Software de Comunicacións

Práctica 2 - Parte 2: Beaconing e Acceso ao Medio con IEEE 802.11

Obxectivo

Comprender e analizar diversos aspectos básicos do funcionamento das capas física, enlace e rede do estándar IEEE 802.11.

Enunciado

As redes wifi actuais están baseadas nos estándares IEEE 802.11 (nas súas variantes a/b/g/n/ac). Estas redes, aínda que difiren en distintos aspectos da capa física, son moi similares a nivel de capa de enlace e rede. Nesta práctica, imos a despregar un entorno que nos permitirá capturar tramas IEEE 802.11 destas dúas últimas capas. Gracias a este entorno seremos capaces de obter numerosa información sobre as redes wifi que nos rodean e poderemos analizar os tempos e mecanismos de acceso ao medio proporcionados por cada estándar.

Entorno de Probas

Se intentamos capturar paquetes con Wireshark a través dunha tarxeta de rede sen fíos, IEEE 802.11, coas opcións por defecto, tan só recibiremos paquetes despois de conectarnos con éxito a unha red wifi. Dado que normalmente o AP (*Access Point*, Punto de Acceso) concede IPs dinámicamente, o primeiro paquete que capturará Wireshark será a petición de nova IP por DHCP. Polo tanto, non capturaremos ningunha das mensaxes que nos interesan nesta práctica: as asociadas co acceso ao medio e coa negociación inicial entre a nosa tarxeta wifi e o AP.

O motivo polo que Wireshark non mostra os paquetes ven detallado na súa Wiki[1]: coa configuración por defecto, só os paquetes que envíe/reciba o usuario serán capturados. Para poder capturar o resto de paquetes, hai que poñer a tarxeta wifi en **modo monitor**. Se temos acceso a un sistema Linux, poñer a tarxeta en modo monitor é moi sinxelo: tan só precisamos instalar un software como *airmon* (nas súas versións ng ou zc), o cal está incluído na suite *Aircrack*. Se non queremos alterar a nosa instalación Linux, a solución más fácil é recorrer a algunha das múltiples distribucións *Live* que xa traen instalado tanto Wireshark como airmon (a continuación explicarase como usar estas ferramentas coa distribución *Kali*).

En Windows, pasar a tarxeta a modo monitor é máis complicado, dado que actualmente WinPcap, a librería de *sniffing* máis usada, non o soporta por defecto. Tan só algúns adaptadores wifi deseñados ad-hoc (e.g. *AirPcap* de Riverbed Technology) permiten a conmutación directa a modo monitor. Polo tanto, se non posuímos o hardware adecuado, o máis fácil é recorrer a unha instalación Linux (por exemplo *Kali*).

Existe tamén a posibilidade de crear un *Live CD* ou un pendrive USB arrancable con algunha das numerosas distribucións Linux dispoñibles, pero se somos usuarios de

[1] Captura de paquetes con Wireshark en redes WLAN: http://wiki.wireshark.org/CaptureSetup/WLAN

Windows, existe unha alternativa que nos vai permitir executar unha distribución Linux dentro do noso Windows a través da virtualización e coa axuda dun **adaptador wifi externo**. Estes serían os pasos a seguir:

1. Descargar VMWarePlayer[2]. Hai versións para Windows e Linux, tanto de 32 como de 64 bits. Ten unha restrición importante: o portátil/PC onde se instale debe de ter soporte para virtualización (isto verificarao o instalador de VMWarePlayer). En caso de non ter dito soporte, haberá que optar pola opción que se comentaba anteriormente, é dicir, crear un CD ou un pendrive arrancable coa distribución mencionada no seguinte punto (pódense atopar en Internet numerosos manuais sobre como crear USBs arrancables, pero, para evitar problemas, recoméndase usar *Universal USB Installer*[3] ou *Unetbootin*[4]).

2. Se se instalou sen problemas VMWarePlayer, podemos descargar a imaxe dunha distribución Linux que soporte Wireshark e airmon. Existen diversas distribucións con dito soporte, pero imos usar a que é unha das suites de seguridade e *pentesting* máis avanzadas: *Kali*. Na súa páxina de descargas aparece tamén un enlace que permite descargar directamente unha máquina virtual para VMWare[5] (hai tamén imaxes para VirtualBox e Hyper-V).

3. Iniciar VMWarePlayer, aceptar a licenza, facer click en *Open a Virtual Machine* e indicar onde se copiou a máquina virtual (hai que seleccionar o ficheiro .vmx extraído do .7z).

4. Pulsar sobre o botón de *Play* para arrancar a máquina virtual. Esto debería iniciar sen problemas a máquina virtual e deberíamos poder facer login coas credenciais **root/toor**.

5. Enchufar unha tarxeta wifi externa (i.e., USB), non vale coa interna do equipo, que será detectada coma interfaz Ethernet virtual. Esta tarxeta USB só pode ser usada por un SO á vez. Para usala con *Kali* en lugar de con Windows, no menú de VMWarePlayer, ir a ***Player->Removable Devices->Nuestra tarjeta inalámbrica->Connect***.

[2] URL de descarga de VMWarePlayer: https://my.vmware.com/web/vmware/free#desktop_end_user_computing/vmware_player/7_0|PLAYER-714|product_downloads

[3] Universal USB Installer: https://www.pendrivelinux.com

[4] Unetbootin: http://unetbootin.github.io

[5] Descarga de imagen de Kali para VMWare: https://www.offensive-security.com/kali-linux-vm-vmware-virtualbox-hyperv-image-download/

6. Para verificar que a tarxeta wifi se pode usar correctamente, dende un Terminal, escribir *iwconfig* e verificar que existe unha interface chamada "wlan0" ou similar.

