UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Análisis de paquetes con Wireshark, estudio de vulnerabilidades.

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Iván Mitkov Angelov

Tutor: Juan Carlos Martínez González

2020-2021

Resumen

El siguiente trabajo se centra en ilustrar, de forma profunda y detallada, cómo funcionan ciertos protocolos y sistemas que usamos en nuestra vida cotidiana al conectarnos a internet. Mediante el analizador de protocolos Wireshark, se estudian exhaustivamente todas las piezas que construyen algunos protocolos y sistemas fundamentales para el correcto funcionamiento de nuestras redes.

Para nuestro estudio elegimos analizar el protocolo ARP (*Address Resolution Protocol*), desglosarlo con Wireshark y demostrar cómo funciona un ataque MITM con *ARP Cache Poisoning*. También explicaremos en detalle cómo funcionan los protocolos HTTP y HTTPS.

Para cada uno de los protocolos y sistemas elegidos para realizar el estudio, hemos diseccionado los paquetes que los forman para comprender exactamente qué sucede. Acto seguido, se comentan las vulnerabilidades que tienen y se intentan simular escenarios donde se explotan las susodichas, ilustrando en profundidad cómo funciona. Para ello, se emplean máquinas virtuales con un sistema operativo especializado en ciberseguridad, por ejemplo, Kali Linux, que incluye varias herramientas muy útiles. Por último, se proponen alternativas o parches que mejoran la seguridad, con ejemplos prácticos que ilustran las ventajas de estas.

Como conclusión, justificaremos la utilización en la práctica de unos u otros protocolos, aportando nuestro análisis y datos.

Palabras clave: protocolo, Wireshark, vulnerabilidad, paquetes, Kali Linux

Abstract

The following work is centered around illustrating in a detailed way how certain protocols and systems, which we use in our daily lives, function in our networks. With the help of the protocol analyzer Wireshark, we will thoroughly analyze each of the pieces which build some of the fundamental protocols and systems that allow our networks to work properly.

For our study we choose to analyze the ARP (Address Resolution Protocol) protocol, break it down with Wireshark and show how an ARP Cache Poisoning attack works. We are also going to explain how the HTTP and HTTPS protocols work.

For each of the protocols and systems we have chosen for the study, we dissect the packets that make them up to understand exactly what happens. Then, we discuss the vulnerabilities they have and simulate scenarios where the aforementioned are exploited, illustrating in depth how it works. For this, virtual machines with an operating system specialized in cybersecurity are used, for example, Kali Linux, which includes several very useful tools. Finally, alternatives or patches that improve security are proposed, with practical examples that illustrate the advantages of these. In conclusion, we justify the use in practice of one or the other protocols, providing our analysis and data.

Keywords: protocol, Wireshark, vulnerability, packets, Kali Linux

Resum

El següent treball se centra en il·lustrar, de manera profunda i detallada, com funcionen uns certs protocols i sistemes que usem en la nostra vida quotidiana en connectar-nos a internet. Mitjançant l'analitzador de protocols Wireshark, s'estudien exhaustivament totes les peces que construeixen alguns protocols i sistemes fonamentals per al correcte funcionament de les nostres xarxes.

Per al nostre estudi triem analitzar el protocol ARP (*Address Resolution Protocol*) desglossar-lo amb Wireshark i demostrar com funciona un atac MITM (*Man In The Middle*) amb *ARP Cache Poisoning*.

Per a cadascun dels protocols i sistemes triats per a realitzar l'estudi, hem disseccionat els paquets que els formen per a comprendre exactament què succeeix. Tot seguit, es comenten les vulnerabilitats que tenen i s'intenten simular escenaris on s'exploten les susdites, il·lustrant en profunditat com funciona. Per a això, s'empren màquines virtuals amb un sistema operatiu especialitzat en ciberseguretat, per exemple, Kali Linux, que inclou diverses eines molt útils. Finalment, es proposen alternatives o pegats que milloren la seguretat, amb exemples pràctics que il·lustren els avantatges d'aquestes.

Com a conclusió, justificarem la utilització en la pràctica d'els uns o els altres protocols, aportant la nostra anàlisi i dades.

Palabras clave: protocol, Wireshark, vulnerabilitat, paquets, Kali Linux

Tabla de contenidos

1. Introducción

Actualmente, comunicarse con casi cualquier persona resulta muy sencillo a través de un *smartphone* o un ordenador. El auge de las telecomunicaciones ha supuesto un antes y un después en la manera que tenemos de entender el mundo. Cada vez con más frecuencia nos llegan noticias sobre ciberseguridad y vulnerabilidades en nuestras aplicaciones. Todos estos avances se basan en cómo creamos redes, cómo funcionan y qué vulnerabilidades tienen. En este trabajo vamos a revisar protocolos a tres niveles distintos según el estándar OSI, mostrar cómo deberían funcionar correctamente mediante el analizador de protocolos Wireshark, explicando paso por paso qué sucede. A continuación, discutiremos las vulnerabilidades que presentan desde un punto de vista teórico. Por último, mediante Kali Linux explotaremos las vulnerabilidades y mostraremos mediante Wireshark cómo se producen.

1.1. Motivación

Las redes de computadoras son un elemento fundamental en nuestro día a día y su correcto funcionamiento es un requisito para casi cualquier empresa. Con el paso de los años nos hemos ido concienciando sobre cómo necesitamos medidas de seguridad en nuestras redes. Este trabajo consiste en una aproximación muy básica al mundo de la seguridad en las redes, centrándonos en protocolos sencillos que se usan diariamente en cualquier red.

Por otra parte, a mí siempre me ha llamado el mundo de la ciberseguridad y quería realizar un trabajo que desarrollase algunas de las ideas estudiadas en la carrera, sobre todo en este último año con las asignaturas de RCO (redes corporativas) y SSR (sistemas y servicios de red). Se ha partido de lo aprendido y se han desarrollado ciertos aspectos centrados sobre todo en la parte de seguridad.

1.2. Objetivos

El propósito de este trabajo de fin de grado será el de ilustrar cómo funcionan los protocolos ARP, HTTP y HTTPS analizando los paquetes de datos a través de Wireshark y explicando porqué se dan las vulnerabilidades que permiten este tipo de ataques. Acto seguido, se propondrán soluciones y alternativas para solventar las vulnerabilidades previamente observadas. Esquematizando:

- Explicar el funcionamiento de los protocolos ARP, HTTP, HTTPS y las tecnologías necesarias para su implementación.
- Analizar el tráfico de datos mediante Wireshark.
- Identificar las vulnerabilidades.
- Mostrar de manera sencilla mediante Wireshark cómo son explotables.
- Proponer soluciones o alternativas.

1.3. Estructura de la memoria

Primero, en el capítulo 2 - Estado del arte se comentan los últimos avances en materia de ciberseguridad, comentando brevemente los diferentes tipos de ataques posibles.

A continuación, capítulo 3 – Fundamentos Teóricos se expondrán las tecnologías sobre las que se basan los protocolos que vamos a estudiar, ofreciendo una explicación detallada sobre cada una de ellas, el modelo OSI, criptografía asimétrica, ataques MITM y varios protocolos.

Después, en el capítulo 4 – Metodología se describirán brevemente los procedimientos que usaremos para llevar a cabo nuestro análisis.

Avanzando en el tema, en el capítulo 5 – Desarrollo nos pondremos manos a la obra, ilustraremos el correcto funcionamiento de los protocolos mediante Wireshark para acto seguido tratar de explotar sus vulnerabilidades mediante las herramientas proporcionadas por Kali Linux.

A continuación, en el capítulo 6 – Resultados discutiremos qué hemos logrado y haremos un pequeño repaso de lo conseguido al acabar el capítulo 5.

Llegados a este punto, en el capítulo 7 – Conclusiones analizaremos los resultados obtenidos y los compararemos con los objetivos, mencionando también posibles mejoras o alternativas.

Para terminar, en el capítulo 8 – Trabajos futuros propondremos temas de interés y posibles ampliaciones a nuestro estudio.

2. Estado del arte

A lo largo de este capítulo, comentaremos ataques típicos que pueden comprometer el correcto funcionamiento de las redes de computadores y comentar algún ejemplo reciente.

2.1. Malware

En primer lugar, vamos a tratar el *malware* (programas malignos). Estos son un tipo de programas informáticos diseñados con la intención de causar daños a ordenadores, servidores o a redes enteras. Existen diversas maneras mediante las cuales estos programas se propagan, como, por ejemplo: abriendo archivos ejecutables, accediendo a sitios web no seguros, descargando archivos sospechosos o mediante dispositivos de almacenamiento infectados.

Veamos un par de ataques actuales:

- Stuxnet: es un gusano informático (programa que crea copias de sí mismo con la finalidad de infectar a otros ordenadores) que tiene como objetivo sistemas SCADA (un tipo de sistema informático que se encarga de controlar y supervisar procesos industriales a distancia). Se propaga inicialmente a través de memorias USB infectadas. Su finalidad es acceder a dispositivos PLC (*Programmable Logic Controller),* ordenadores utilizados en ingeniería automática para automatizar procesos) y, una vez dentro, podía monitorizar el código y los datos transmitidos, introducir su propio código reemplazando el ya existente y enmascarar sus acciones. Además, contaba con dos *webs* que se utilizaron como servidores para Stuxnet, permitiéndole actualizarse y subir información confidencial. [1]
- Wannacry: fue un ciberataque a escala global propiciado por el criptogusano (es un gusano informático diseñado mediante técnicas criptográficas) Wannacry dirigido a los sistemas operativos Windows. Funciona cifrando los datos de ordenadores y pidiendo bitcoins a cambio de desencriptarlos. Se propaga a través de una vulnerabilidad en el puerto 445 TCP. Se estima que infectó a unos 300.000 ordenadores y causó daños estimados en los cientos de millones de euros. [2]

2.2. DoS

A continuación, vamos a enfocarnos en los ataques de denegación de servicio (Dos *Denial of Service*). Estos ataques se basan en interrumpir servicios o imposibilitar el acceso recursos a usuarios de estos, típicamente mediante el lanzamiento continuado de peticiones superfluas a alguna máquina, causando la sobrecarga del sistema. En la actualidad, se utilizan a menudo redes de *bots* (conjunto de robots informáticos que se ejecutan de manera autónoma) para causar ataques DDos (*Distributed Denial of Service*). Estos se diferencian en que el ataque se lanza desde varios puntos de conexión.

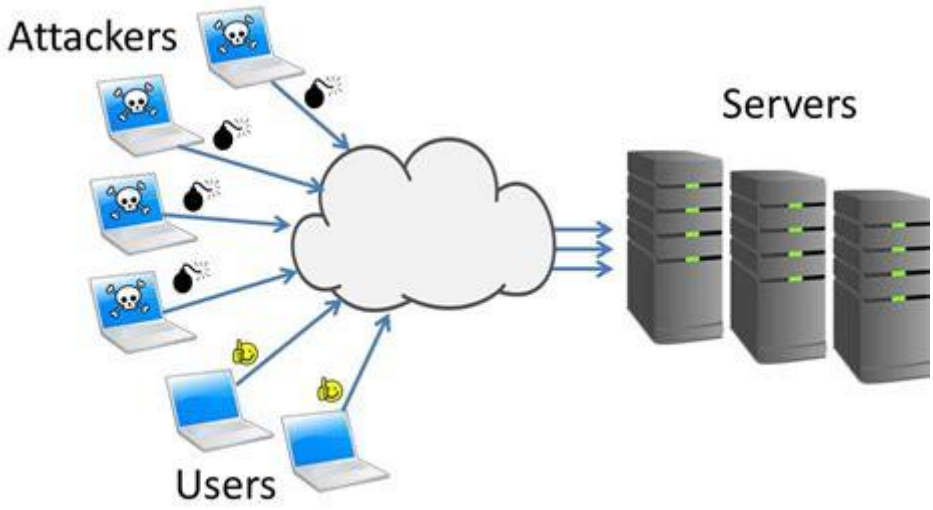


Figura 2.1: Ejemplo de ataque de denegación de servicio.

Como ejemplo, vamos a ver el caso de la botnet Mirai.Parte de un *malware* que tiene como objetivos dispositivos de la IOT (*Internet Of Things,* una red de objetos físicos conectados a internet) como *routers* o cámaras IP. Funciona escaneando internet continuamente en busca de dispositivos vulnerables usando nombres y contraseñas comunes de fábrica. Esto sucede porque una gran parte de estos dispositivos siguen teniendo sus ajustes de fábrica. A continuación, estas máquinas se usan para lanzar peticiones a los servidores objetivos y causar una caída del servicio. Esta red de *bots* ha sido responsable del ataque contra el proveedor de DNS (*Domain Name System*) Dyn que dejó páginas como Reddit, Github o Twitter inaccesibles. [3][4][5]

2.3. *Eavesdropping*

Se trata de un método que recopilar información de usuario a través de internet. Típicamente se produce cuando usuarios se conectan a redes públicas inseguras. Es considerado una de las amenazas más urgentes para las industrias que se basan en el almacenamiento de datos.

