Práctica 3: Análisis de Tráfico

Introducción

El análisis de tráfico de red es necesario en distintos contextos: monitorización, diagnóstico de problemas de red, análisis forense, seguridad, etc.

En esta práctica veremos el uso de la librería libpcap, que nos permite capturar tráfico o leer datos de una captura desde un programa C .

Programando nuestro propio analizador de tráfico: la librería libpcap:

La librería libpcap nos permite capturar paquetes desde un programa C. En sistemas Windows, la librería se llama Winpcap.

Para compilar cualquier programa que haga uso de la librería libpcap necesitamos incluir el fichero de cabecera pcap.h en las fuentes (#include <pcap.h>) y añadir en la cadena de compilación la orden –lpcap.

También se puede usar el makefile junto con el resto de archivos de cabecera y librería que se ha colgado en el Moodle de la asignatura.

Para instalarla la librería en un sistema basado en Debian (Ubuntu por ejemplo) se puede ejecutar: sudo apt-get install libpcap-dev

Veamos primero las funciones básicas que nos permite la libpcap:

Capturar de un interfaz

Para abrir un interfaz para captura:

pcap_t *pcap_open_live(const char *device, int snaplen,int promisc, int to_ms, char *errbuf);

- Device: es el nombre de la interfaz que se quiere abrir (eth0, eth1...).
- snaplen: es la cantidad de bytes que se quieren guardar por cada paquete. Es útil cuando no nos interesa la carga útil del paquete y así reduciríamos el tamaño de la captura. Por defecto usaremos el valor 1514 (MTU Ethernet).
- promisc: indica si queremos abrirla en modo promiscuo (promisc=1) o no (promisc=0).
- to_ms duración del timeout de lectura. Tiempo que se espera para leer varios paquetes en una misma transacción.
- En errbuf se guarda el mensaje de error, si procede.

La función nos devuelve el puntero al descriptor de fichero pcap. Este descriptor es necesario cuando vayamos a realizar alguna operación sobre el tráfico. Por ejemplo leer un paquete. A efectos prácticos esta estructura es similar a un FILE *

Hay que recordar que para abrir una interfaz es necesario tener permisos de superusuario.

Ejemplo:

p=pcap_open_live("eth0",0,1, 100, errbuf);

Abre para la interfaz eth0 en modo promiscuo, capturando el paquete en su totalidad, con un timeout de lectura de 100 ms.

En caso de error, guarda el mensaje en la cadena errbuf. En esta práctica no se realizarán capturas sobre interfaces.

Abrir un fichero pcap

Un fichero que contiene tráfico capturado se denomina traza. Para abrir una traza se usará la siguiente función:

pcap_t *pcap_open_offline(const char *fname, char *errbuf);

- fname es el nombre del fichero pcap que se desea abrir.
- En errbuf se guarda el mensaje de error, si procede.

La función nos devuelve el puntero al descriptor de fichero pcap.

Ejemplo

p=pcap_open_offline("traza.pcap", errbuf);

Abre para lectura el fichero traza.pcap. En caso de error, guarda el mensaje en la cadena errbuf.

Leer un paquete de un fichero o interfaz

Para leer un paquete de un fichero o interfaz se utilizará la siguiente función:

const u_char *pcap_next(pcap_t *p, struct pcap_pkthdr *h);

- p es el puntero al descriptor de fichero del que queramos leer (que anteriormente hemos abierto con pcap_open_live o pcap_open_offline).
- h es la cabecera pcap del paquete. Esta cabecera es un struct con cuatro campos:
  - h->ts.tv_sec, timestamp del paquete en segundos.
  - h->ts.tv_usec, timestamp del paquete en microsegundos.
  - h->len: longitud real del paquete
  - h->caplen: longitud capturada del paquete. Esto es, el puntero que nos devuelve sólo contiene h->caplen bytes.

La función nos devuelve el puntero al inicio del paquete.