Pola tanto, resumindo, temos diferentes opcións para preparar o entorno requirido para capturar os paquetes wifi que nos interesan:

1. Instalar Wireshark e airmon na nosa distribución de Linux.
2. Instalar *Kali* no noso disco duro ou crear un usb arrancable coa distribución de *Kali*.
3. Empregar un virtualizador e arrancar a máquina virtual con Kali que se pode descargar dende a páxina indicada anteriormente.

A partir de aquí, vamos asumir que estamos traballando en *Kali*, pero o prodemento sería equivalente no caso de optar pola primera opción. Unha vez dentro de Kali podemos acceder a Wireshark dende o terminal ou dende ***Applications->Sniffing & Spoofing***.

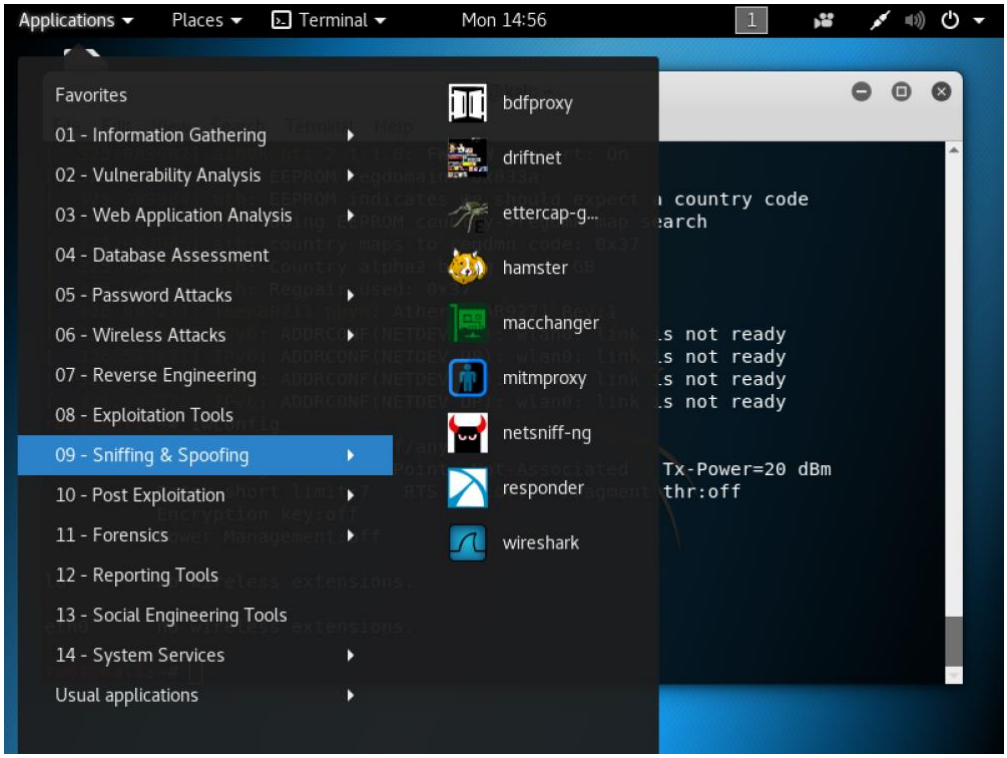

Figura 1: Acceso a Wireshark dende o menú Applications.

Poñendo a Tarxeta en Modo Monitor

Antes de comezar a capturar paquetes en modo monitor, é necesario comprender primeiro os catro modos en que pode funcionar unha tarxeta wifi:

- ***Modo Managed***. A tarxeta wifi conéctase a un AP local que lle da acceso á rede. Este é o modo máis usado, no que un cliente se conecta ao AP para conseguir acceso a Internet ou a unha rede local.

- ***Modo Ad-hoc***. Emprégase para comunicar directamente varias estacións entre si cando non hai dispoñible un AP.
- ***Modo Master.*** A tarxeta wifi compórtase coma un AP para proveer de conectividade a outras estacións. Obviamente, require do software adecuado para xestionar as conexións.
- ***Modo Monitor.*** A tarxeta deixa de transmitir e simplemente recibe todo o enviado pola canle na que estea configurada. A gran vantaxe deste modo é que a tarxeta envía ao SO (i.e. a Wireshark) os paquetes tal e como os recibe, sen ningún procesado adicional, o que nos permite realizar unha análise máis precisa das comunicacións.

No caso de usar Kali, podemos usar a suite Aircrack para poñer a tarxeta wifi en modo monitor. Para elo escribimos dende un terminal:

```
airmon-ng start wlan0
```

onde habería que substituír wlan0 pola interface correspondente. No caso de que a tarxeta quede colgada, pódese probar con *airmon-zc*. Con calquera destes dous comandos crearase unha interface virtual en *mon0* que **será a que deberemos seleccionar ao capturar con Wireshark**.

NOTA 1: se se fai uso das últimas versións de Kali, para certas distribucións Linux ou versión de OSX, pode ser necesario executar o seguinte comando antes de executar o comando airmon-ng para cambiar a modo monitor:

```
sudo airmon-ng check kill
```

Este comando é basicamente para deshabilitar o administrador de redes sen fíos e outros procesos que podan impedir cambiar a interface a modo Monitor.

