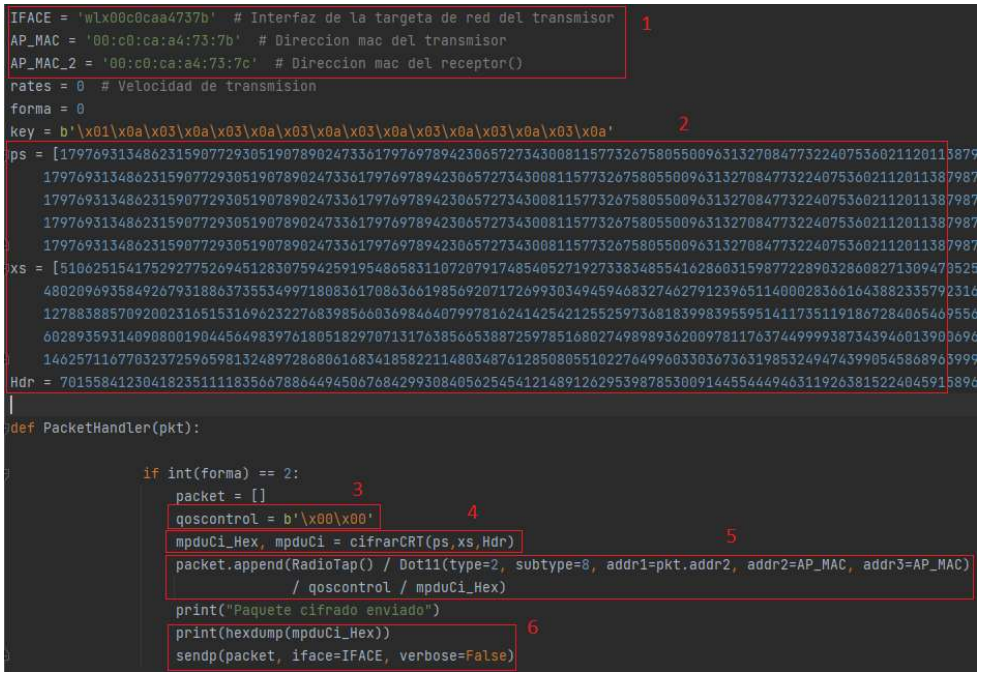

Figura 55: Creación y envío de una trama agregada segura

En la figura 56 vemos el código de la función cifrarCRT el cual se encuentra dentro del fichero datos.py de la rama ***encryptado*** de los repositorios de GitHub. La figura 56.2 muestra la construcción de paquetes MSDUs de diferentes longitudes mediante la función AMSDU_gen. En el que se indica el número de paquetes que se quiere obtener y su respectiva longitud de payload (datos útiles sin contar las cabeceras). En este caso construimos 5 paquetes con diferentes tamaños de payload. En la figura 56.3 convertimos estos MSDUs en MPDUs mediante la función MPDU_gen que se encuentra en el fichero Generatramas.py. Esta función tiene como datos de entrada un paquete MSDU y un valor entero para indicar qué dirección MAC destino utilizar, tal como vemos en la figura 57 podemos asignar una dirección destino hacia el receptor 1 (Debian10_rx), al receptor 2 (Debian_rx2) o genera una dirección aleatoria.

Agregamos todos los MPDUs obtenidos con anterioridad a un vector para luego cifrarlos mediante la función MPDUs_enc explicado en el anexo 2.3. Finalmente devuelve la trama cifrada en formato numérico y binario.