Ejemplo

bp = (u_int8_t *) pcap_next (p, &h);

En este caso bp es un puntero que con tiene un array de bytes con el contenido del paquete actual de la traza. Cada vez que se llame a la función pcap_next se devolverá el siguiente paquete de la traza. Cuando se alcanza el final o hay un error se devuelve NULL.

Guardar fichero pcap

Para guardar un fichero pcap necesitamos primero crear el fichero donde vamos a ir guardando los paquetes. Para ello se usan la funciones pcap_open_dead y pcap_dump_open:

pcap_t *pcap_open_dead(int linktype, int snaplen);

- linktype es el tipo de enlace de los paquetes que vamos a guardar. Típicamente, redes Ethernet: DLT_EN10MB
- snaplen es el tamaño máximo de los paquetes que queramos guardar.

Devuelve, como las otras funciones pcap_open, el puntero al descriptor de fichero pcap.

Ejemplo

p2=pcap_open_dead(DLT_EN10MB,1514);

Se crea un descriptor de fichero pcap para paquetes Ethernet, guardando como máximo 1514 Bytes de cada paquete.

pcap_dumper_t *pcap_dump_open(pcap_t *p, const char *fname);

- p es el descriptor de fichero pcap previamente abierto con pcap_open_dead.
- Fname es el nombre del fichero pcap en el que queramos guardar los paquetes.

Se devuelve el puntero a fichero pcap donde se van a volcar los paquetes.

Ejemplo

pdumper=pcap_dump_open(p2,"salida.pcap");

Se crea un fichero llamado salida.pcap con las características (tipo de enlace, y tamaño máximo de paquete) de p2 (que indicamos anteriormente pcap_open_dead). Nos devuelve el puntero a fichero de volcado de paquetes que vamos a utilizar para guardar los paquetes.

Para guardar paquetes en el fichero creado con pcap_dump_open usamos la función:

void pcap_dump(u_char *user, struct pcap_pkthdr *h,u_char *sp);

- user es el puntero devuelto por pcap_dump_open.
- h es la cabecera pcap del paquete que vamos a guardar.
- sp es el puntero al paquete. Se van a guardar tantos bytes como indiquemos en el campo caplen del parámetro h.

Ejemplo

pcap_dump(pdumper,&h,bp);

Se guarda en pdumper el paquete apuntado por bp con cabecera h.

Cerrar Fichero

Los ficheros abiertos con pcap_open_live, pcap_open_offline y pcap_open_dead se cierran con pcap_close. Mientras que los ficheros abiertos con pcap_dump_open se cierran con pcap_dump_close.

void pcap_close(pcap_t *p);

- p es el fichero a cerrar

void pcap_dump_close(pcap_dumper_t *p);

- p es el fichero a cerrar

Filtros de captura/lectura

En ocasiones nos interesa no capturar (o leer) todos los paquetes sino sólo los que cumplen un determinado criterio (cómo veíamos con los filtros de captura en Wireshark). Para ello, es necesario primero "compilar" un filtro y luego aplicarlo al descriptor de fichero pcap previamente abierto (ya sea con pcap_open_live o pcap_open_offline). Para ello se usan las funciones pcap_compile y pcap_setfilter:

int pcap_compile(pcap_t *p, struct bpf_program *fp, const char *str, int optimize, bpf_u_int32 netmask);

- p es el fichero previamente abierto con pcap_open
- fp es la variable donde se va a guardar el filtro
- str es la cadena donde se escribe el filtro a aplicar.
- optimize es un parámetro de optimización de código. En nuestro caso, podemos dejarlo a 0.
- netmask es la máscara de la red en la que se están capturando los paquetes. Lo dejamos a 0 para que sirva para cualquier subred.

Nos devuelve -1 en caso de error. Si queremos ver el mensaje de error, usar **pcap_geterr(p)**.