Un ejemplo de este tipo de ataques sería el programa de vigilancia PRISM.

2.3.1. MITM

Es un tipo de ataque en el que se adquiere la capacidad de leer insertar y modificar a voluntad. Los ataques *Man In the Middle* se basan en interceptar la comunicación entre dos víctimas. En este trabajo se estudiará cómo usar este tipo de ataques para adquirir claves wifis, para llevar a cabo ataque de ARP *spoofing*, ilustrar mediante el analizador de protocolos Wireshark qué sucede con los paquetes desde el punto de vista de la víctima y la del atacante.

Para explicar cómo funciona se mostrará un ejemplo sencillo que muestra un ataque MITM. Supongamos el siguiente escenario:

Figura 2.2: Ataque MITM. Mallory intercepta la comunicación entre Alice y Bob.

Alice quiere enviar un mensaje a Bob mientras que Mallory desea interceptarlo.

1. Alice envía un mensaje a Bob, que es interceptado por Mallory:

Alice *"Hola Bob, soy Alice. Dame tu clave."* → Mallory Bob

2. Mallory reenvía este mensaje a Bob; Bob no puede decir que no es realmente de Alice:

Alice Mallory *"Hola Bob, soy Alice. Dame tu clave."* → Bob

3. Bob responde con su clave de cifrado:

Alice Mallory ← *[clave de Bob]* Bob

4. Mallory reemplaza la clave de Bob con la suya, y transmite esto a Alice, afirmando que es la clave de Bob:

Alice ← *[clave de Mallory]* Mallory Bob

5. Alicia encripta un mensaje con lo que ella cree que es la clave de Bob, pensando que sólo Bob puede leer:

Alice *"¡Nos vemos en la parada de autobús!" [Cifrada con la clave de Mallory]* → Mallory Bob

6. Sin embargo, debido a que en realidad estaba cifrada con la clave de Mallory, Mallory puede descifrarlo, leerlo, modificarlo (si se desea), volver a cifrar con la clave de Bob, y lo remitirá a Bob:

Alice Mallory "¡*Nos vemos en la furgoneta de al lado del río!" [Cifrada con la clave de Bob]* → Bob

En este caso se aprecia la facilidad que tiene un atacante para aprovecharse de la transmisión de mensaje a través de canales inseguros para manipular mensajes.

Como ejemplo reciente, tenemos la NSA (*National Security Agency*, una agencia de inteligencia del departamento de defensa de EEUU) haciéndose pasar por Google. Podían potencialmente espiar cualquier búsqueda de Google. Los usuarios que intentaban conectarse a Google eran redirigidos a servidores de la NSA que se hacían pasar por la página original. [6]

2.4. Ingeniería Social

Consiste en obtener información confidencial a través de la manipulación de usuarios legítimos. Un ejemplo típico es el *phishing*. Se trata de una técnica fraudulenta para obtener información privada. Uno de los ataques más comunes se basa en el envío de un correo electrónico haciéndose pasar por alguna autoridad que redirige a una página web que aparenta ser legítima, copiando el contenido y los logos de la página original para recolectar información personal.

Este tipo de ataques ocurren muy a menudo,

En la actualidad las redes sociales ofrecen a los atacantes el medio perfecto para recabar información, ya que muchos usuarios comparten datos confidenciales a través de estas.

3. Fundamentos teóricos

3.1. El modelo OSI

Vamos a empezar explicando una serie de conceptos fundamentales sobre redes. *OSI REFERENCE MODEL*: es un modelo de referencia para los protocolos de la red desarrollado por la "ISO" ("*International Standards Organization*"). [7]

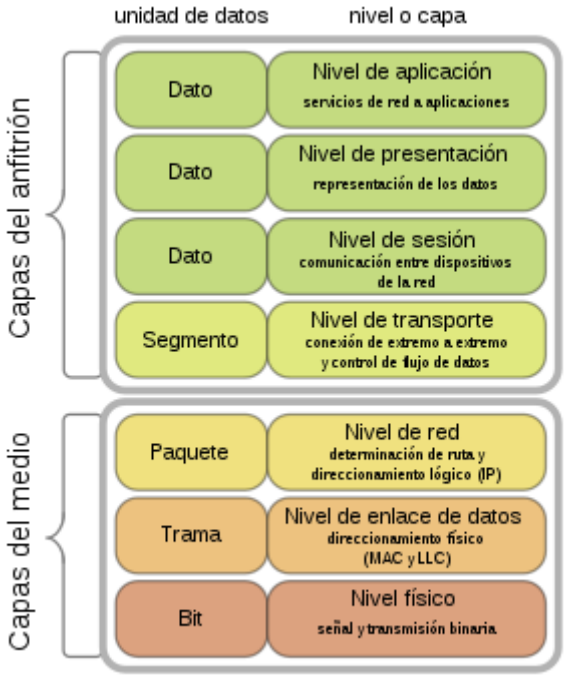


Figura 3.1: El modelo OSI.

Es un estándar que tiene por objetivo conseguir interconectar sistemas de procedencia distinta para que estos pudieran intercambiar información sin ningún tipo de impedimentos debido a los protocolos con los que estos operaban de forma propia según su fabricante.

El modelo OSI está formado por siete capas o niveles de abstracción. Cada uno de estos niveles tendrá sus propias funciones para que en conjunto sean capaces de poder alcanzar su objetivo final. Precisamente esta separación en niveles hace posible la intercomunicación de protocolos distintos al concentrar funciones específicas en cada nivel de operación.

Las capas que lo conforman son las siguientes:

- Nivel físico: proporciona los medios mecánicos, eléctricos, funcionales y de procedimiento para activar, mantener y desactivar conexiones físicas.
- Nivel de enlace de datos: es responsable de la transferencia fiable de información a través de un circuito de transmisión de datos. En este nivel vamos a realizar nuestro primer estudio de vulnerabilidades, mediante el estudio del protocolo ARP.
- Nivel de red: su misión es conseguir que los datos lleguen desde el origen al destino, aunque no tengan conexión directa. En este nivel vamos a realizar nuestro segundo estudio de vulnerabilidades.
- Nivel de transporte: está encargado de la transferencia libre de errores de los datos entre el emisor y el receptor, aunque no estén directamente conectados, así como de mantener el flujo de la red. Indagaremos un poco sobre el protocolo TCP (*Transmission Control Protocol*) pues es fundamental para entender HTTP.
- Nivel de sesión: proporciona los mecanismos para controlar el diálogo entre las aplicaciones de los sistemas finales.
- Nivel de presentación: es el que se encarga de la representación de la información, de manera que, aunque distintos equipos puedan tener diferentes representaciones internas de caracteres, números, sonido o imágenes, los datos lleguen de manera reconocible.
- Nivel de aplicación: Ofrece a las aplicaciones la posibilidad de acceder a los servicios de las demás capas y define los protocolos que utilizan las aplicaciones para intercambiar datos, como correo electrónico, gestores de bases de datos y protocolos de transferencia de archivos. Aquí se encuentran los protocolos HTTP y el HTTPS que veremos en detalle para finalizar nuestro análisis.

3.2. Criptografía asimétrica

Es un método criptográfico que usa un par de claves para el envío de mensajes. Ambas claves pertenecen a la misma persona que recibirá el mensaje. Una de ellas es pública y se puede entregar a cualquier persona, la otra es privada y no debe revelarse

a nadie. Los métodos criptográficos garantizan que no es posible que dos personas obtengan la misma pareja de claves. [8] [9]

Estas claves tienen dos usos principales:

- Cifrado de clave pública: un mensaje cifrado con la clave pública de un destinatario no puede ser descifrado por nadie (incluyendo al que lo cifró), excepto un poseedor de la clave privada correspondiente, presumiblemente su propietario y la persona asociada con la clave pública utilizada. Su función es garantizar la confidencialidad del mensaje.

Figura 3.2: Ejemplo de cifrado de clave pública.

Ejemplo de cifrado: Ana envía un mensaje a David. Veamos cómo funciona paso a paso:

1. Ana redacta un mensaje.
2. Ana cifra el mensaje con la clave pública de David.
3. Ana envía el mensaje cifrado a David a través de internet, ya sea por correo electrónico, mensajería instantánea o cualquier otro medio.
4. David recibe el mensaje cifrado y lo descifra con su clave privada.
5. David ya puede leer el mensaje original que le mandó Ana.

- Firmas digitales: un mensaje firmado con la clave privada del remitente puede ser verificado por cualquier persona que tenga acceso a la clave pública de dicho remitente, lo que demuestra que este remitente tenía acceso a la clave privada (y, por lo tanto, es probable que sea la persona asociada con la clave privada utilizada). Se asegura así que el mensaje no ha sido alterado, puesto que cualquier manipulación del mensaje repercutiría en un distinto resultado del algoritmo de resumen del mensaje (*encoded message digest*). Se utiliza para garantizar la autenticidad del mensaje.

Figura 3.3: Ejemplo de firma digital con clave asimétrica.

Ejemplo de firma digital con clave asimétrica:

1. David redacta un mensaje.
2. David firma digitalmente el mensaje con su clave privada.
3. David envía el mensaje firmado digitalmente a Ana a través de internet, ya sea por correo electrónico, mensajería instantánea o cualquier otro medio.
4. Ana recibe el mensaje firmado digitalmente y comprueba su autenticidad usando la clave pública de David.
5. Ana ya puede leer el mensaje con total seguridad de que ha sido David el remitente.

3.3. ARP

ARP es un protocolo de comunicación que actúa sobre la capa de enlace de datos del sistema OSI. Su funcionalidad es la de encontrar la dirección MAC (un identificador único que se asigna a una tarjeta o dispositivo de red) que corresponde a una dirección IP en una red local. [10]

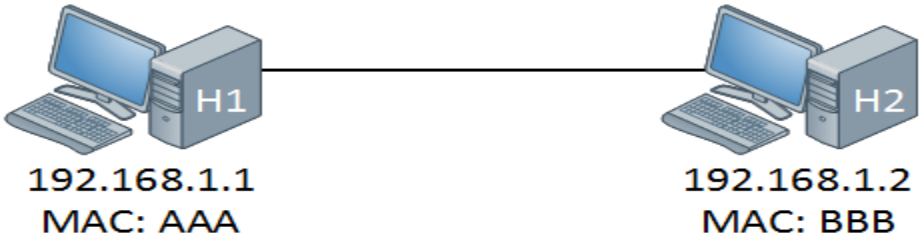


Figura 3.4: ARP.H1 necesita averiguar la MAC de H2.

En la imagen aparecen dos ordenadores, H1 y H2, con sus direcciones IP y MAC correspondientes. Si hacemos un ping desde H1 a H2 podemos usar la IP de H2. La capa de red se encarga del proceso. Sin embargo, al bajar a la capa de enlace nos encontramos que H1 no conoce la MAC de H2 y viceversa. En este momento es cuando interviene el protocolo ARP. Su funcionamiento se expone a continuación:

- El primer paso es mandar un ARP *request* desde H1 a H2. En esencia, se trata de un mensaje que pregunta: "¿Quién es 192.168?1.2 y cuál es tu dirección MAC?". Es importante destacar que este primer mensaje es un *broadcast* y llegará a todos los ordenadores de la red para averiguar cuál es la MAC de la IP proporcionada.
- Después el ordenador con la IP indicada contesta al mensaje de *broadcast*, con su dirección IP y su MAC.
- H1 crea su tabla de ARP y añade a la caché a H2.

La vulnerabilidad que permite un ataque MITM consiste en que ARP es un protocolo sin estado, cualquier respuesta se guarda en la caché, incluso sobre escribiendo entradas ya existentes cuando llegan nuevas. Al no haber método a través del cual el host pueda autenticar la identidad de los ordenadores, un atacante puede suplantar la identidad de algunos de los ordenadores de la red para manipularla. Los usuarios no pueden saber si un atacante ha suplantado la identidad del host. [11]

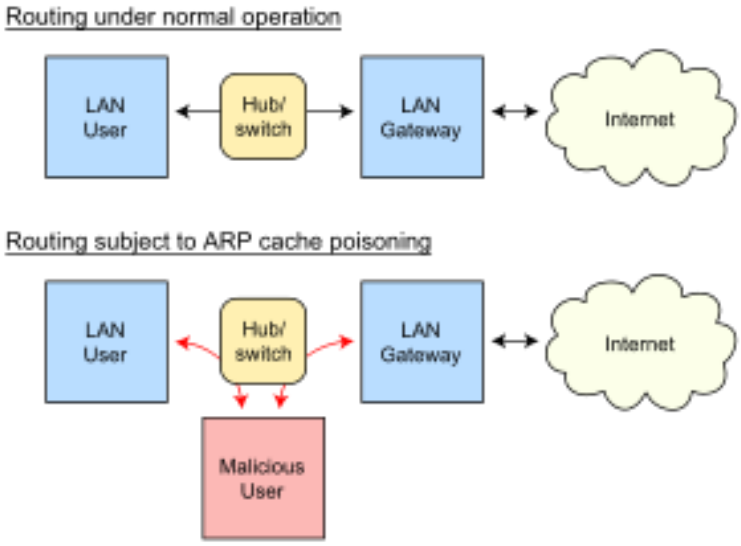


Figura 3.5: Ejemplo de ARP *cache poisoning*.