NOTA 2: se nos conectamos a un AP (rede wifi), a tarxeta pasará a modo Managed: os datos que capture Wireshark virán como se procedesen dunha interface Ethernet, co cal non se poderán ver nin as tramas de control e administración 802.11, nin os paquetes destinados a outras redes wifi.

Un detalle importante a ter en conta é que, unha tarxeta wifi, nun determinado instante, só pode estar funcionando nunha canle (i.e., nunha banda de frecuencias). Aínda que depende do país/rexión, en 2,4 GHz existen ata 14 canles distintas. Se o que queremos é obter as tramas 802.11 dun AP concreto ao que teñamos acceso, o máis fácil é conectarse a dito AP, desconectarse e poñer a tarxeta en modo monitor; mentres non nos asociemos a outra rede, a tarxeta continuará monitorizando a canle do AP ao que nos asociemos. Existe tamén a posibilidade de monitorizar a canle/frecuencia que queiramos especificando os parámetros correspondentes en airmon.

Se estamos capturando paquetes con Wireshark, tamén é posible cambiar de canle dinamicamente. Obviamente, perderanse paquetes debido aos saltos de canle en canle, pero valeranos para facer un barrido e ver que hai en cada canle. Para realizar este cambio

de canle de maneira manual, sen necesidade de reinicializar a tarxeta (ou conectarnos a outra canle), podemos escribir dende un terminal:

iwconfig mon0 channel X

onde *mon0* é o nome da interface e *X* un enteiro entre 1 y 13 que indica o número de canle a monitorizar. Se se quere realizar o salto de canle de maneira automatizada, pódese usar o *script* da Figura 2 (é necesario cambiar o nome da interface polo correspondente, gardar o código nun ficheiro .sh e darlle permisos de execución (755) con chmod).

```
!/bin/bash
IFACE=mon0
IEEE80211bg="1 2 3 4 5 6 7 8 9 10 11"
IEEE80211bg_intl="$IEEE80211b 12 13 14"
IEEE80211a="36 40 44 48 52 56 60 64 149 153 157 161"
IEEE80211bga="$IEEE80211bg $IEEE80211a"
IEEE80211bga_intl="$IEEE80211bg_intl $IEEE80211a"

while true ; do
        for CHAN in $IEEE80211bg ; do
                echo "Cambiando a canle $CHAN"
                iwconfig $IFACE channel $CHAN
                sleep 1
        done
done
```

Figura 2: Script para automatizar o salto de canle

IEEE 802.11: Interferencias e Colisións

Antes de comezar a realizar as diferentes probas con tramas IEEE 802.11, convén recordar que, ao contrario que ocorre en Ethernet, onde as tarxetas poden transmitir e monitorizar a rede simultaneamente, as tarxetas wifi actuais xeralmente só poden transmitir ou recibir, non realizar ambas accións á vez. Isto dá lugar a que existan maiores posibilidades de que varias estacións transmitan á vez.

Os sistemas de prevención de colisións funcionan xeralmente ben, pero, aínda así, é común que existan colisións entre estacións transmisoras, con outras redes WLAN que transmitan na mesma canle ou con outros dispositivos que funcionan na mesma banda de frecuencia (e.g. teléfonos sen fíos, microondas, etc …). Cando dous dispositivos transmiten á vez, a transmisión corrómpese e o receptor dará por inválidos os paquetes. Os transmisores terán que esperar un tempo aleatorio e volverán transmitir.

Con todo isto podemos concluír que, ao capturar tráfico dunha rede wifi, ***non temos a certeza de que vaiamos capturar todo o tráfico***: algúns paquetes chegarán corrompidos, haberá retransmisións que fagan que se reciban múltiples copias dun mesmo paquete, etc…É máis, é importante ter en conta ao analizar unha captura que **non temos garantía de que os paquetes capturados se reciban na orde correcta**.

IEEE 802.11: Filtrado de Paquetes

Debido ás interferencias mencionadas, no momento en que poñamos Wireshark a capturar nunha canle, é moi probable que recibamos tráfico de diversas redes, tanto das que operan nunha mesma canle, como das de se atopan en canles adxacentes. Isto xa depende da sensibilidade da tarxeta wifi: por exemplo, unha tarxeta configurada para operar na canle 3 podería recibir paquetes das canles 2 e 4.

Por outro lado, o máis frecuente é que nos atopemos con redes IEEE 802.11 nas que un AP provee de conectividade a un ou máis clientes: isto é o que se denomina unha BSS (*Basic Service Set*). Cada AP dunha BSS identifícase mediante un BSSID (*BSS IDentifier*), o cal a miúdo é a propia MAC do AP. Tamén existe o identificador de rede (SSID, *Service Set IDentifier*, tamén denominado ESSID, *Extended SSID*), que é o identificador alfanumérico que os clientes ven cando se conectan a unha rede wifi. É importante diferenciar entre BSSID e ESSID: o primeiro identifica a un único AP, mentres que o segundo a unha rede (a cal pode constar de un ou máis APs).

Durante as análises que realizaremos a continuación, en circunstancias normais, será suficiente con filtrar os paquetes por MAC e/ou BSSID. Para filtrar por MAC podemos aplicar un filtro como:

```
wlan.sa == AA:BB:CC:DD:EE:FF
```

mentres que para filtrar por BSSID utilizaríamos o filtro:

```
wlan.bssid == AA:BB:CC:DD:EE:FF
```

1.1 Beacons: Intervalo de Beaconing

Un *beacon* é unha trama IEEE 802.11 que serve para que os APs anuncien a súa "existencia".