Ejemplo

pcap_compile(p, &bpf, "dst net 192.168.1.0/24", 0, 0);

Se compila el filtro "dst net 192.168.1.0/24" guardándolo en bpf.

int pcap_setfilter(pcap_t *p, struct bpf_program *fp);

- p es el fichero, previamente abierto con pcap_open, al que se le quiere aplicar el filtro.
- fp es el filtro previamente compilado con pcap_compile

Nos devuelve -1 en caso de error. Si queremos ver el mensaje de error, usar **pcap_geterr(p)**.

Ejemplo

pcap_setfilter(p, &bpf);

Se aplica el filtro bpf, previamente compilado, al fichero p, previamente abierto con pcap_open.

Análisis de protocolos: Estadísticas a nivel de paquete

Como hemos visto, la función pcap_next nos devuelve el puntero al siguiente paquete. A partir de ese puntero, podemos "parsear" las distintas cabeceras del paquete y su contenido. Los primeros 14 Bytes pertenecen a la cabecera Ethernet. Los siguientes, dependen de los protocolos de capas superiores. En el caso más típico, encontraremos la cabecera IP y sobre ella la de TCP o UDP.

Ejercicios

Se deben completar los archivos analisis_trafico.c y functions.c para responder a las siguientes preguntas de evaluación haciendo uso de las funciones propuestas. Una vez el código haya sido completado se ejecutará pasando como argumento por la consola un fichero pcap que se proporcionará en la página de Moodle. Se pueden utilizar y crear todas las funciones auxiliares que el alumno desee.

Además del código se debe entregar una pequeña memoria (en pdf) donde se responda a las siguientes preguntas:

Abre/lee/cierra un fichero de captura (2 puntos)

- ¿Cuántos paquetes tiene el fichero?
- ¿Cuál es el tamaño medio del paquete?
- ¿Cuál es la tasa en bps (bits por segundo) del enlace que monitoriza la traza? ¿Y en pps (paquetes por segundo)?

Abre/recorre/cierra un fichero de captura aplicándole filtros BPF (2 puntos)

- ¿Cuál es la tasa (en Mb/s) de salida de la subred 163.0.0.0 con máscara 255.0.0.0? ¿Y de entrada? ¿hay paquetes internos en esta subred (IP origen y destino en subred 163.0.0.0)?.

Análisis de protocolos a nivel de paquete (6 puntos)

- ¿Cuántos paquetes de la traza son IP? ¿y ARP?¿Hay otro tipo de paquetes? En caso afirmativo, ¿sabrías decir qué tipo de paquetes son?
- De los paquetes IP, ¿cuántos paquetes TCP hay en la traza? ¿y UDP? ¿e ICMP? ¿hay paquetes de otros protocolos? En caso afirmativo, ¿sabrías decir qué tipo de paquetes son?

- Pintar la ECDF (Empirical Cumulative Distribution Function) para el tamaño de los paquetes. ¿Se observa alguna moda? ¿Qué porcentaje de paquetes son mayores de 400 bytes?
- ¿Cuáles son los 5 puertos (destino y origen) más populares tanto en número de paquetes como en número de bytes? ¿A qué se debe la diferencia de popularidad en paquetes y en bytes? Para explicar dicha diferencia comparar las ECDF del tamaño de los paquetes para el puerto top en bytes y para el puerto top en número de paquetes.

N.B. los filtros de captura BPF SÓLO se pueden utilizar para la pregunta relacionada con filtros BPF para el resto de apartados las comprobaciones deben realizarse con código C.

FECHAS

La entrega debería hacerse antes de las 00:00 del día límite establecido en la página de Moodle.

EXAMEN

Se realizará un examen INDIVIDUAL al el día indicado en la página de Moodle.