En general, el objetivo del atacante es inspeccionar los datos transmitidos, (comúnmente conocido como *eavesdropping*) aunque también podría lanzar un ataque de denegación de servicio.

3.4. HTTP

Es un protocolo de comunicación empleado para enviar datos a través de la *World Wide Web*. Requiere del protocolo de transporte TCP para funcionar.

Su funcionamiento típico es el siguiente:

1. Abrir una conexión con el puerto 80 al servidor de HTTP.
2. Se envía una petición con las cabeceras correspondientes.
3. El servidor procesa la información de la petición
4. El servidor envía la respuesta al cliente
5. Cierre de la conexión.

Los mensajes del protocolo HTTP tienen la siguiente estructura:

Línea inicial (termina con retorno de carro y un salto de línea) con

- Para las peticiones: la acción requerida por el servidor (método de petición) seguido de la URL del recurso y la versión HTTP que soporta el cliente.
- Para respuestas: La versión del HTTP usado seguido del código de respuesta (que indica qué ha pasado con la petición seguido de la URL del recurso) y de la frase asociada a dicho retorno.

Los métodos de petición habituales son:

- GET: para recuperar datos del servidor
- HEAD: para recuperar metadatos de los encabezados de respuesta sin tener que transportar todo el contenido
- POST: Envía datos para que sean procesados por el recurso identificado en la URL de la petición, se usa sobre todo para formularios.
- PUT: Similar a POST, pero se utiliza para actualizar contenidos.
- DELETE: Borra el recurso especificado.

En cuanto a las vulnerabilidades que presenta, estas son:

- Es un protocolo simple, fácilmente legible por un humano, los mensajes se muestran en texto plano.

- Es un protocolo sin estado, no guarda ninguna información sobre conexiones anteriores, son necesarias *cookies* para mantener la sesión e identificar al cliente. Las *cookies* no son más que un conjunto de datos enviado que un sitio web envía y son almacenadas por el navegador. [12]

3.4.1. TCP

El protocolo TCP (*Transmission Control Protocol*) permite a los programas crear conexiones entre sí a través de las cuales puede darse un flujo de datos. El protocolo garantiza que los datos serán entregados en su destino sin errores y en el mismo orden en que se transmitieron. Una parte importante de las comunicaciones que se producen en Internet emplean TCP. [13]

Su funcionamiento es el siguiente:

1. Establecimiento de la conexión: se usa la negociación en tres pasos (*3-way handshake*). El cliente realiza una apertura activa de un puerto enviando un paquete SYN inicial al servidor. El servidor comprueba si el puerto está abierto y si lo está responde al cliente con un mensaje SYN/ACK. Finalmente, el cliente responde con un ACK y comienza a enviar datos.

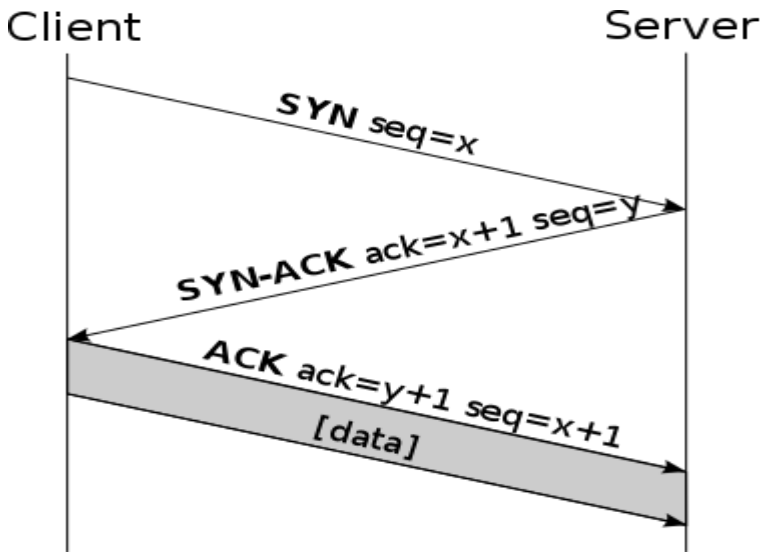


Figura 3.4.1.1: Protocolo TCP. Estableciendo conexión.

2. Transferencia de datos: Cliente y servidor intercambian datos, TCP incluye mecanismos para ordenar los paquetes TCP recibidos, detectar duplicados, pérdidas y retrasos.
3. Fin de la conexión: El cliente manda un mensaje FYN, el servidor contesta con un FYN/ACK. El cliente recibe el mensaje, manda un ACK al servidor y cierra su conexión. Cuando el servidor recibe el ACK, también cierra su conexión.

3.5. HTTPS

Es un protocolo de aplicación basado en el protocolo HTTP, destinado a la transferencia segura de datos de hipertexto, es decir, es la versión segura de HTTP. Está diseñado para resistir ataques de tipo MITM y *eavesdropping* (escuchar secretamente), que podrían obtener acceso a información confidencial.

A diferencia de HTTP, HTTPS usa los puertos 443 o 4433 y sus *URLs* comienzan por “https://”, muchos navegadores actuales indican que la conexión es segura si implementa HTTPS e insegura si se trata de HTTP.

HTTPS proporciona seguridad al implementar TLS (un protocolo criptográfico) para crear un canal cifrado en una red insegura para el tráfico de información sensible. En este caso el atacante solo tendría acceso a información cifrada si consigue interceptar la comunicación, que le serían imposibles de descifrar. TLS permite encriptar todo el protocolo HTTP subyacente, la URL, los parámetros de búsqueda, las cabeceras y las cookies. Sin embargo, como las direcciones de las páginas y los números de puerto forman parte de los protocolos TCP/IP, un atacante podría averiguar nuestra dirección IP,

Es importante destacar que a pesar de que HTTPS opera en la capa más alta del modelo OSI (aplicación), TLS opera en una capa inferior, la de transporte.

3.5.1. TLS

TSL es un protocolo criptográfico que proporciona comunicaciones seguras a través de una red. Usa criptografía asimétrica para autentificar (proceso de confirmar que algo es quién dice ser) la contraparte con quien se estén comunicando y para intercambiar una clave simétrica. Esta sesión es usada después para cifrar el flujo de datos entre cliente y servidor. Es un protocolo que se encuentra implementado en varias aplicaciones de uso extendido como navegación web, correo electrónico, mensajería instantánea y voz sobre IP. [14] [15]

Funcionamiento: El cliente realiza una petición de conexión a un servidor que implementa TLS, comúnmente por el puerto 443. Una vez servidor y cliente se ponen de acuerdo para usar TLS, negocian una conexión con estado mediante un *handshake* que funciona de la siguiente manera:

1. El cliente se conecta a un servidor que implementa TLS y pide abrir una conexión segura. El cliente debe también facilitar una lista de *cipher suits* (conjunto de algoritmos que ayudan a establecer un canal de comunicación seguro) soportados, un número aleatorio, especificar la versión más actualizada de TLS

que soporta y sugerir métodos de compresión. Esto se conoce como el mensaje *Client Hello*.

2. El servidor contesta con un mensaje *Server Hello*, en el que elige los parámetros de conexión de la lista proporcionada por el cliente en el mensaje anterior.
3. El servidor ofrece identificación a través de un certificado digital. Este, a su vez, contiene el nombre, del servidor, la CA (autoridad certificadora, entidad de confianza responsable de emitir y revocar certificados) y la clave de encriptación pública del servidor.
4. El cliente valida el certificado.
5. Se generan las claves de sesión, el cliente tiene dos alternativas:
   - Encriptar un número aleatorio (*PreMasterSecret*) con la clave pública del servidor y mandar el resultado al servidor. Ambos usan esta clave para generar un id de sesión único que utilizarán para encriptar y desencriptar los mensajes que intercambien.
   - Usar Diffie-Hellman (protocolo de establecimiento de claves entre dos partes que no han tenido contacto previo, utilizando un canal inseguro y de manera anónima) para generar las ids de sesión. Este método tiene la ventaja adicional de que, aunque se averigüe la clave privada del servidor en el futuro, no se podría desencriptar la comunicación.

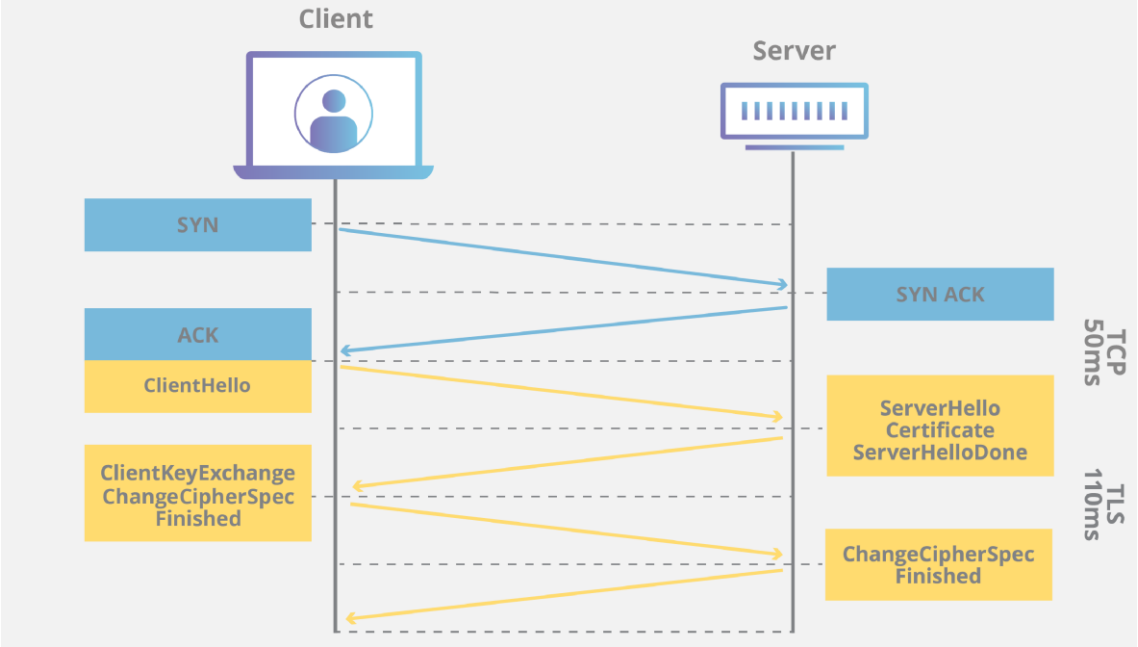


Figura 3.5.1.1: Funcionamiento del protocolo TLS.

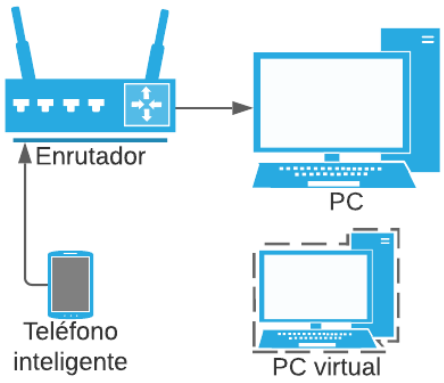
A pesar de la seguridad de este protocolo, existen vulnerabilidades.

4. Metodología

Con la finalidad de realizar un análisis del funcionamiento de los protocolos anteriormente mencionados, se seguirá la siguiente metodología:

1. Para cada uno de los protocolos estudiados, lanzaremos Wireshark y filtraremos los paquetes de interés en las capturas obtenidas.
2. Ejecutaremos los comandos necesarios para poner en marcha los protocolos.
3. Compararemos las capturas obtenidas con la descripción del protocolo dada en los fundamentos teóricos, explicando qué sucede y si el comportamiento es el esperado.
4. Trataremos de explotar las vulnerabilidades mencionadas en capítulos anteriores.
5. Volveremos a capturar datos con Wireshark.
6. Mostraremos cómo las capturas obtenidas cambian y explicaremos porqué.

Para realizar estos análisis, vamos a utilizar la siguiente configuración de red:



e

Figura 4.1: Configuración de red.

El enrutador está conectado mediante un cable Ethernet al PC y mediante wifi al teléfono inteligente. En el PC tenemos instalada una máquina virtual con el sistema operativo Kali Linux con una conexión de tipo *bridged* (puente).

Como herramientas a destacar, necesitaremos una máquina virtual (hemos elegido VMware), el sistema operativo Kali Linux por sus múltiples herramientas para detectar vulnerabilidades y explotarlas y Wireshark para explicar qué sucede en cada caso.