Esta función se puede modificar para crear diferentes MPDUs de diferentes longitudes para luego cifrarlas, siempre y cuando se tengan las claves necesarias para todas ellas. Esto se puede implementar mediante la creación de claves explicadas en el anexo 2.1.

```python
from Generatramas_new import *
from Criptotramas_new import *
from random import getrandbits
import scapy.all as scapy
from datetime import datetime, timedelta

def cifrarNORMAL(key):  #
    f = open('datos3_new.DAT', "r")
    AMPDUs = AMPDU_enc2(f, key)
    cifrado = AMPDUs[0].to_bytes((AMPDUs[0].bit_length() + 7) // 8, byteorder='big')
    return (cifrado, AMPDUs)
def cifrarCRT(ps,xs,Hdr):  1
    AMSDU, MSDUs1 = AMSDU_gen(1, 10)  2
    AMSDU, MSDUs2 = AMSDU_gen(1, 20)
    AMSDU, MSDUs3 = AMSDU_gen(1, 30)
    AMSDU, MSDUs4 = AMSDU_gen(1, 40)
    AMSDU, MSDUs5 = AMSDU_gen(1, 50)
    MPDU_Total=[]  3
    MPDU_Total.append(MPDU_gen(MSDUs5[0],2))
    MPDU_Total.append(MPDU_gen(MSDUs2[0],1))
    MPDU_Total.append(MPDU_gen(MSDUs4[0],1))
    MPDU_Total.append(MPDU_gen(MSDUs1[0],0))
    MPDU_Total.append(MPDU_gen(MSDUs3[0],1))
    mpduCi=MPDUs_enc(MPDU_Total,ps,xs,Hdr)  4
    mpduCi_Hex= mpduCi.to_bytes((mpduCi.bit_length() + 7) // 8,byteorder='big')

    return (mpduCi_Hex,mpduCi)  5
```

Figura 56: Creación y cifrado de paquetes MPDUs

```python
def MPDU_gen (MSDU,opc):
    frame_control = b'\x20\xe5'
    Duration = b'\x11\x00'
    Address2 = b'\x00\xc0\xca\xa4\x73\x7b'  # No sabemos
    Address3 = b'\x00\xc0\xca\xa4\x73\x7b'
    if opc==1:
        Address1 = b'\x00\xc0\xca\xa4\x73\x7c'

    elif opc==2:
        Address1 = b'\x00\xc0\xca\xa8\x87\xa5'
    else:
        Address1 = secrets.token_bytes(6)  # No sabemos a

    Sequence_control = b'\x01\x00'
    Address4 = b'\x01\xc0\xca\xa4\x73\x7b'
```

Figura 57: Opciones de Mac destino en la creación de MPDUs

ANEXO 5

5.1 Descargar repositorios GitHub

Como requisito inicial se pide tener instalado el programa Git el cual es un software de control de versiones. En la figura 41 vemos cómo está estructurado el código del proyecto en dos diferentes ramas:

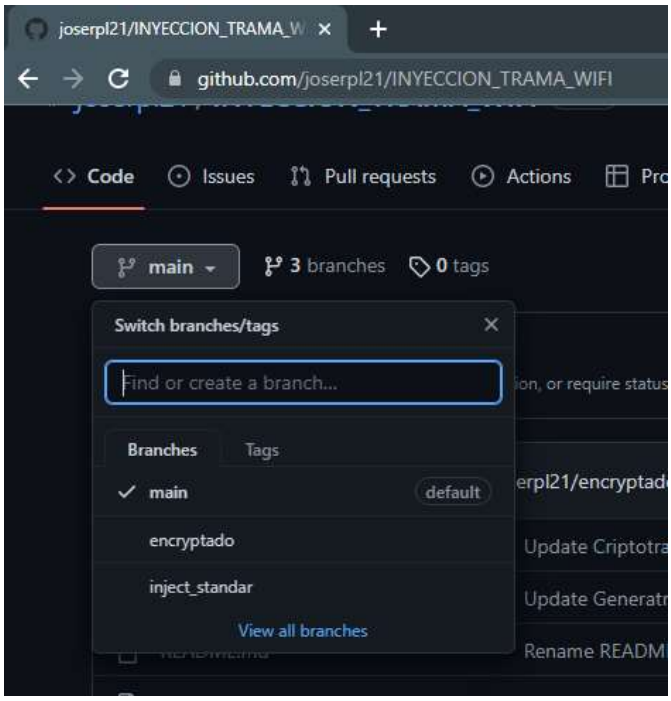

Figura 58: Estructura de repositorio en GitHub

El comando para descargar cada rama son las siguientes:

- git clone – branch inject_standar https://github.com/joserpl21/INYECCION_TRAMA_WIFI.git

- git clone –branch encryptado https://github.com/joserpl21/INYECCION_TRAMA_WIFI.git

índice de figuras