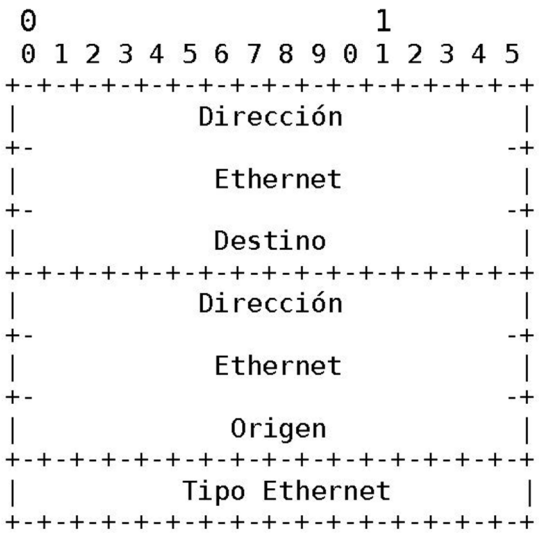
El examen consistirá en una o dos cuestiones sobre la implementación concreta de la práctica por parte de cada pareja de prácticas.

La calificación del examen será APTO/NO APTO. En caso ser NO APTO, la calificación de la práctica para ese miembro de la pareja será de 0 (cero).

Información Adicional:

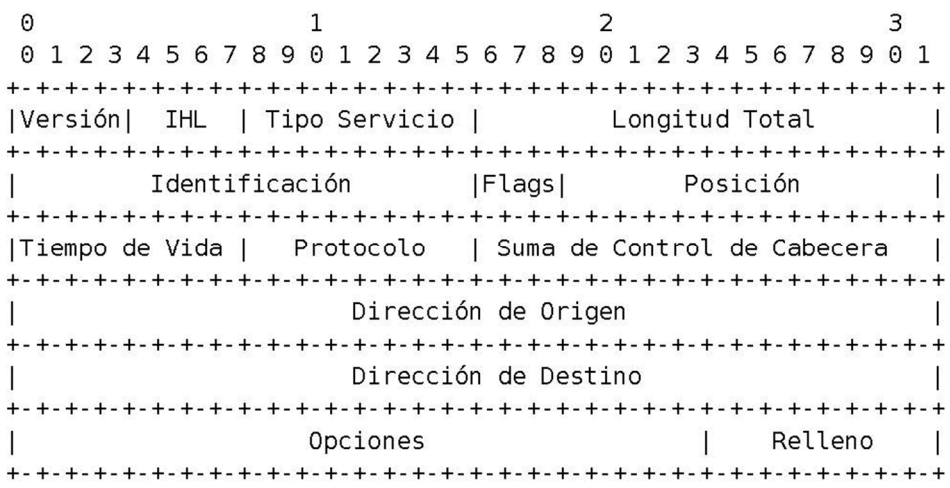
A continuación se incluyen las cabeceras de los protocolos que son necesarios para el desarrollo de la práctica así como otra información relevante.

Ethernet



- Dirección Ethernet Destino: dirección MAC de nivel 2. Este campo indica el destino del paquete Ethernet. Su longitud es 6 bytes.
- Dirección Ethernet Origen: dirección MAC de nivel 2. Este campo indica el origen del paquete Ethernet. Su longitud es 6 bytes.
- Tipo Ethernet: Este campo de 2 bytes indica el protocolo que se encuentra en el nivel superior (nivel 3). Los valores mas comunes son:
    - 0x0800 Protocolo IPv4
    - 0x0806 ARP
    - 0x86DD Protocolo IPv6