En el capítulo de resultados comentaremos brevemente lo conseguido y lo compararemos con los objetivos que nos habíamos propuesto.

Finalmente, en el capítulo de conclusiones haremos una interpretación del estudio y propondremos alternativas y soluciones.

5. Desarrollo

5.1. Análisis del problema

Se trata de analizar el funcionamiento de algunos protocolos básicos, ver cómo funcionan, explicar sus vulnerabilidades y mostrar de manera sencilla cómo se realizan los ataques sobre estos. Nos centraremos primero en el protocolo ARP y mediante Wireshark mostraremos cómo operan. Acto seguido, utilizaremos Kali Linux y Ettercap para realizar un ataque MITM.

Una vez completada la parte del protocolo ARP, haremos un estudio sobre los protocolos HTTP y HTTPS y explicaremos con la ayuda de Wireshark qué sucede al conectarnos a páginas que soportan estos protocolos.

5.2. Herramientas

Para realizar el análisis vamos a necesitar un programa para poder capturar datos, en nuestro caso, Wireshark. Este es un analizador de protocolos utilizado para llevar a cabo estudios y solucionar problemas en redes de comunicaciones. [16]

En nuestro caso vamos a utilizarlo para capturar tramas y analizar el tráfico de red de los protocolos que nos interesen. Incluye varias herramientas que nos permitirán ver qué sucede en cada momento.

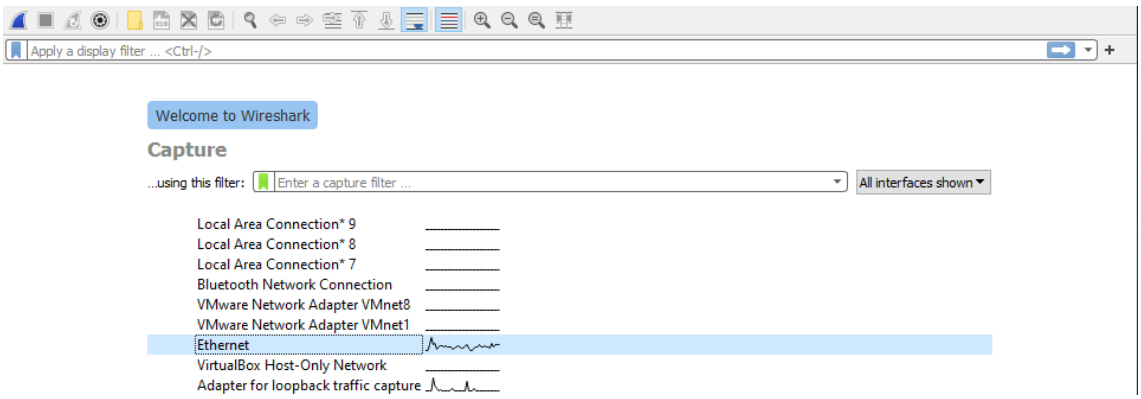


Figura 5.1: Interfaz de Wireshark.

Otra de las herramientas que vamos a necesitar es una máquina virtual (software que simula un sistema de computación y puede ejecutar programas como si fuese una computadora) para poder instalar el sistema operativo desde el cual realizaremos los ataques. Se ha elegido utilizar VMware, por la experiencia obtenida con esta en la asignatura de redes corporativas.

Como sistema operativo desde el cual lanzar los ataques y capturar paquetes vamos a utilizar Kali Linux. Esta es una distribución de Linux diseñada para tareas de auditoría y seguridad informática. Cuenta con multitud de herramientas, pero en nuestro caso nos centraremos en Wireshark y en Ettercap.

Por último, vamos a comentar Ettercap. Es una herramienta usada para realizar ataques MITM, auditoría y análisis de protocolos. Es capaz de interceptar tráfico, capturar contraseñas y *eavesdropping.*

5.3. Funcionamiento del protocolo ARP

En este apartado vamos a hacer pruebas para mostrar cómo funciona normalmente el protocolo ARP. Para ello, vamos a ejecutar Wireshark en nuestra máquina y a realizar un ping a un smartphone de nuestra red. A continuación, filtraremos los paquetes en Wireshark para ver los que forman parte del protocolo ARP.

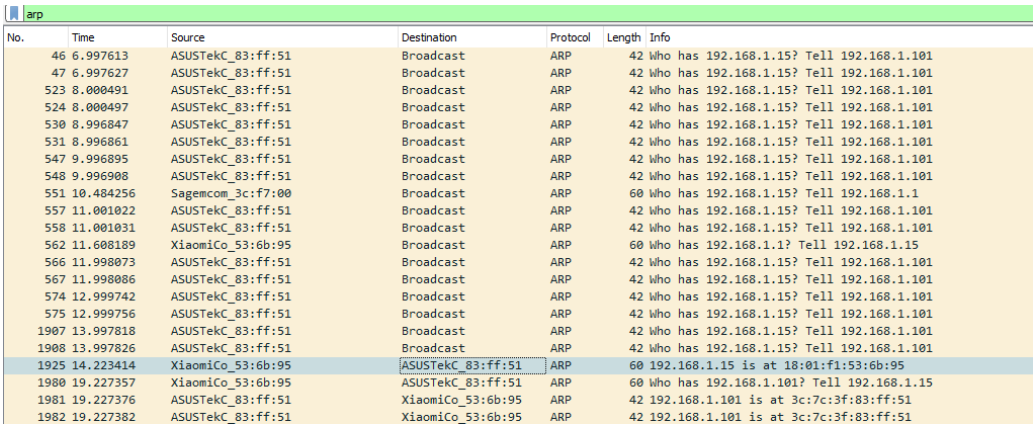
```
Microsoft Windows [Version 10.0.19043.1165]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>ping 192.168.1.15

Pinging 192.168.1.15 with 32 bytes of data:
Reply from 192.168.1.15: bytes=32 time=193ms TTL=64
Reply from 192.168.1.15: bytes=32 time=121ms TTL=64
Reply from 192.168.1.15: bytes=32 time=688ms TTL=64
Reply from 192.168.1.15: bytes=32 time=258ms TTL=64

Ping statistics for 192.168.1.15:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 121ms, Maximum = 688ms, Average = 315ms
```

Figura 5.3.1: Ping a nuestro *smartphone*.



arp

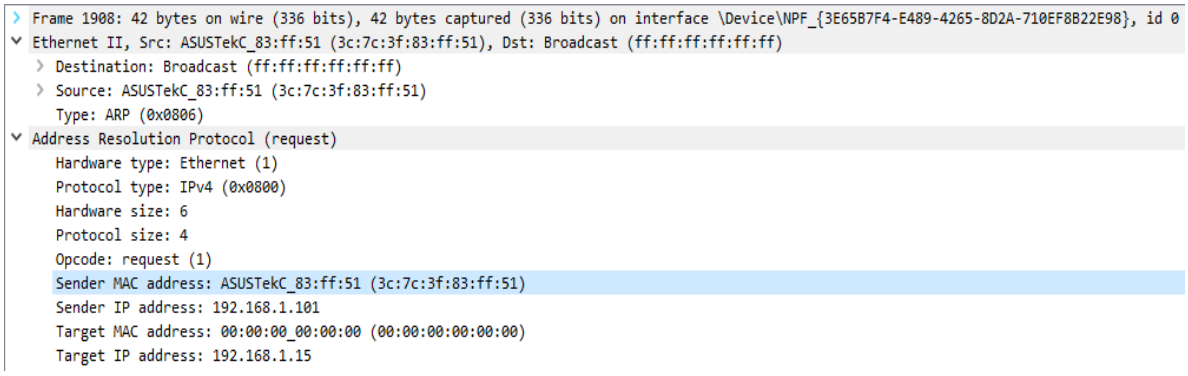
No.	Time	Source	Destination	Protocol	Length	Info
46	6.997613	ASUSTekC_83:ff:51	Broadcast	ARP	42	Who has 192.168.1.15? Tell 192.168.1.101
47	6.997627	ASUSTekC_83:ff:51	Broadcast	ARP	42	Who has 192.168.1.15? Tell 192.168.1.101
523	8.000491	ASUSTekC_83:ff:51	Broadcast	ARP	42	Who has 192.168.1.15? Tell 192.168.1.101
524	8.000497	ASUSTekC_83:ff:51	Broadcast	ARP	42	Who has 192.168.1.15? Tell 192.168.1.101
530	8.996847	ASUSTekC_83:ff:51	Broadcast	ARP	42	Who has 192.168.1.15? Tell 192.168.1.101
531	8.996861	ASUSTekC_83:ff:51	Broadcast	ARP	42	Who has 192.168.1.15? Tell 192.168.1.101
547	9.996895	ASUSTekC_83:ff:51	Broadcast	ARP	42	Who has 192.168.1.15? Tell 192.168.1.101
548	9.996908	ASUSTekC_83:ff:51	Broadcast	ARP	42	Who has 192.168.1.15? Tell 192.168.1.101
551	10.484256	Sagemcom_3c:f7:00	Broadcast	ARP	60	Who has 192.168.1.15? Tell 192.168.1.1
557	11.001022	ASUSTekC_83:ff:51	Broadcast	ARP	42	Who has 192.168.1.15? Tell 192.168.1.101
558	11.001031	ASUSTekC_83:ff:51	Broadcast	ARP	42	Who has 192.168.1.15? Tell 192.168.1.101
562	11.608189	XiaomiCo_53:6b:95	Broadcast	ARP	60	Who has 192.168.1.1? Tell 192.168.1.15
566	11.998073	ASUSTekC_83:ff:51	Broadcast	ARP	42	Who has 192.168.1.15? Tell 192.168.1.101
567	11.998086	ASUSTekC_83:ff:51	Broadcast	ARP	42	Who has 192.168.1.15? Tell 192.168.1.101
574	12.999742	ASUSTekC_83:ff:51	Broadcast	ARP	42	Who has 192.168.1.15? Tell 192.168.1.101
575	12.999756	ASUSTekC_83:ff:51	Broadcast	ARP	42	Who has 192.168.1.15? Tell 192.168.1.101
1907	13.997818	ASUSTekC_83:ff:51	Broadcast	ARP	42	Who has 192.168.1.15? Tell 192.168.1.101
1908	13.997826	ASUSTekC_83:ff:51	Broadcast	ARP	42	Who has 192.168.1.15? Tell 192.168.1.101
1925	14.223414	XiaomiCo_53:6b:95	ASUSTekC_83:ff:51	ARP	60	192.168.1.15 is at 18:01:f1:53:6b:95
1980	19.227357	XiaomiCo_53:6b:95	ASUSTekC_83:ff:51	ARP	60	Who has 192.168.1.101? Tell 192.168.1.15
1981	19.227376	ASUSTekC_83:ff:51	XiaomiCo_53:6b:95	ARP	42	192.168.1.101 is at 3c:7c:3f:83:ff:51
1982	19.227382	ASUSTekC_83:ff:51	XiaomiCo_53:6b:95	ARP	42	192.168.1.101 is at 3c:7c:3f:83:ff:51

Figura 5.3.2: Captura de Wireshark filtrando por ARP.

Observamos cómo se envían muchos paquetes desde nuestro ordenador preguntando quién tiene la dirección IP del smartphone, mediante un *broadcast.*

Paquete 1908: mandamos el mensaje ARP *request*. Nuestro ordenador envía un broadcast a toda la red para averiguar qué dispositivo corresponde a la IP solicitada (192.168.1.15). Obtenemos los siguientes datos:

- La dirección MAC del emisor se corresponde a nuestro adaptador de red: 3C:7C:3F:83:FF:51.
- La dirección IP del emisor es la de nuestro ordenador físico: 192.168.1.101.
- La dirección MAC de nuestro teléfono inteligente aún es desconocida, por esta razón aparece como: 00:00:00:00:00:00.



```
> Frame 1908: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface \Device\NPF_{3E65B7F4-E489-4265-8D2A-710EF8B22E98}, id 0
v Ethernet II, Src: ASUSTekC_83:ff:51 (3c:7c:3f:83:ff:51), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
  > Destination: Broadcast (ff:ff:ff:ff:ff:ff)
  > Source: ASUSTekC_83:ff:51 (3c:7c:3f:83:ff:51)
    Type: ARP (0x0806)
v Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: ASUSTekC_83:ff:51 (3c:7c:3f:83:ff:51)
    Sender IP address: 192.168.1.101
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 192.168.1.15
```

Figura 5.3.3: Nuestro ordenador envía un broadcast para averiguar la MAC de nuestro smartphone. ARP *request*.

