Reporte Ejercicio 1 [Redes 2017]

Integrantes: Bernal Cedillo Enrique Antonio
Velásquez Garzón Angie Daniela

- **Describa a profundidad cómo funciona la biblioteca *pcap***

  Pcap es una interfaz de aplicación que permite capturar paquetes, la implementación de *pcap* es conocido como *libpcap* pueden ser utilizados por un programa para capturar paquetes que estén viajando a través de la red. *Libpcap++* es un envoltorio de libpcap, proporcionando una interfaz de alto nivel a sistemas de capturad de paquetes.
  Libcap es una librería que al ser utilizada por procesos ejecuta sus funciones a nivel de usuario, sin embargo la captura de paquetes se realiza efectivamente en el sistema operativo en la zona kernel (*kernel área)* debe existir un mecanismo que permita traspasar dicha frontera evitando fallas en capas internas que eventualmente afectan el rendimiento de todo el sistema.

  Esquematización de un programa con *Libcap:* de forma general un programa en *libcap* tiene el siguiente esquema:

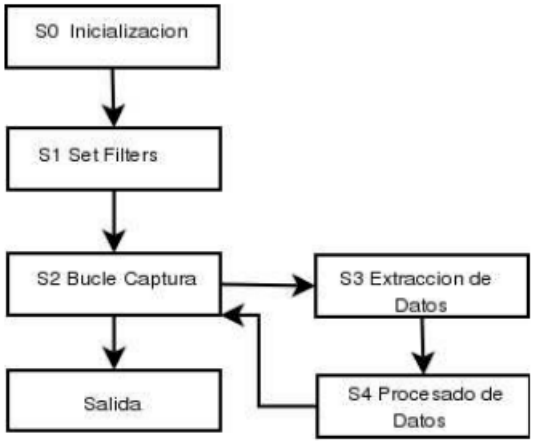

Ilustración 1. Tomado de http://goo.gl/7jfXah

  Inicialización: funciones capaces de obtener información del sistema, es decir las interfaces de red, las configuraciones de dichas interfaces (Mascaras de red, dirección de red). Permite obtener una lista de las interfaces contenidas en el sistema y sus configuraciones.
  Captura de paquetes: hay varias maneras de capturar paquetes, las características diferenciadoras entre ellas radican en: el número de paquetes a capturar, el modo de captura, normal o promiscuo y en la definición de la llamada callback la cual es invocada cada vez que se captura un paquete (Monge, 2005).
  Filtrar paquetes: La aplicación establece un filtro en la zona kernel que solo permita el paso de los paquetes que interesan al usuario del sistema, es decir capturar el tráfico con un destino u origen determinado, dicha labor es llevada a cabo por el Packet/Socket Filter; existen varios sistemas de filtrado, uno de ellos es BPF el cual está basado en dos componentes: *Network tap* se encarga de recopilar paquetes desde el driver del dispositivo y entregarlo al proceso destino; y *packet filter* decide si un paquete es aceptado y cuánto de ese paquete debe ser entregado (elimina encabezados no útiles para el proceso destino).
  Cada que un paquete llega a la interfaz de red el driver lo envía a la pila de protocolos, en caso de que BPF se encuentre activo antes de enviarse a la pila de protocolos el paquete es

procesado por este. BPF comprara los paquetes con cada uno de los filtros establecidos y entrega una copia del paquete a los buffers de las aplicaciones cuyo filtro se ajuste al contenido del paquete.

- **¿Cuáles son las principales vulnerabilidades del protocolo http?**

  El nivel de protección depende completamente de la implementación del navegador web, el software del servidor y los algoritmos de cifrado que se utilicen.
  No puede regular el contenido de los datos que se transfiere o determinar la sensibilidad de una pieza, las aplicaciones deben suministrar control sobre dicha información.

- **Principales ataques informáticos que explotan el protocolo http.**
  - **Ataques basados en nombres de archivos y ruta**
    Las implementaciones de servidores de origen HTTP deben ser cuidadosos de restringir los documentos que devuelven por peticiones HTTP a sólo aquellos que fueron destinados.
    Si un servidor HTTP procesa una HTTP URIs directamente en las llamadas al sistema de archivos, el servidor deberá tener cuidado de no entregar archivos que no estaban destinados a ser entregados.
  - **DNS Spoofing**
    Los clientes utilizan HTTP dependen en gran medida del servicio de nombres de domino y son generalmente propensos a ataques de seguridad basados en la deliberada mala asociación de direcciones IP y los nombres DNS

Referencias

Monge, A. L. (2005). *Aprendiendo a programar con Libpcap.*

RFC 2616. (s.f.). *https://www.w3.org/Protocols/rfc2616/rfc2616-sec15.html*.

- **Flujo del programa**

  El programa comienza con la selección de modo para analizar paquetes, los modos disponibles son los siguientes:

  - Escucha indefinida de dispositivo
    En este modo, se presenta una lista con los dispositivos disponibles para ser escuchados por el programa.
    Basta con escribir el nombre del dispositivo que se desea escuchar para que el programa:
    1.- Realice una captura del dispositivo (pcap_open_live)
    2.- Obtenga los atributos de mascara de red e IP (pcap_lookupnet)
    3.- Realice un filtro (y lo compile) para centrarse en mensajes HTTP
    4.- Comience la lectura indefinida de paquetes

- Archivo de captura
  Para este modo es necesario escribir en terminal la ruta (relativa al directorio de ejecución del programa) que apunta hacia el archivo de captura que quiere leerse.
  Una vez leído el archivo, se abre la captura (pcap_open_offline)
  Y comienzan a leerse los paquetes.

Para leer los paquetes se utiliza “pcap_loop” con la captura obtenida del paso anterior, el ciclo continua ejecutándose hasta terminar con los paquetes del archivo, o hasta que un error ocurra.
Es necesario definir una función “callback” que dicta lo que se realizará con los paquetes cada vez que se reciba alguno, en este caso dicha función fue llamada “procesarPaquete”. En la cual se acomoda la información de cada paquete utilizando estructuras auxiliares, y una vez obtenida la carga de información de la cabecera TCP se procede a identificar qué tipo de mensaje es, y entonces se imprimen los parámetros requeridos en terminal.

El programa termina cuando se acaban los paquetes del archivo de captura, o cuando se detiene con Ctrl+C en el modo de escucha indefinida.

- **Principales problemas que se encontraron**
  Recordar sintaxis y demás detalles de C.

  Terminología en general y flujo que se le debía dar al programa para cumplir con lo requerido por el Ejercicio.

  Con qué dispositivo probar, o cómo, la captura de paquetes. Y cómo indicar en C con qué dispositivo se quería trabajar.

  Erroneamente intentar aplicar un filtro de paquetes a un archivo de captura (no tiene sentido, pues dicho filtro debería ser aplicado al crear el archivo de captura, no después).

  Distinguir y saber dónde se encuentra la información ya que se obtiene el paquete, asi como lograr representar su contenido en manera de texto en la terminal.

  Saber qué tipo de respuesta debía dar el programa (No se sabía ni siquiera con capturas de ejemplo obtenidas o un ambiente controlado si la salida del programa era realmente correcta).

- **Problemas que no fueron solucionados**
  Optimizar el código, parece que aún hay partes que podrían ser más eficientes, pero por falta de confianza en C se dejará para después.