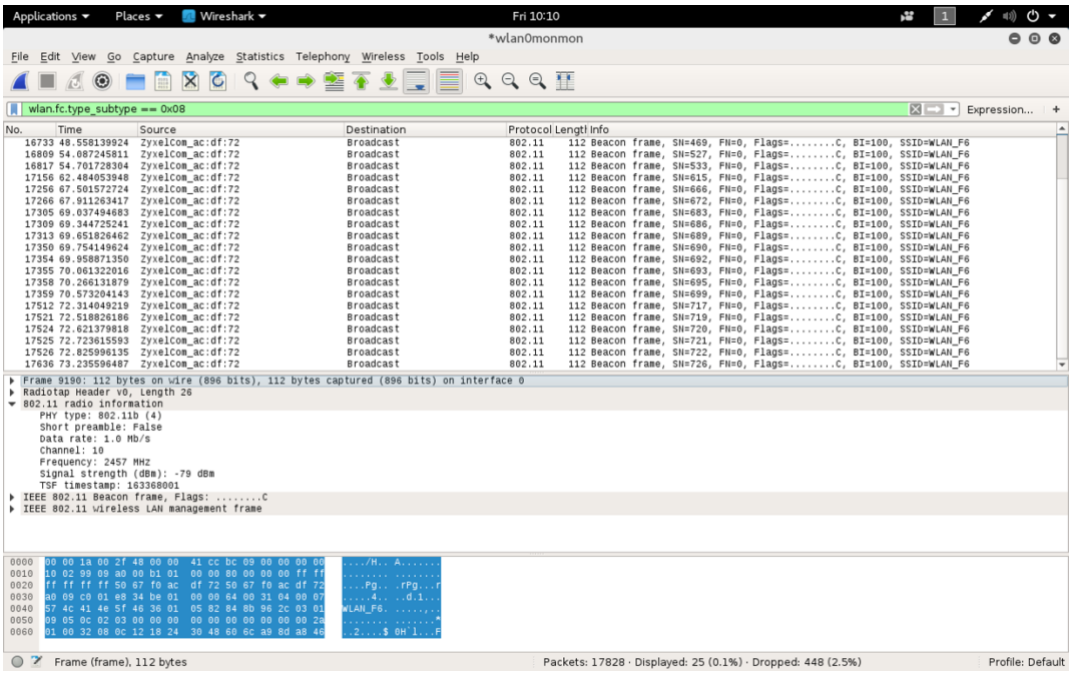

Figura 3: Exemplo de captura de beacons 802.11 con Wireshark.

A primeira proba que imos realizar consiste en medir o intervalo de *beaconing* (i.e. o intervalo de envío de *beacons*) dun punto de acceso determinado. Isto é: o tempo que transcorre entre o envío de dous *beacons* consecutivos. Dado que o adaptador wiFi vai recibir numerosos paquetes, o máis sinxelo é filtrar os paquetes capturados para ver só os *beacons*:

```
wlan.fc.type_subtype == 0x08
```

No caso de recibir varios *beacons* consecutivos **dun mesmo AP**, Wireshark facilítanos a tarefa mediante o cálculo do valor *Time Delta* que se atopa dentro da cabeceira Frame (ATENCIÓN: o contido da cabeceira *Frame* realmente NON se encontra dentro do paquete recibido, tan só é un resumo da información sobre o mesmo que engade artificialmente Wireshark).

Tras capturar polo menos 10 *beacons* dun mesmo AP, pídese:

a) Medir empiricamente o intervalo de *beaconing* do AP, indicando os valores mínimo, máximo e medio. Xustificar esta medición mediante capturas de pantalla dos paquetes mostrados en Wireshark.

b) Fíxate agora no contido do campo *IEEE 802.11 wireless LAN management frame* de cada un destes *beacons*. ¿Cal é intervalo de *beaconing* que ten estipulado o AP? ¿Existen diferencias respecto do tempo medido empiricamente? En caso afirmativo, ¿A que crees que se poden deber estas diferencias?

1.2 Beacons: Obtendo Datos sobre as Capas PHY y MAC

Trala cabeceira *Frame*, no caso dos paquetes con *beacons*, Wireshark mostra as cabeceiras *Radiotap* (engadida por Wireshark) e *IEEE 802.11 Radio Information*. Tras observar os campos destas cabeceiras, indicar:

a) ¿A que velocidade (rate) se transmiten os *beacons*? ¿Por que crees que se realiza a esta velocidade concreta?

b) ¿Que estándar (ou estándares) IEEE permite utilizar o AP? ¿En que banda de frecuencias se envían os paquetes? ¿A que canle corresponde esa banda?

c) ¿Cal é a dirección de destino dun *beacon*?

d) ¿Cal é a dirección MAC do AP? ¿E o BSSID?

e) Analiza os distintos campos relacionados con *flags* (i.e., *Present Flags*, *Flags*, *Channel Flags*) das cabeceiras *Radiotap* e *IEEE 802.11 Radio Information*, e indica que significa a maior cantidade posible de campos e os seus valores para un *beacon* concreto.