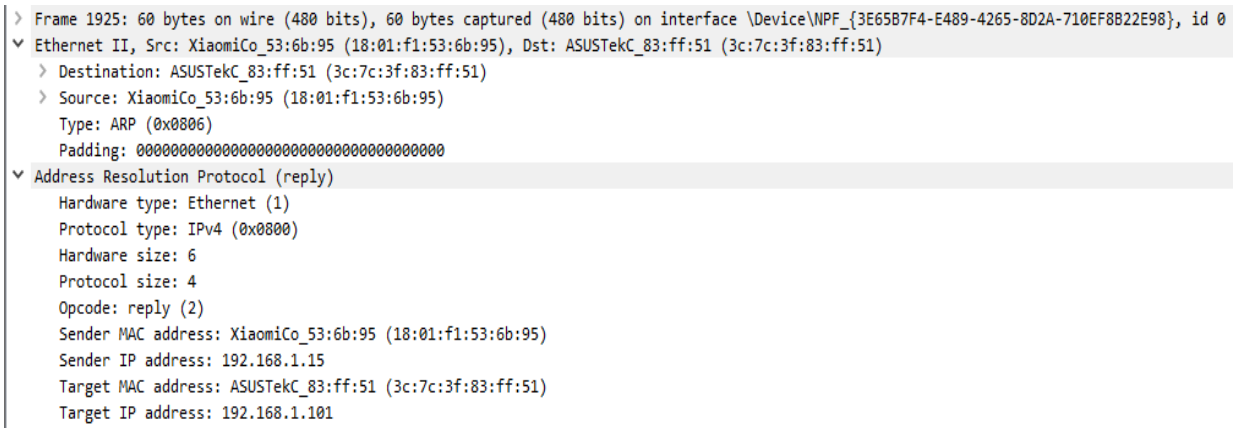


Figura 5.3.4: ARP *reply*. Nuestro *smartphone* contesta con su dirección IP.

Paquete 1925: Recibimos el ARP *reply*: En este paquete nuestro *smartphone* contesta a nuestro ordenador con su dirección MAC. La MAC del emisor se corresponde a nuestro *smartphone*: 18:01:F1:53:6B:95. La IP del emisor es la del teléfono inteligente. 192.168.1.15. Las direcciones MAC e IP objetivo del mensaje son las de nuestro ordenador.

Paquete 1980: Se repite el proceso, esta vez es nuestro *smartphone* el que intenta localizar la dirección MAC de nuestro ordenador a través de un mensaje ARP *request*.

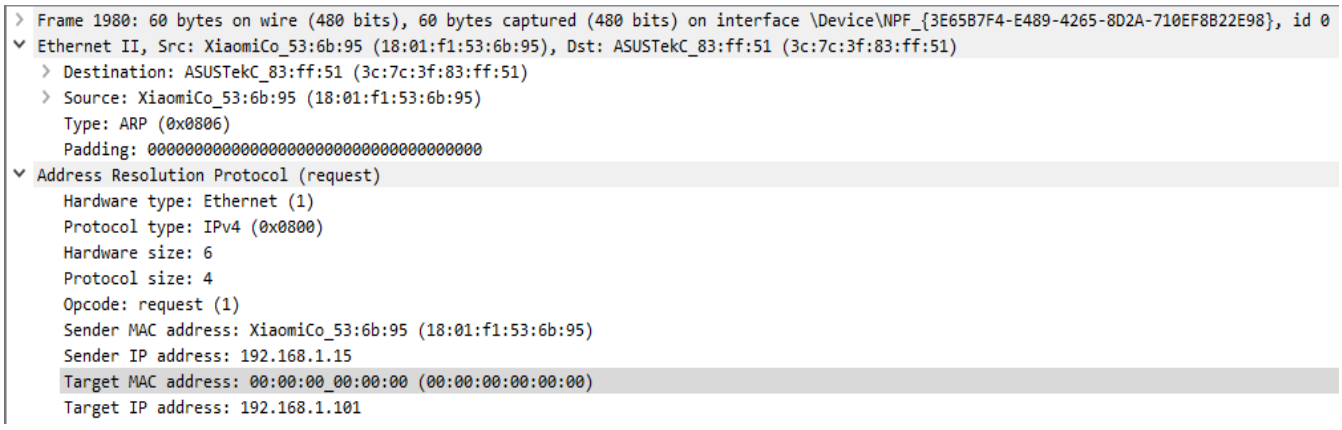


Figura 5.3.5: Nuestro smartphone pide la MAC a nuestro ordenador.

Paquete 1981: Acaba el proceso, nuestro ordenador contesta a nuestro teléfono inteligente con su dirección MAC.

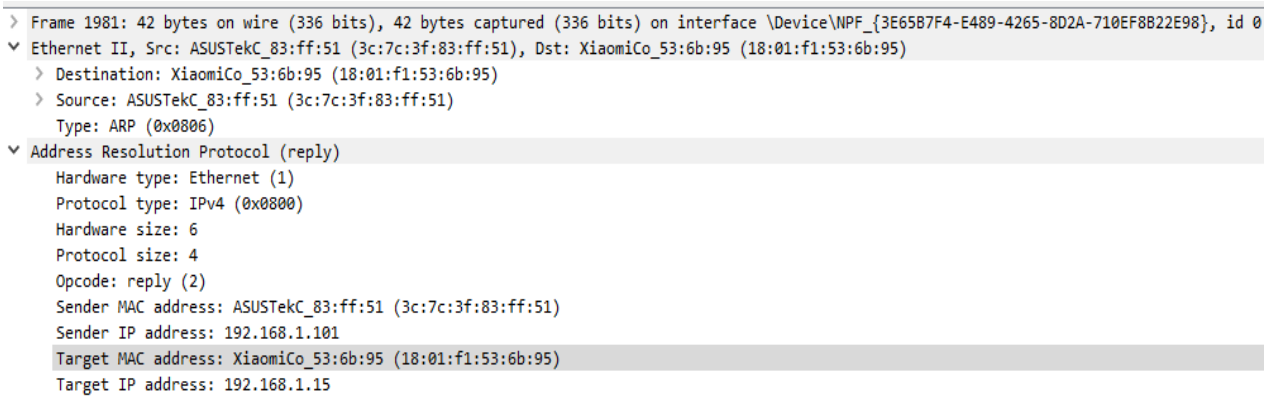


Figura 5.3.6: Nuestro ordenador contesta al smartphone con su MAC.

A continuación, mostramos la tabla ARP de nuestro ordenador:

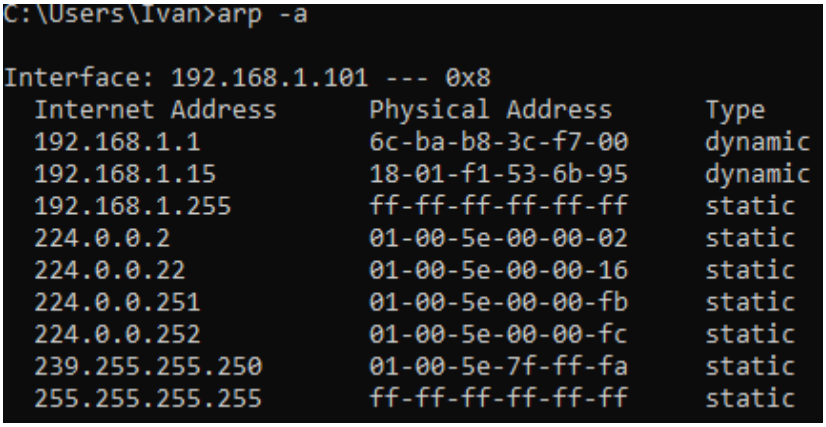


Figura 5.3.7: Tabla ARP de nuestro ordenador.

En la tabla ARP vienen recogidas todas las entradas de cada una de las interfaces del ordenador. Podemos ver cada dispositivo con su dirección física y su IP correspondiente.

5.3. Funcionamiento del protocolo HTTP

Para estudiar el funcionamiento del protocolo HTTP vamos a utilizar esta página: http://webscantest.com/login.php.Esta página no implementa ningún mecanismo de seguridad y resulta idónea para realizar comprobaciones.

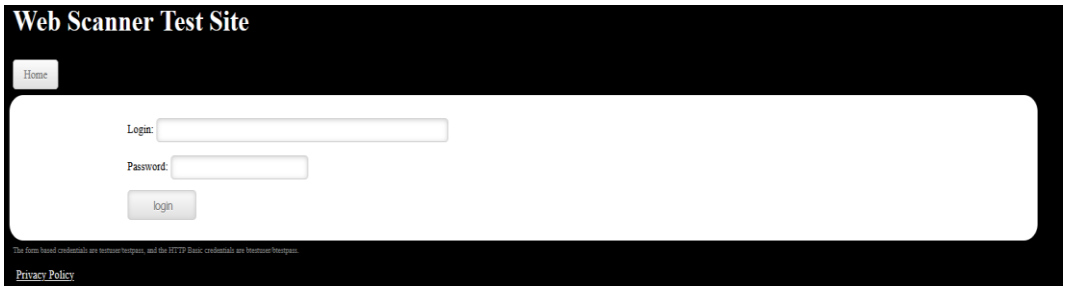


Figura 5.3.8: Formulario de inicio de sesión.

En primer lugar, abrimos Wireshark y lanzamos una captura, filtrando los paquetes del protocolo http y seleccionando la interfaz eth. Después, introducimos “hola” en la casilla de “Login” y “1234” en el campo “Password” en el formulario y pulsamos el botón de login para iniciar sesión.

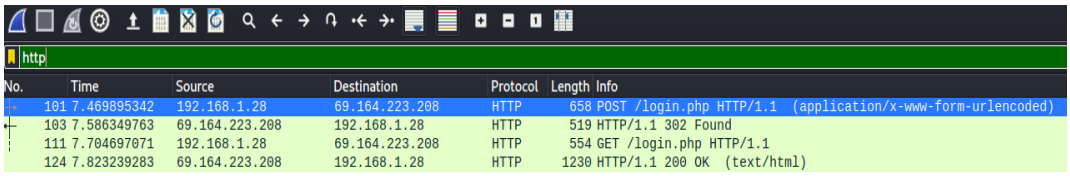


Figura 5.3.9: Captura de Wireshark filtrando por HTTP.

En esta captura, seleccionamos el paquete de datos que comprende el método POST. Es el que ha enviado la información del inicio de sesión y el que nos permitirá indagar de qué manera han sido guardados los datos.

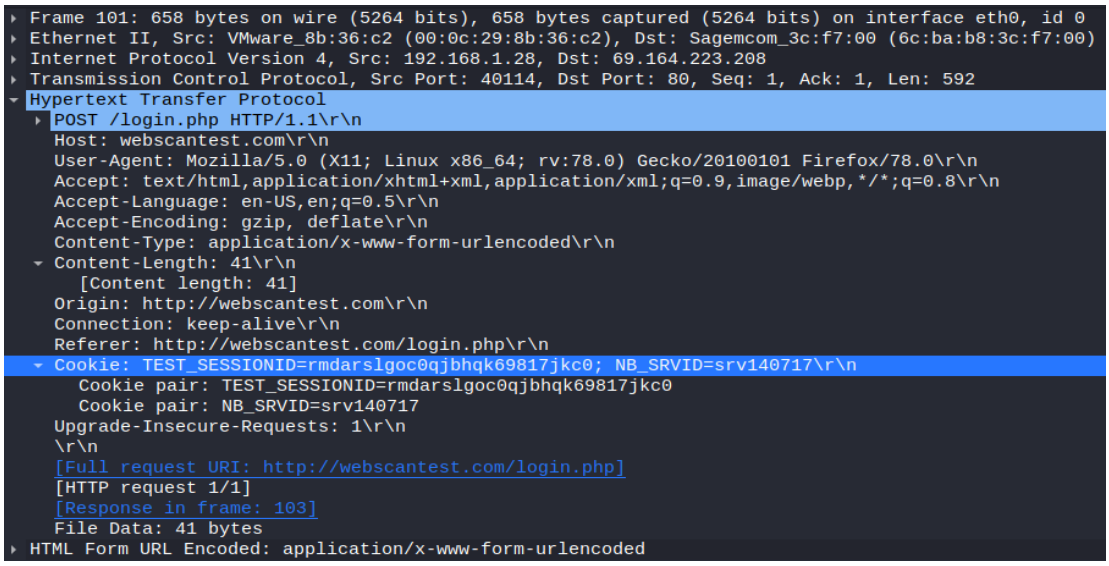


Figure 5.3.10: Captura de Wireshark con la cabecera POST.

En esta figura podemos ver el paquete desglosado, apreciamos las cabeceras del protocolo HTTP. Nos centramos en la cookie, en ella es donde estarán guardados los datos que introdujimos en el formulario de la página.

Wireshark incluye la opción de *Follow HTTP Stream*. Esta opción nos muestra la secuencia de paquetes tal y como ha viajado por la red, de la misma manera que lo percibe la capa de aplicación.