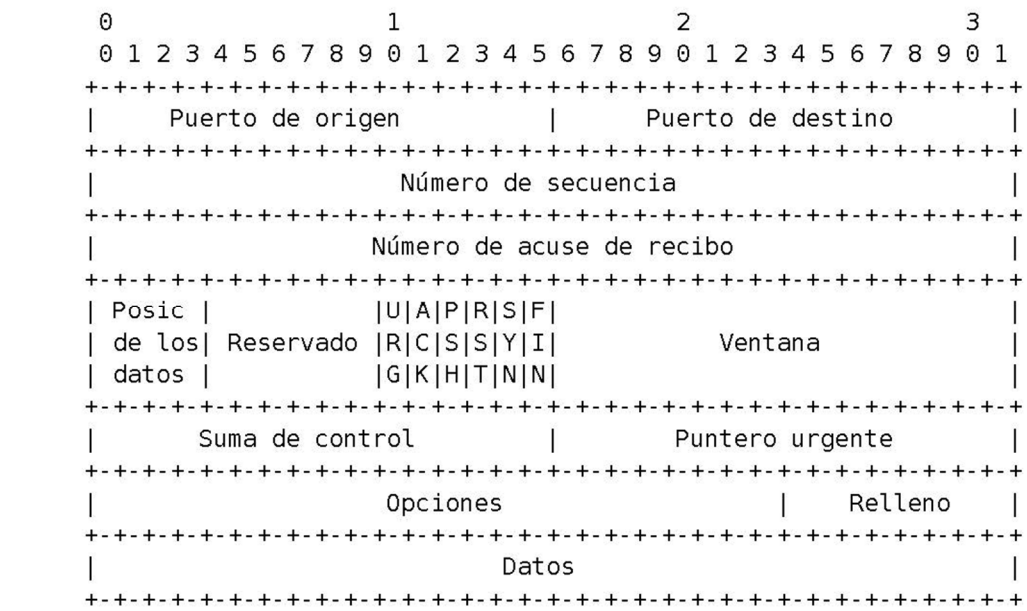
Internet Protocol (IP)



- Version: campo de 4 bits que indica la version del protocolo. En IPv4 este campo vale 4.
- Internet Header Length (IHL): este campo indica la longitud de la cabecera IP en numero de palabras de 4 bytes. Por ejemplo, en el caso tipico en el que no hay campo de opciones, este campo tiene un valor de 5. Por tanto, la cabecera ocupa 5x4=20 bytes.
- Tipo de Servicio: campo que permite etiquetar el tipo de trafico que contiene IP (multimedia, transferencia de datos, etc) para que puedan ser aplicadas politicas de calidad de servicio sobre el paquete. Este campo ocupa 1 byte.
- Longitud total: este campo indica la longitud total de un paquete IP incluyendo su cabecera y sus datos. El tamaño minimo es 20 bytes (20 bytes de cabecera y 0 de datos) y el tamaño maximo es 65,535 bytes. Este campo ocupa 2 bytes.
- Identificacion: valor numerico que identifica de manera unica el paquete. Este campo ocupa 2 bytes.
- Flags: campo de 3 bits que contiene banderas relativas a la fragmentacion IP.
    - Bit 0: esta reservado y su valor debe ser 0
    - Bit 1: bandera Don't Fragment. Si esta bandera esta a 1 el datagrama IP nunca se fragmentara. En caso de que sea necesario fragmentar el paquete dicho paquete se descartara y se avisara al origen de la conexión.
    - Bit 2: bandera More Fragments. Si esta bandera esta a 1 indica que el datagrama ha sido fragmentado y que todavia quedan mas fragmentos por llegar.
- Posicion: este campo de 13 bits indica, en unidades de 8 bytes, la posicion de los datos del fragmento actual en el datagrama IP completo. En caso de que no haya fragmentacion este campo valdra 0.

- Tiempo de vida: Este campo de 1 byte representa el numero maximo de saltos que un datagrama IP puede atravesar en una red. Este valor se decrementa en cada salto y en caso de llegar a 0 el datagrama se descarta y se notifica el extremo origen.
- Protocolo: Con este campo de 1 byte se indica el protocolo de nivel superior (nivel 4) que esta encapsulado en los datos del datagrama. Los valores mas comunes son:
    - 1: ICMP
    - 6: TCP
    - 17: UDP
- Suma de control de cabecera: valor de verificacion calculado sobre la cabecera IP utilizado para verificar que dicha cabecera no ha sido modificada durante el viaje del datagrama IP.
- Direccion Origen: campo de 4 bytes que contiene la direccion IP origen del paquete.
- Direccion Destino: campo de 4 bytes que contiene la direccion IP destino del paquete.
- Opciones: campo de tamaño variable que puede contener diversas opciones de control y operación. En caso de que el tamaño de las opciones no sea multiplo de 4 se añade relleno. Este campo no sera comun en la realizacion de las practicas.