1.3 Beacons: Características Básicas dun AP

Os *beacons* capturados provéennos tamén de información moi completa sobre as características do AP. Dita información atópase baixo o campo *802.11 Radio Information* (este nome pode variar segundo a versión de Wireshark). Tras capturar os *beacons* emitidos por un AP concreto, responder ás seguintes cuestións:

a) Se se capturaron *beacons* doutros APs distintos do analizado, ¿En que canles se transmitiron? ¿Estanse producindo interferencias?

b) ¿Cal é o SSID do AP monitorizado?

c) Se nos fixamos agora un momento na cabeceira *IEEE 802.11 wireless LAN management frame*, ¿Que velocidades de transmisión soporta?

d) Se se explora todo o *beacon*, ¿Que información contén sobre o uso de seguridade (i.e., WEP/WPA)?

1.4 Obtendo información de clientes: *probe requests*

Cando un cliente quere conectarse a unha rede wifi ten dúas opcións:

- Escaneo pasivo: o cliente vai saltando de canle en canle ata recibir un *beacon* dun AP.
- Escaneo activo: o cliente salta de canle en canle emitindo un paquete especial (*probe request*) no que indica a súa presencia aos distintos APs.

O problema do escaneo pasivo é que, dende o punto de vista do consumo de enerxía, pode resultar custoso, sobre todo en dispositivos móbiles/portátiles: se, por exemplo, o cliente espera entre salto e salto pola transmisión de dous *beacons*, asumindo un intervalo de *beaconing* de 100 ms, daríase o caso de que para escanear 12 canles tardaríase 12x2x100 ms = 2,4 segundos. Durante este tempo o cliente está nun modo de transmisión a alta potencia, o cal prexudica a duración da batería.

Para evitar este problema sóese utilizar o escaneo activo. Neste modo, cando un cliente (i.e. un PC, un portátil, un móbil) acende a súa interface wifi, o primeiro que fai é ver se hai no seu alcance algunha das redes ás que se conectou previamente ou enviar un *broadcast* a tódolos APs da canle na que se atope. Para iso, faise uso duns paquetes chamados *probe requests* que, no caso de ser recibidos por un AP, son respondidos cunha *probe response*. Este proceso é realmente rápido en caso de éxito e o consumo é moito menor que nun escaneo pasivo: o adaptador wifi, tras enviar o *probe request*, só espera unha fracción do tempo respecto ao caso do escaneo pasivo. Debido a isto, o escaneo activo é o método utilizado por defecto en practicamente tódolos dispositivos wifi.

Este intercambio de *probes* enmárcase dentro do proceso de autenticación habitual contra un AP, o cal, de maneira xeral, segue os pasos indicados na Figura 4.

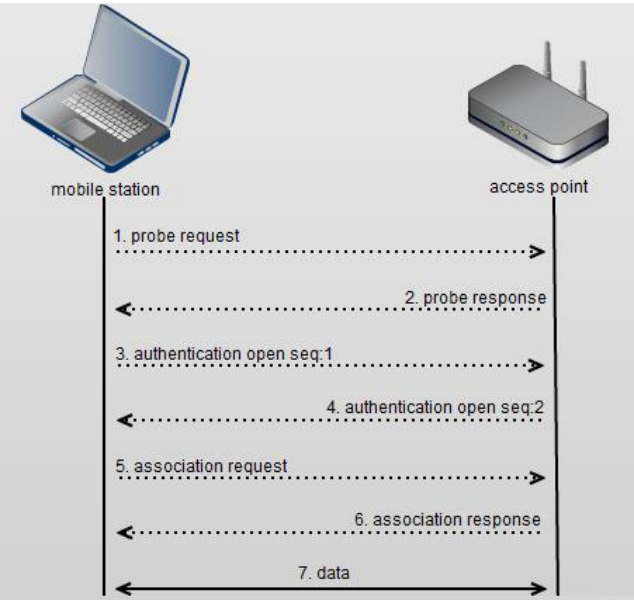

Figura 4: Proceso de autenticación habitual nunha wifi.

O uso de *probes* é cómodo porque acelera o tempo de conexión ás wifis predeterminadas, pero pode supoñer un problema obvio de privacidade e seguridade, dado que ditos paquetes inclúen información interesante para terceiros:

- O *probe request* contén información única sobre o cliente (a súa MAC), o que permite facer *tracking* do usuario. Nota: algúns dispositivos aleatorizan parte da MAC do *probe request* para evitar o *tracking* automatizado, pero aínda así, é sinxelo correlar o paquete co usuario, como se indica no seguinte punto.
- Moitos dispositivos, en lugar de facer un *broadcast*, fan un *probe request* sobre o histórico de APs aos que se conectaron. Se esos APs son xeoposicionados, é fácil determinar onde vive alguén, a que se dedica ou en que lugares estivo. Ademais, neste caso, dado que a lista de APs aos que se conectou unha persoa é, cunha alta probabilidade, única (i.e., é pouco probable que dous dispositivos que non sexan dun mesmo usuario se conectaran exactamente aos mesmos APs ao largo do tiempo), é fácil detectar a súa presenza nun área e facer *tracking* de dita persoa.
- Igualmente, se se fan *probe requests* sobre APs concretos, outros usuarios poden ver a que APs se nos queremos conectar. Isto da pé a que se poidan crear APs falsos con algún dos ESSIDs capturados nun *probe* e facer que a nosa interface se conecte a el automaticamente. Isto dá lugar a un tipo de ataques *man-in-the-middle* que se coñece como ataque *Honeypot*, e é habitual en lugares públicos como cafeterías ou aeroportos.
- A captura dos *probe requests* permite descubrir redes que ocultan o SSID.