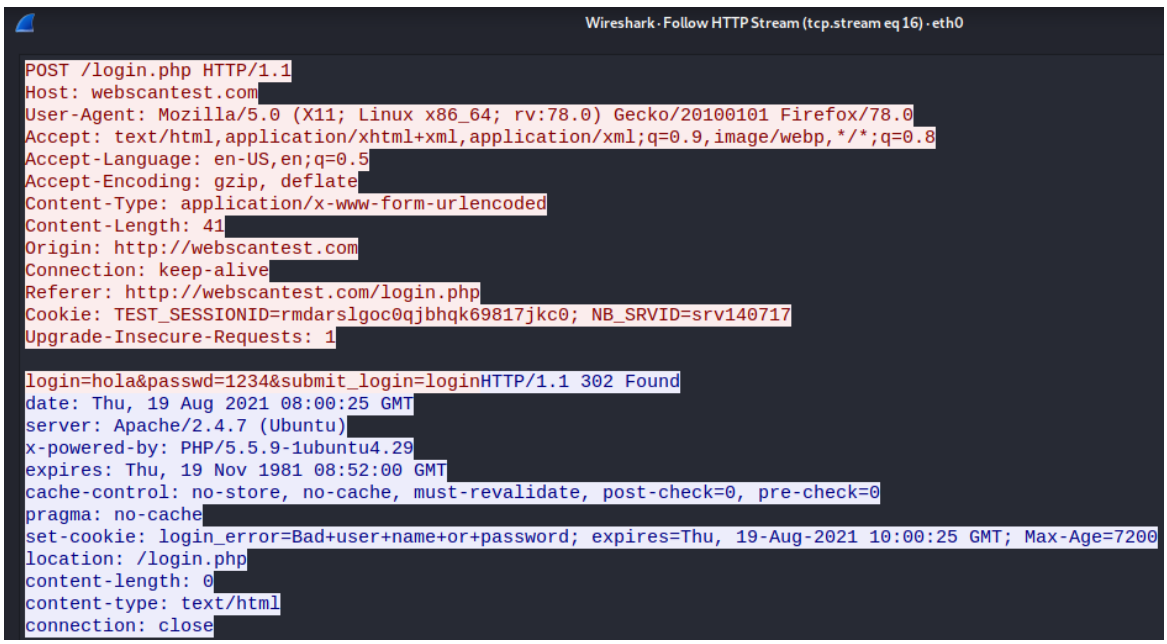


Figura 5.3.11: Captura de Wireshark del tráfico de TCP.

Esta opción nos permite ver la comunicación entre el cliente y el servidor mediante el protocolo HTTP en el orden en el que ha ocurrido. Primero, en letras rojas se nos muestra la comunicación por la parte del cliente, el método usado, el *host* al que se quiere conectar, nuestra versión de navegador, el sistema operativo, las cabeceras *accept*, el tipo de contenido y su longitud…

Al final se nos muestra la información de inicio de sesión y los datos que hemos introducido como contraseña. Después en letras azules tenemos la respuesta del servidor, con sus respectivas cabeceras.

5.4. HTTPS

En este apartado vamos a conectarnos a una página web que implementa https y desglosarla mediante Wireshark. Lanzamos Wireshark desde la terminal con la orden: “Wireshark &” y filtramos por TLS. Acto seguido, nos conectamos a la página: chess.com.

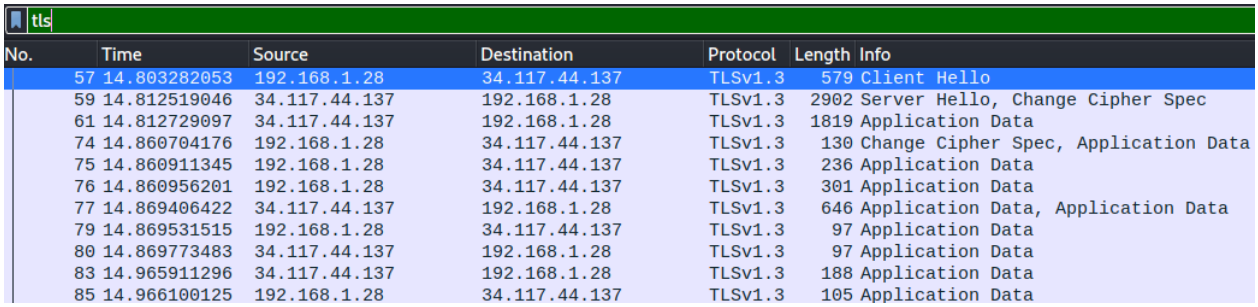


Figura 5.4.1: Captura de Wireshark del tráfico protocolo TLS.

Observamos el intercambio de mensajes entre el servidor de *chess.com* y nuestra máquina virtual.

El primer mensaje es un *Client Hello*, que mandamos nosotros al servidor. Podemos ver que estamos utilizando el TLSv1.3 y en la parte de TCP observamos los puertos que intervienen, el 51130 por nuestra parte y el 443 del servidor que implemente TLS.

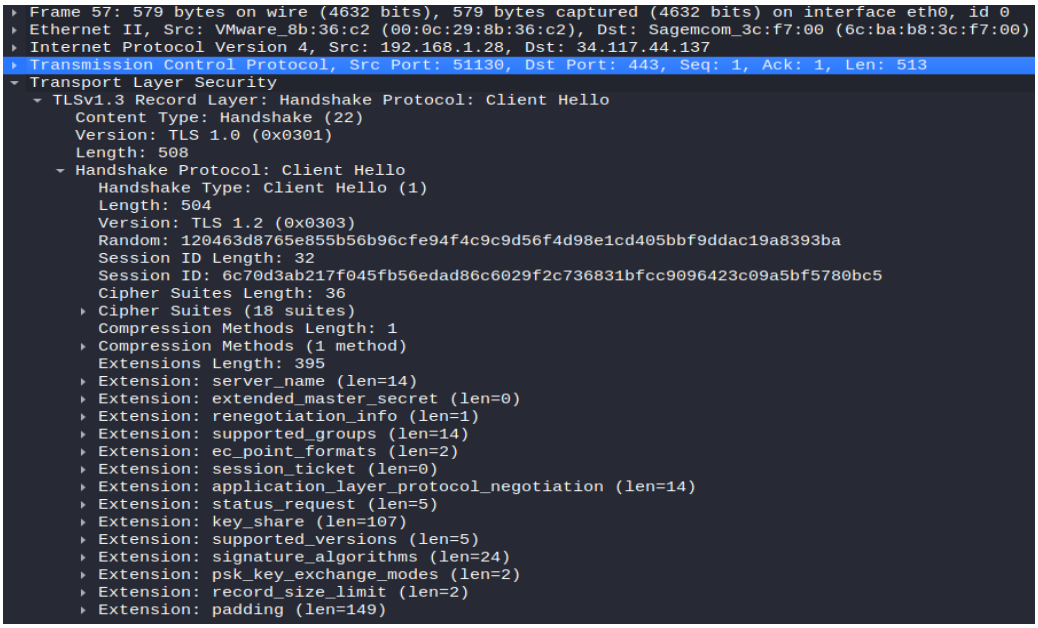


Figura 5.4.2: Captura de Wireshark con el mensaje *Client Hello*.

Apreciamos todo lo que mandamos al servidor en el *Client Hello*.

Los métodos de cifrado y compresión que propone:

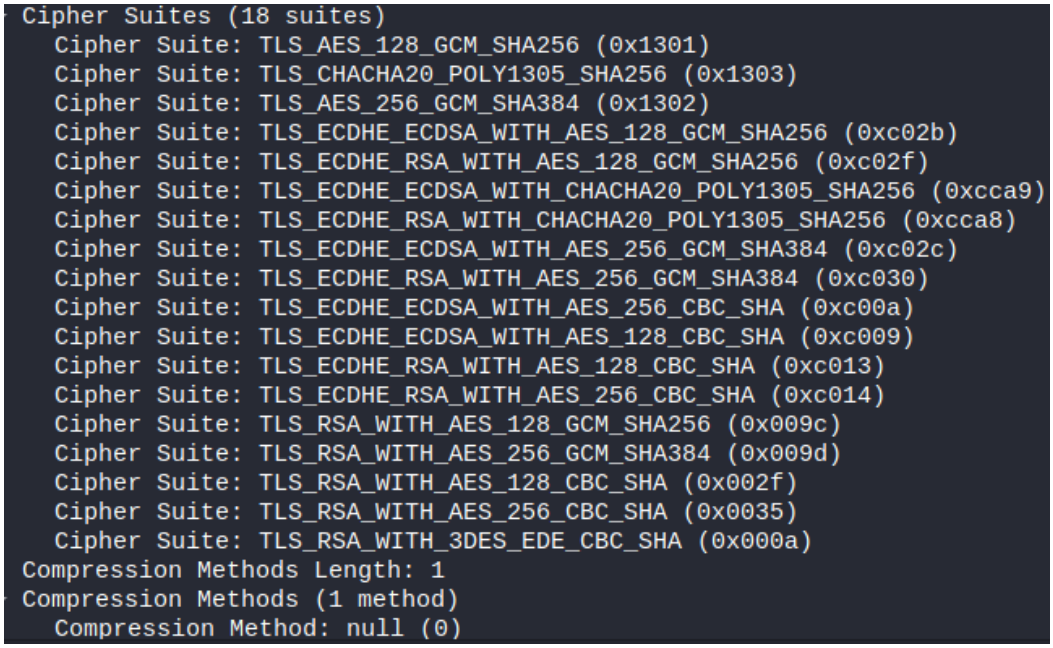


Figura 5.4.3: Captura de Wireshark con los métodos de encriptación y compresión que el cliente manda al servidor.

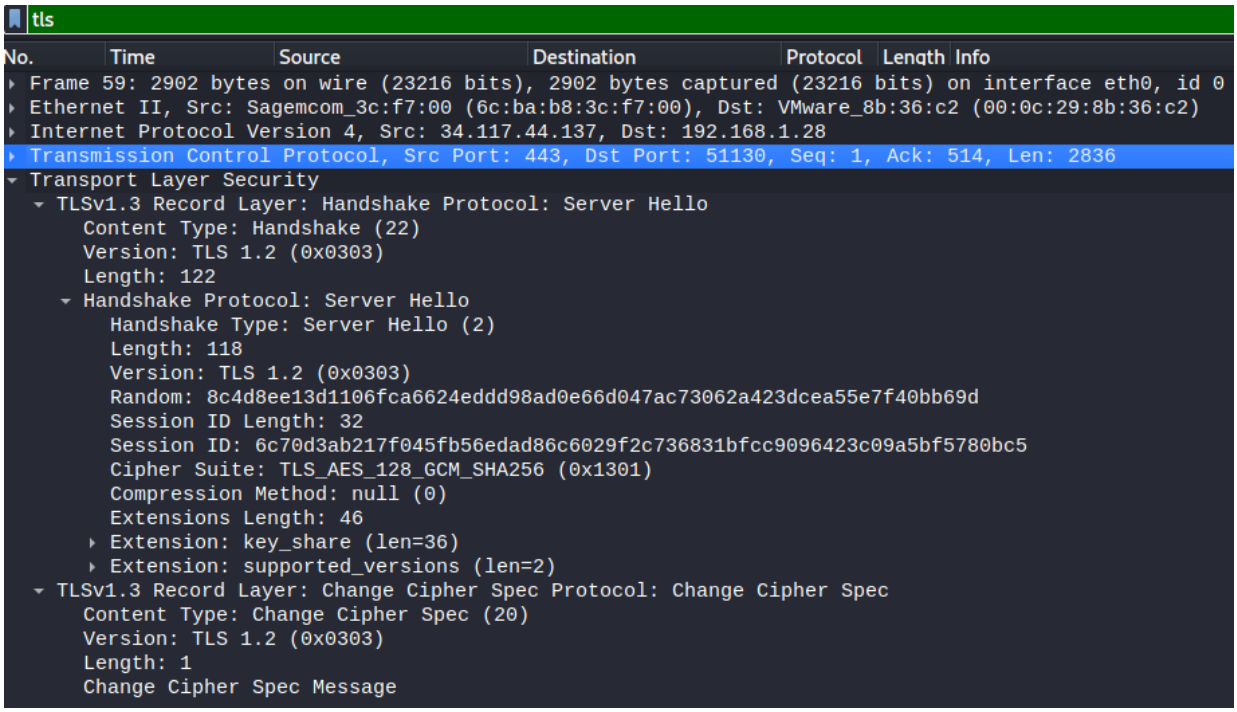


Figure 5.4.4: Captura de Wireshark del mensaje *Server Hello* del servidor.

En esta captura vemos la respuesta del servidor, *el Server Hello*.

Se ha acordado usar la versión TLS 1.2, una ID de sesión, el método de encriptación: TLS_AES_128_GCM_SHA256 (TLS se refiere al protocolo TLS, AES es el algoritmo para la clave simétrica de 128 bits, GCM es el modo de operación del algoritmo y SHA256 se refiere al algoritmo *Secure Hash Algorithm* de 256 bits).