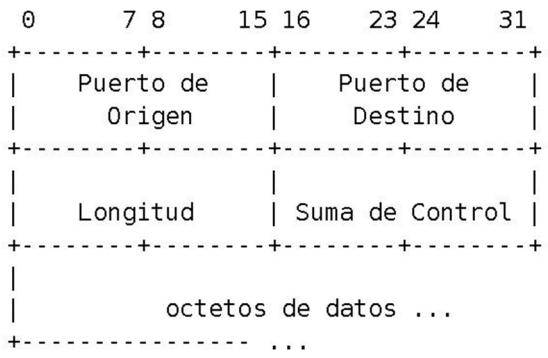
TCP:



- Puerto Origen: campo de 2 bytes que contiene el puerto origen del paquete TCP
- Puerto Destino: campo de 2 bytes que contiene el puerto origen del paquete TCP
- Numero de secuencia: este número identifica cada segmento de datos TCP que se envía. En esta práctica este valor no es relevante. Este campo es de 4 bytes.

- Número de acuse de recibo: indica el número de secuencia del paquete que se espera recibir. En esta práctica este valor no es relevante. Este campo es de 4 bytes.
- Posición de los datos: campo de 4 bits que indica la posición de los datos del segmento TCP contando a partir del inicio de la cabecera TCP. Al igual que en el caso de IP este valor esta expresado en palabras de 4 bytes.
- Reservado: Campo de 8 bits reservado para futuros usos.
- Banderas:
    - URG
    - ACK
    - PSH
    - RST
    - SYN
    - FIN
- Ventana: tamaño máximo de datos que pueden enviarse al receptor sin recibir un asentimiento. Este campo es de 2 bytes.
- Suma de control: suma de verificación de la cabecera y los datos Este campo es de 2 bytes.
- Puntero Urgente: indica hasta qué posición del segmento son considerados datos urgentes. Este campo es de 2 bytes.
- Opciones: opciones de control y gestión. Al igual que en el caso de IP si no son múltiplos de 4 se añade relleno.

UDP:



- Puerto Origen: campo de 2 bytes que contiene el puerto origen del paquete UDP
- Puerto Destino: campo de 2 bytes que contiene el puerto origen del paquete UDP
- Longitud: longitud del datagrama UDP incluyendo cabecera y datos.
- Suma de control: este valor normalmente se encuentra a 0. En caso de estar relleno se realiza la suma de control de la pseudo-cabecera UDP.

Representación Gráfica de Datos:

En la práctica se pide realizar el graficado de funciones ECDF. Para ello se hará uso del programa GNUplot instalado en los laboratorios. Se proporciona un script de bash que hace la llamada correspondiente al programa gnuplot. Dicho script llamado plot_cdf.sh recibe como argumentos el nombre del fichero de datos así como los valores de las etiquetas de los ejes para mostrar en la gráfica y genera un archivo de extensión PNG con la gráfica dibujada.

Ejemplo: ./plot_cdf.sh fichero_de_datos.dat “Eje X” “Eje Y”

Generación de traza de datos:

La traza que se debe usar para las pruebas y evaluación de esta práctica será única para cada pareja. Para ello se utilizara el número de DNI del primer miembro de la pareja. Dicho número se pasará a un programa (llamado md5) que se proporciona que generará la traza pcap que se debe utilizar para la práctica. Por ejemplo si el número de pareja es el 9 y el DNI es 12345678 ejecutaremos:

./md5 9 12345678

Este comando producirá una traza llamada, en este ejemplo, practica3_P9.pcap que deberá ser utilizada para la realización de la práctica.

Es FUNDAMENTAL que el NUMERO DE DNI aparezca en la MEMORIA que se adjunte.