Neste apartado da práctica imos facer unha proba sinxela para capturar os *probes* que emite un cliente que coñezamos para obter información sobre este. Para iso, seguiremos os seguintes pasos:

1. Poñer en modo monitor a nosa interface de rede wifi na canle na que se conectará o cliente que queremos monitorizar.
2. Abrir Wireshark e poñer a capturar na interface configurada en (1).
3. Activar a interface WiFi do cliente a conectar (e.g., o noso teléfono móbil).
4. Deter a captura de Wireshark e filtrar os paquetes para que só se mostren os *probe requests*. Para elo, pódese usar o seguinte filtro:

```
wlan.fc.type_subtype == 0x04
```

Se hai demasiados paquetes (i.e., clientes intentando conectarse), pódese filtrar a maiores pola MAC do dispositivo a monitorizar:

wlan.fc.type_subtype == 0x04 && wlan.sa == MAC_cliente

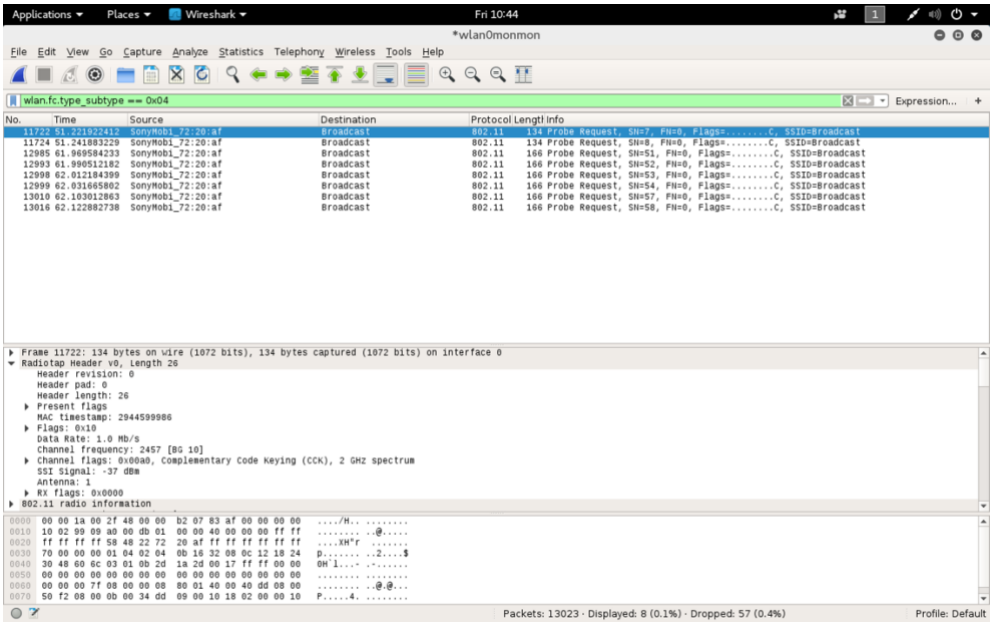

Figura 5: Exemplo de captura de *probe requests*.

Á vista dos *probes* capturados, indicar:

1. ¿Cal é o destino dos *probes*?¿Trátase dun *broadcast* ou dun *probe request* directo (i.e., dirixido cara un ESSID concreto)? En caso de tratarse dun *probe request* directo, ¿A que ESSID se envían?

2. ¿É posible identificar univocamente o emisor do *probe*?¿A través de que parámetro(s)?¿Pódese identificar o fabricante do dispositivo?

3. ¿Que velocidades indica o cliente que soporta?

4. Proba agora a filtrar a información das *probes responses* mediante o filtro:

wlan.fc.type_subtype == 0x05

¿Que información interesante se pode extraer?¿En que difire dos *beacons*? En caso de non atopar ningunha *response*, ¿A que crees que se pode deber?

1.5 Acceso ao medio con CSMA/CA: Análise Temporal

O método fundamental de acceso ao medio da familia de estándares IEEE 802.11 é unha DCF (*Distributed Coordination Function*) baseada en CSMA/CA (*Carrier-Sense Multiple-Access with Collision-Avoidance*). Este algoritmo consta dos seguintes pasos[6]:

1. **Sensado do sinal**. Para que unha estación poda transmitir, debe primeiro determinar se o medio está libre. Para determinar se o medio está ocupado por outra estación, o estándar IEEE 802.11 contempla dúas funcións de sensado de sinal:

   - ***Sensado físico do sinal***. É unha función provista pola capa PHY e o resultado é enviado á capa MAC. Basicamente, o que fai a función é monitorizar a enerxía dos sinais transmitidos e recibidos a través da interface radio. Dado que depende da PHY, non podemos monitorizar o seu comportamento con Wireshark.

   - ***Sensado virtual do sinal.*** Denomínase habitualmente NAV (*Network Allocation Vector*). É un mecanismo de nivel MAC polo cal as estacións reservan un tempo na canle para realizar as súas transmisións. O NAV permite establecer unha predición do tráfico futuro no medio gracias ao campo *Duration* especificado na cabeceira da trama MAC. Este NAV pode verse coma unha especie de contador que é indicado ao resto de estacións, as cales, antes de transmitir, monitorizan a cabeceira MAC (onde vai o mencionado campo *Duration*). Desta maneira, as estacións abstéñense de transmitir durante NAV microsegundos. No momento en o que o NAV chegue a 0, a función de sensado virtual indicará que o medio está libre e deberase volver a determinar se se pode transmitir (sensado físico).