Si intentamos leer los datos de la conexión nos encontramos con que los datos han sido encriptados.

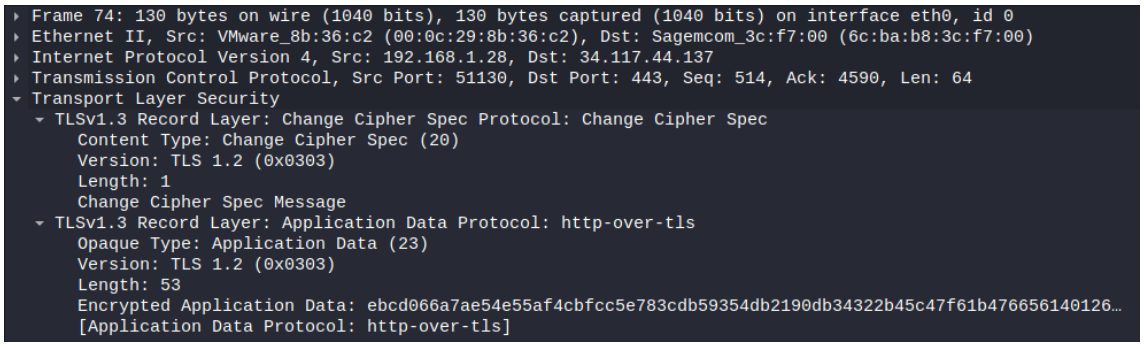


Figura 5.4.5: Captura de Wireshark de los mensajes intercambiados entre nuestro ordenador y el servidor de chess.com.

Intentando seguir el flujo TCP:

Figura 5.4.6: Captura de Wireshark de los datos encriptados

Solo podemos apreciar el nombre del servidor (chess.com), el resto de los datos han sido encriptados.

5.5. ARP *spoofing*

En este apartado vamos a tratar de explotar la vulnerabilidad del protcolo ARP previamente mencionada a través de un ataque MITM usando el programa Ettercap desde nuestra máquina virtual con Kali Linux.

Partimos de la siguiente configuración de red:

Nuestro ordenador está conectado al *router* a través de un cable Ethernet. En nuestro ordenador ejecutamos una máquina virtual con el sistema operativo Kali Linux y con el adaptador de red en modo *bridged*.

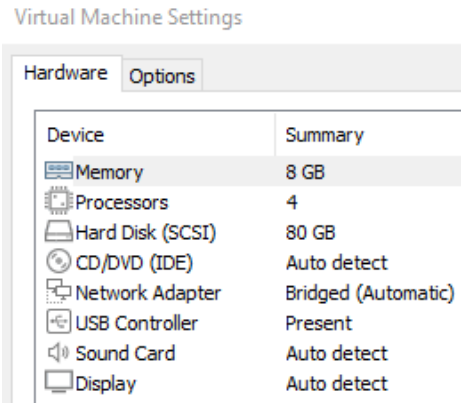


Figura 5.5.1: Configuración de nuestra máquina virtual.

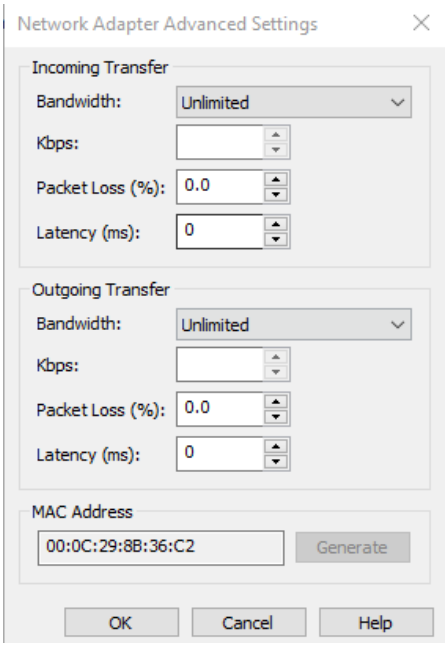


Figura 5.5.2: Dirección MAC de nuestra máquina virtual.

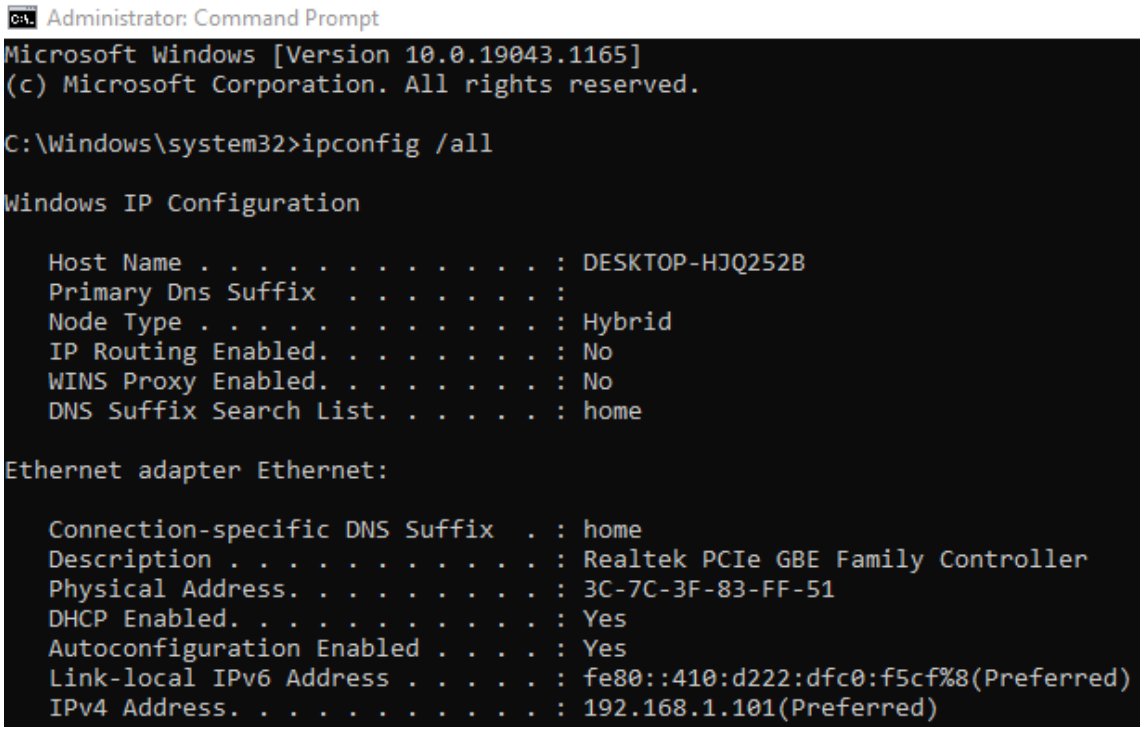


Figura 5.5.3: Configuración de red de nuestro ordenador tras ejecutar ipconfig /all.

Antes de empezar con el ataque mostramos la configuración de red de nuestro PC mediante la consola de Windows. Apreciamos que la dirección MAC aparece como la *Physical Address* de nuestro adaptador de Ethernet y es: 3C:7C:3F:83:FF:51, mientras que nuestra dirección IP aparece en el campo de *IPv4 Address* y es: 192.168.1.101.

A continuación, realizamos lo mismo con el teléfono inteligente, buscamos en ajustes/acerca del teléfono y obtenemos la dirección física 18:01:F1:53:6B:95 y la dirección IP: 192.168.1.15.

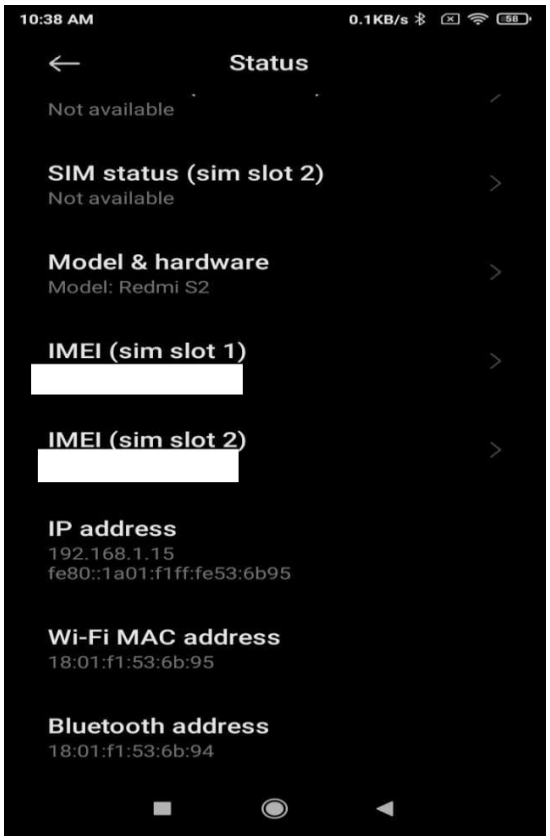


Figura 5.5.4: Captura de las direcciones IP y MAC de nuestro *smartphone*.

Ahora realizaremos un ataque MITM entre el smartphone y el *router* mediante Ettercap.

Ettercap nos proporciona una interfaz gráfica sencilla para realizar el ataque. El primer paso será de buscar objetivos, mediante la opción de escanear los *hosts*.

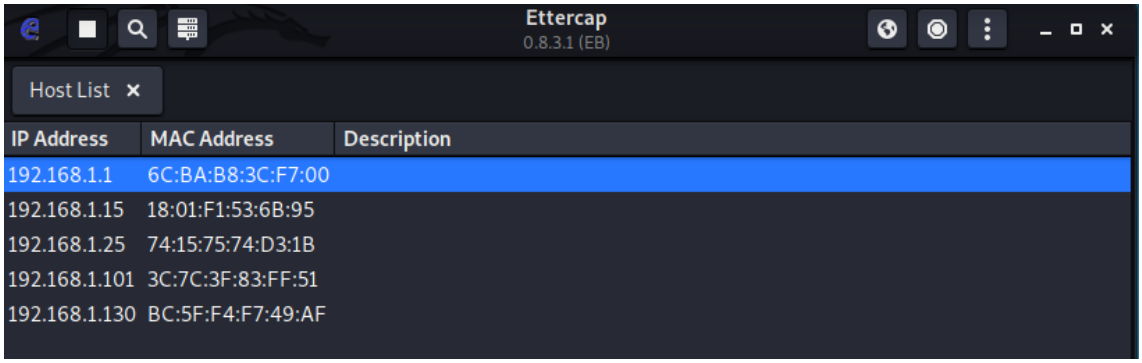


Figura 5.5.5: Lista de *hosts* que ha encontrado Ettercap.

A continuación, seleccionamos las IPs correspondientes en Ettercap como víctima uno y víctima dos y lanzamos el ataque.

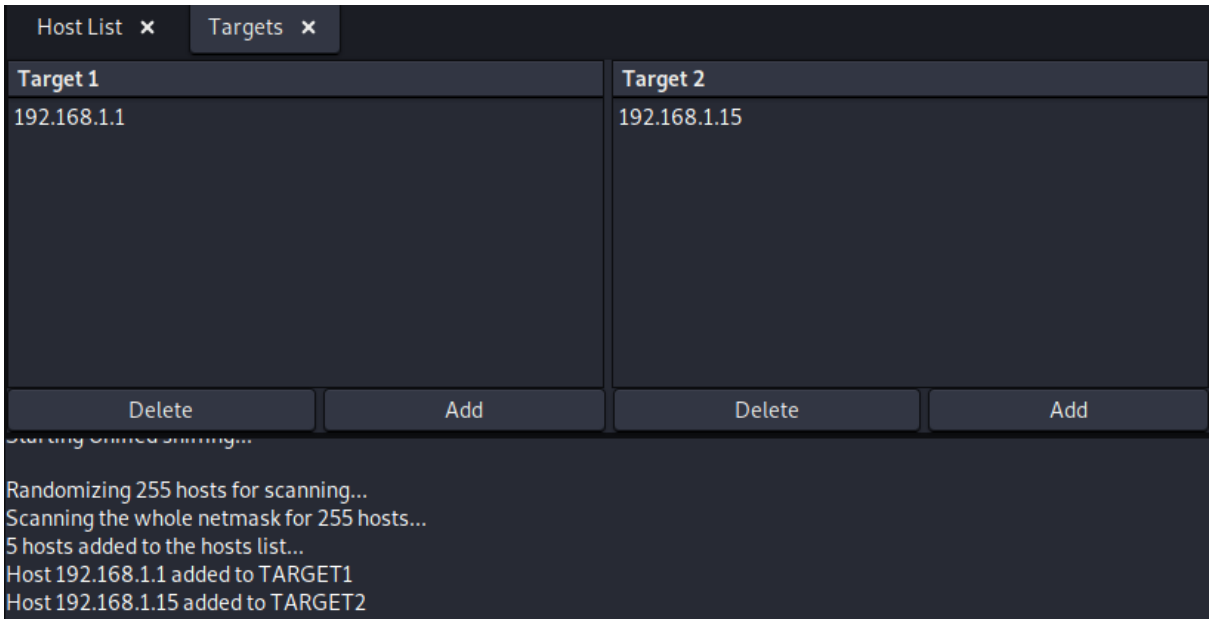


Figura 5.5.6: Seleccionamos las víctimas, el *router* y el *smartphone*.

Acto seguido, realizamos un ping desde nuestro ordenador (IP:192.168.1.101) a nuestro smartphone para ver qué sucede con Ettercap en funcionamiento.

Captura de “Wireshark” después de lanzar el ataque:

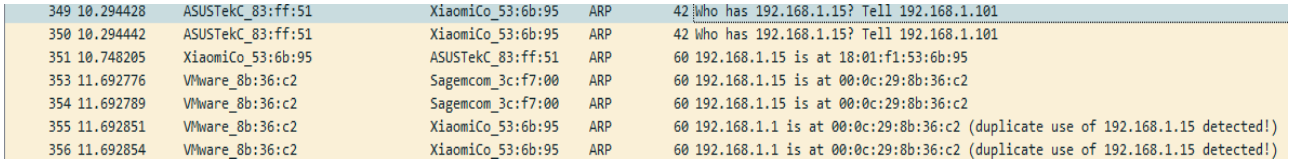


Figura 32: Captura de Wireshark tras lanzar el ataque MITM mediante Ettercap.

En esta captura se muestra cómo ahora hay dos dispositivos que nos contestan pretendiendo ser nuestro smartphone, en las tramas número 351, 353 y 354. En la trama 351, observamos:

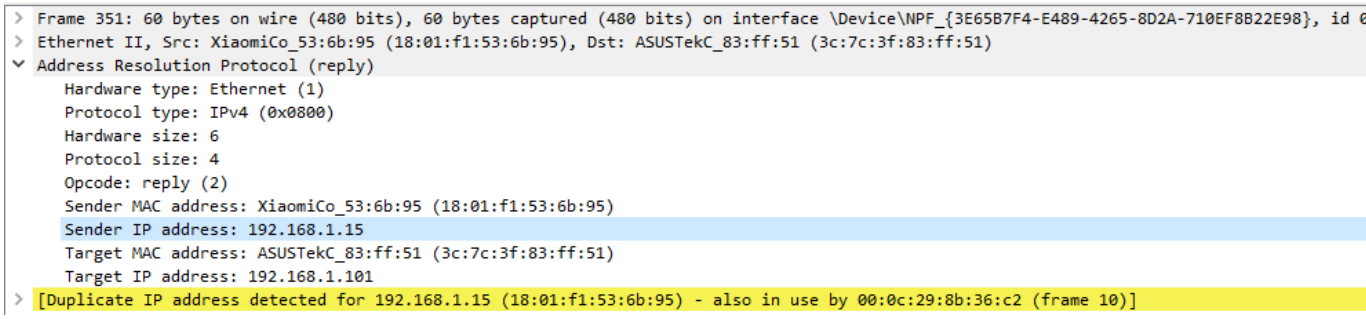


Figura 5.5.7: Captura de Wireshark de la trama 351.

En este caso, la captura es idéntica a la que obtuvimos en el caso anterior, con la misma drección MAC. Sin embargo, si nos fijamos, Wireshark ya está detectando que hay dos direcciones MAC asociadas a esta IP.

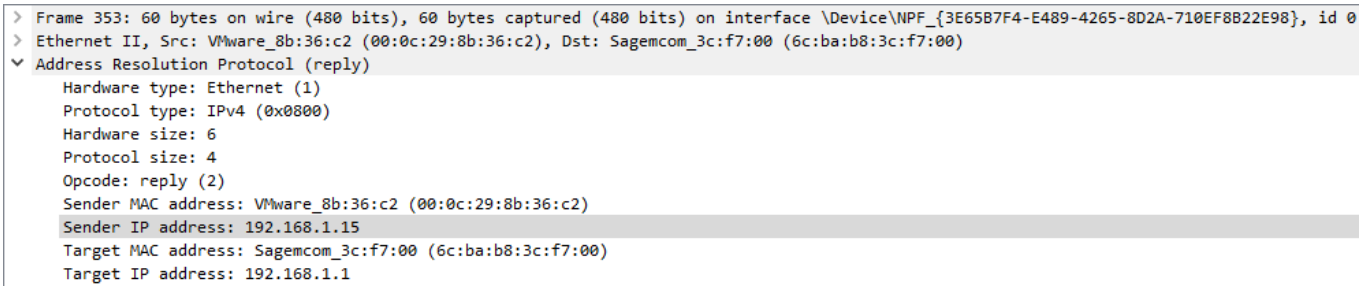


Figura 5.5.8: Captura de Wireshark de la trama 353.

Podemos ver cómo nuestra máquina virtual, con MAC: 00:0C:29:8B:36:C2 se está haciendo pasar por nuestro teléfono inteligente (pretende tener su IP) y cómo manda su MAC haciéndose pasar por el dispositivo.

En este momento cualquier mensaje que pase entre nuestro smartphone y nuestro *router* será también interceptado por un atacante, en nuestro caso, nuestra máquina virtual que ejecuta Kali Linux.

Si ejecutamos la orden arp -a podemos ver el efecto sobre las tablas arp:

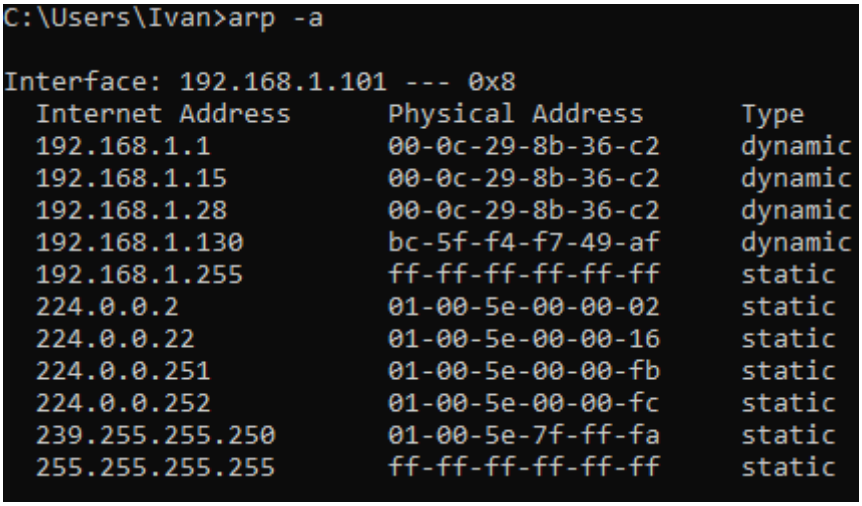


Figura 5.5.9. Tabla ARP tras el ataque MITM

Observamos como en este caso, tanto la dirección MAC del *router* como la del teléfono inteligente han sido suplantadas por la dirección física de nuestra máquina virtual. Podemos ver que han sido cambiadas dinámicamente en cuanto se ha realizado el ataque, la tabla se ha actualizado automáticamente.

6. Resultados

En este capítulo se exponen los resultados obtenidos al intentar explotar las vulnerabilidades mencionadas en capítulos anteriores, primero en el protocolo ARP y a continuación en el HTTP y su versión segura, HTPPS.

6.1. ARP

Hemos conseguido mostrar el funcionamiento del protocolo ARP y desglosar varias tramas mediante Wireshark para analizar el tráfico que se genera. A continuación, hemos realizado un ataque MITM a través de Ettercap y se ha mostrado a través de Wireshark el intercambio de mensajes entre las víctimas y el atacante y mediante las tablas arp que las direcciones MAC han sido modificadas.

6.2. HTTP y HTTPS

Mediante una serie de ejemplos sencillos y capturas de Wireshark hemos demostrado cuán inseguro es el protocolo HTTP por sí mismo y lo hemos comparado con el HTTPS, que al implementar TLS resulta casi imposible de vulnerar.

Se ha conseguido, usando una página que implementa el protocolo HTTP sin ningún otro tipo de seguridad adicional descubrir mediante Wireshark:

- Dirección IP y puerto de la página a la que queremos acceder.
- Nombre de dominio, sistema operativo del servidor, datos adicionales.
- Mensajes intercambiados.
    - Método GET para obtener la página. En las cabeceras podemos sacar información sobre el navegador que usa, el sistema operativo, la fecha, etc.
    - Método POST para iniciar sesión. Podemos acceder a la información suministrada por el usuario para crear una cuenta o iniciar sesión.

Tras un estudio del protocolo HTTPS y la realización de pruebas con Wireshark, no se ha podido obtener ninguna información más que la IP de la página a la que nos

conectamos, el puerto y el nombre de esta. Los demás datos han sido encriptados mediante una clave negociada entre el cliente y el servidor y, a pesar de que pueden ser capturados, no revelan información alguna.

7. Conclusiones

En el capítulo 3 describimos cómo funciona el protocolo ARP, en el hemos observado lo fácil que es implementar un ataque de tipo MITM mediante Ettercap.

Una alternativa para evitar este tipo de ataques sería editar manualmente la tabla arp y añadir los dispositivos de forma manual. Las nuevas entradas serían estáticas y un ataque de ARP *spoofing* no podría cambiarlas.

No es muy difícil observar las ventajas que presenta HTTPS frente a HTTP. Un atacante que se encuentra en nuestra red o en cualquier red no segura (wifi público, aeropuertos, cafeterías, gasolineras) puede ver todo el tráfico que enviamos y recibimos que pasa por el protocolo HTTP si está capturando datos.

HTTPS proporciona una alternativa mucho más fiable, basada en el protocolo criptográfico TLS, que crea un canal seguro de comunicación, incluso si la red es pública, nuestros datos viajarán cifrados.

Sin embargo, con el desarrollo de la computación cuántica, no está claro si los algoritmos que proporcionan seguridad al protocolo TLS resistirán al aumento de la potencia de cómputo. [17]

8. Trabajos Futuros

Este trabajo se ha centrado en el estudio de unos protocolos esenciales para el funcionamiento de las redes de computadoras. Sería interesante enfocar futuros estudios sobre otros protocolos, por ejemplo, intentar buscar vulnerabilidades en el protocolo TLS e intentar explotarlas, como ataques de reversión de versiones, ataques CRIME y BREACH, entre otros. También podría mostrarse un ataque de denegación de servicio a partir del ARP *spoofing***.**

Otro enfoque interesante a partir de este trabajo sería realizar análisis sobre ataques de ciberseguridad actuales, investigando las vulnerabilidades que los hacen posibles, incluyendo ejemplos prácticos y análisis de datos.

9. Bibliografía

[1] Nicolas Falliere, Liam O Murchu, Eric Chien. W32.Stuxnet DossierVersion 1.3 (November 2010).

[2] Karthik Selvaraj, Elia Florio, Andrea Lelli, and Tanmay Ganacharya. WannaCrypt ransomware worm targets out-of-date systems (May 12, 2017). https://www.microsoft.com/security/blog/2017/05/12/wannacrypt-ransomware-worm-targets-out-of-date-systems/?source=mmpc

[3] Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou. DDoS in the IoT: Mirai and Other Botnets (07 July 2017) pp 80-84.

[4] https://github.com/jgamblin/Mirai-Source-Code/tree/master/mirai/bot

[5] Saman Taghavi Zargar, Member, IEEE, James Joshi, Member, IEEE, and David Tipper, Senior Member, IEEE, A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks.

[6] https://www.cnet.com/news/nsa-disguised-itself-as-google-to-spy-say-reports/

[7] John E. Canavan, The Fundamentals of Network Security (2001).

[8] Paar, Christof; Pelzl, Jan; Preneel, Bart. Understanding Cryptography: A Textbook for Students and Practitioners (2010).

[9] James F. Kurose, Keith W Ross. Redes de Computadores. Un enfoque descendente. Capítulo 8: Seguridad.

[10] Chappell Laura A, Tittel Ed, Carrell Jeffrey L, Pyles James. Guide to TCP_IP 4th Edition (2013) pp 164-172.

[11] Robert Wagner. Address Resolution Protocol Spoofing and Man-in-the-Middle Attacks. September 27, 2001.

[12] James F. Kurose, Keith W Ross. Redes de Computadores. Un enfoque descendente. Capítulo 2. Capa de Aplicación.

[13] James F. Kurose, Keith W Ross. Redes de Computadores. Un enfoque descendente. Capítulo 3. Capa de Transporte.

[14] RFC 2246 https://datatracker.ietf.org/doc/html/rfc2246

[15] RFC 3546 https://datatracker.ietf.org/doc/html/rfc3546

[16] https://www.wireshark.org/#learnWS

[17] Daniel J. Bernstein. Introduction to post-quantum cryptography. http://www.pqcrypto.org/www.springer.com/cda/content/document/cda_downloaddocument/9783540887010-c1.pdf