     Atención: no caso de tramas *unicast*, o NAV inclúe tanto a transmisión dos datos como do conseguinte ACK (ver Figura 6).

[6] Unha descripción máis detallada deste algoritmo atópase na sección 9.2 do estándar IEEE 802.11-1999.

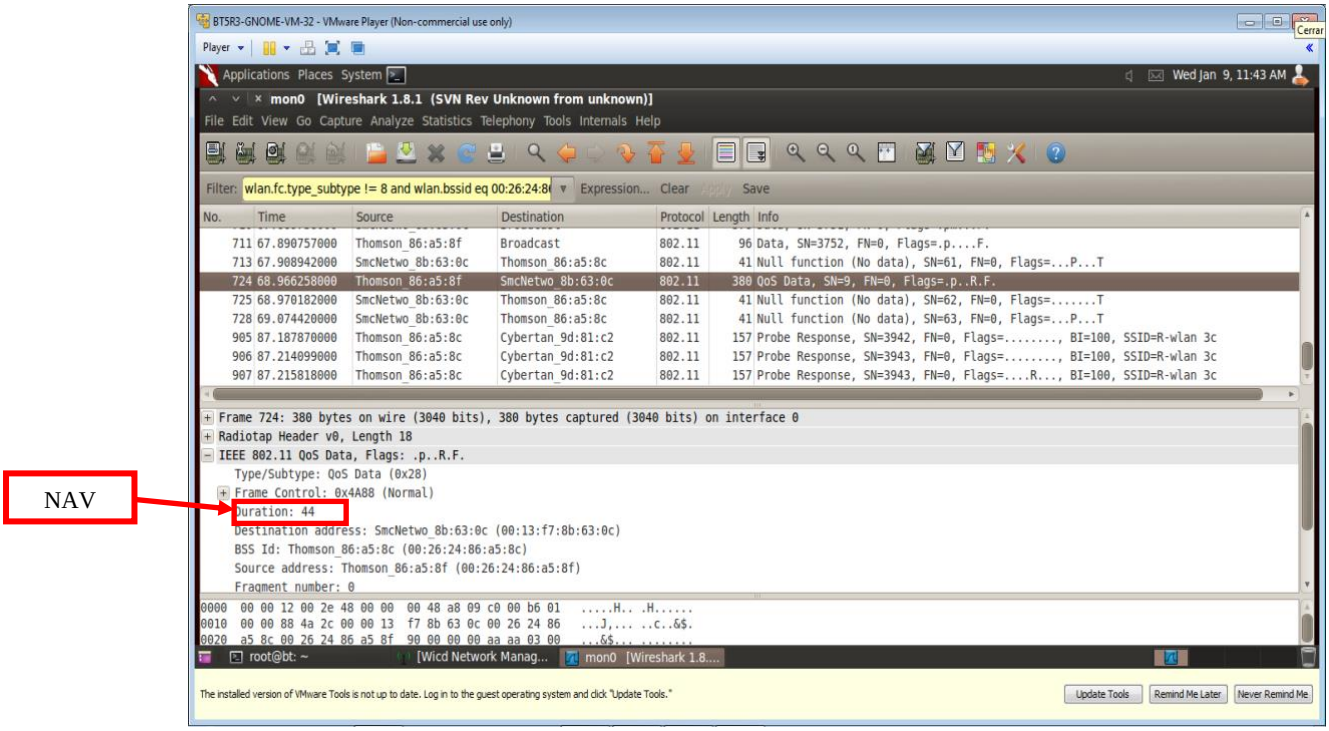

Figura 5: Exemplo de campo NAV.

2. **Espera entre tramas.** Trala expiración do NAV é preciso agardar un tempo mínimo denominado IFS (*InterFrame Spacing*). O tamaño do IFS varía en función do tipo de trama que a estación estea intentando enviar, distinguíndose os intervalos SIFS (*Short IFS*), PIFS (*PCF IFS*), DIFS (*DCF IFS*) e EIFS (*Extended IFS*). Os máis utilizados son o SIFS e o DIFS. O primeiro utilízase con tramas que deben enviarse necesariamente tras outras tramas (e.g. un ACK), mentres que o segundo é máis longo e utilízase con tramas normais á hora de operar en modo DCF (acceso ao medio distribuído).

Estándar	SIFS (µs)	DIFS (µs)
IEEE 802.11b	10	50
IEEE 802.11a	16	34
IEEE 802.11g	10	28 o 50 (este último úsase en modo compatibilidade con IEEE 802.11b)

Utilizando este mecanismo, os paquetes con menor IFS terán prioridade de acceso á rede cando compitan con outros de maior IFS.

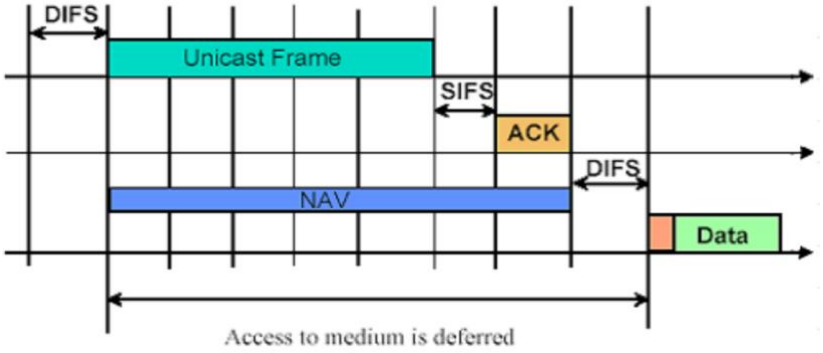

Figura 6: Intervalo temporal ocupado polo NAV. Trala expiración no NAV a estación ten que esperar un tempo DIFS antes de comezar coa transmisión da trama de datos (se a canle está libre).

3. **Espera dun período de *backoff* aleatorio.** Hai que ter en conta que tódolos contadores NAV das estacións chegan a 0 no mesmo instante, polo que é moi probable que varias estacións utilicen o mesmo valor de IFS: isto dará lugar a colisións entre as estacións debido a que tentarán transmitir no mesmo instante. Para evitar este problema, despois de que o NAV chegue a 0 e trala expiración do IFS, as estacións agardan un tempo aleatorio adicional denominado tempo de *backoff*. A estación que escolla o menor tempo de *backoff* será a que transmita primeiro. As demais estacións pausarán o seu contador de tempo adicional ata que o NAV volva a chegar a 0 e expire de novo o IFS.

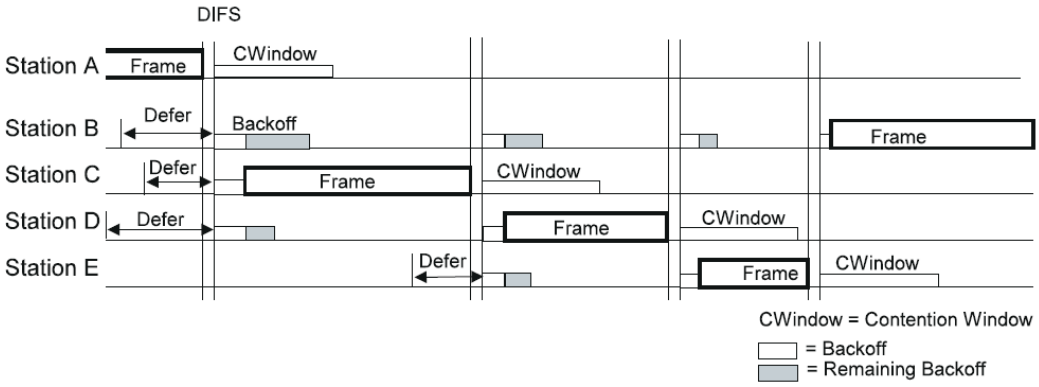

Figura 7: Control del backoff con múltiples estacións.

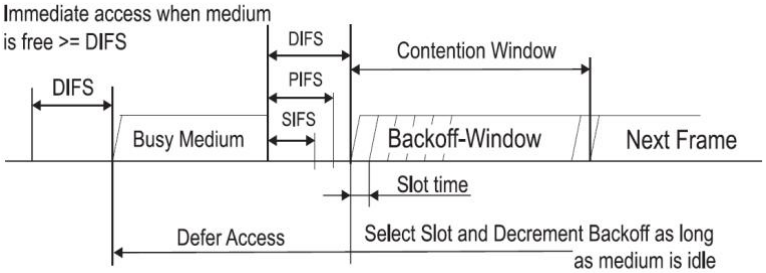

Figura 8: Resumo de tempos de espera en CSMA/CA.

Para ver como funciona CSMA/CA imos acceder ao medio físico compartido solicitando o acceso a unha rede. Para elo:

1. Poñemos o adaptador wifi en modo monitor (agora mesmo é irrelevante a canle).
2. Poñer a capturar Wireshark.
3. Conectámonos a unha rede wifi con outro dispositivo.
4. Unha vez conectados á rede (ou rexeitada a nosa solicitude), detemos a captura.
5. Para facilitar a lexibilidade da captura, descartamos os *beacons*:

wlan.fc.type_subtype != 0x08

6. Os paquetes que se mostran permiten ver o proceso xeral que supón unha conexión: (1) atópase unha BSS, (2) autenticámonos ante esa BSS e (3) asociámonos á BSS.

Cos datos desta captura e fixándose en secuencias de paquetes consecutivos nos que haxa varias parellas seguidas de solicitudes co seu respectivo ACK (i.e. nos que se dá unha situación como a ilustrada na Figura 6), pídese:

a) Escoller un par de casos que demostren que se verifica o cumprimento do NAV.

b) ¿Cal é o valor do NAV nos ACKs?

c) Buscar algún exemplo como o ilustrado na Figura 6 e medir (sempre que sexa posible), os seguintes tempos: tempo necesario para transmitir os datos, tempo para transmitir o ACK e o tempo de *backoff*.

Pódense buscar os paquetes correspondentes aos ACKs utilizando o filtro:

wlan.fc.type_subtype == 0x001d

Entrega da práctica:

Unha vez realizada a práctica, deberá entregarse unha **memoria xustificativa** en PDF coas respostas ás preguntas dos distintos apartados e os ficheiros das **trazas de Wireshark** utilizados para ilustrar a memoria. Estes arquivos deberán empaquetarse nun único ficheiro comprimido que será subido ao Moodle a través da tarefa correspondente.

Importante: Non se valorarán as memoria se non se acompañan das correspondentes trazas.

Data límite de entrega: **17 de Maio de 2020 (inclusive)**.