Taller 1: Wiretapping

Teoría de las Comunicaciones
Departamento de Computación
FCEN - UBA

29.04.2020

1. Introducción

El objetivo de este trabajo es utilizar técnicas provistas por la teoría de la información para estudiar los diversos protocolos de una red de manera analítica. Para la realización de este taller es necesario instalarse un interprete de Python y la biblioteca Scapy [2]. Además sugerimos el uso de la herramienta Wireshark [1] para capturar paquetes en una red.

2. Normativa

- Fecha de entrega: 13-05-2020.
- El trabajo práctico se deberá enviar por correo electrónico con el siguiente formato:
  to: tdc-doc at dc uba ar
  subject: debe tener el prefijo [tdc-wiretapping] seguido de los apellidos separados por comas.
  body: nombres, apellidos y números de libreta y las respectivas direcciones de correo electrónico
  attachments: Informe en formato pdf
- No esperar confirmación a menos que reciban una respuesta indicando explícitamente que el mail fue rechazado. Notar que los avisos por exceso de tamaño no son rechazos.

3. Enunciado

3.1. Introducción

Los medios de acceso compartido (i.e.: Ethernet, WiFi) son un recurso ampliamente difundido para establecer enlaces de comunicación entre 2 o más computadoras. Los paquetes de datos que se transmiten por estos medios llegan a todos los dispositivos conectados al mismo y se usan las direcciones en los encabezados de los paquetes para poder enviar información de un dispositivo a otro específicamente (Comunicación *Unicast*). Además, existen varios protocolos que hacen uso de comunicaciones de tipo *Broadcast*, por ejemplo ARP [3] y DHCP [4]. Ahora bien, estos paquetes de datos pueden ser *capturados* por cualquier dispositivos que se encuentre conectado al medio y pueden ser analizados, por lo menos hasta la última capa de comunicación que no se encuentre encriptada.

3.2. Capturando paquetes

Para identificar la redes sobre las cuales queremos capturar paquetes, se asignan identificadores a las diferentes interfaces de red (Network Interface Controller - NIC) que tiene una computadora. Las interfaces y sus identificadores se pueden consultar mediante comandos como `ifconfig` (puede requerir permisos de root):

```
$ ifconfig
eth0:   flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
ether 3c:92:0e:33:4b:01  txqueuelen 1000  (Ethernet)
RX packets 0  bytes 0 (0.0 B)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 0  bytes 0 (0.0 B)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Las interfaces de red pueden ser físicas (wlan0, eth0) o virtuales (lo). Las interfaces virtuales son útiles si uno está desconectado pero igual necesita funcionalidades de red (acceder a servidores corriendo en la misma máquina, a instancias de máquinas virtuales, etc.). Ahora bien, generalmente la NIC se encarga de descartar los paquetes que no estén dirigidas a ella (que se identifica con la dirección MAC). Pero la NIC puede configurarse en **modo promiscuo**. En este modo, los paquetes con MAC destino ajena no se descartan. Suben hasta el kernel para que podamos consumir las tramas, y para activar el modo promiscuo hace falta tener privilegios de **root**. En particular, tanto Wireshark como Scapy activan el modo promiscuo a la hora de capturar paquetes, por lo que hace falta ejecutar con permiso de **root**.

3.2.1. Capturando paquetes con Wireshark

Para instalar **wireshark** en distribuciones basadas en Debian, hacer:

```
$ sudo apt install wireshark
```

Para poder capturar paquetes en modo promiscuo y no tener que ejecutar **wireshark** con **sudo**, suele ser recomendable aceptar la creación del grupo 'wireshark' cuando en la instalación nos pregunte y luego agregar nuestro usuario a dicho grupo. Para eso:

```
$ sudo usermod -a -G wireshark <user>
```

y reiniciar la sesión para que sean tomados los cambios.

Una vez iniciado wireshark, nos permite elegir la interfaz por la cual va a capturar los paquetes. Ver Figura 1.

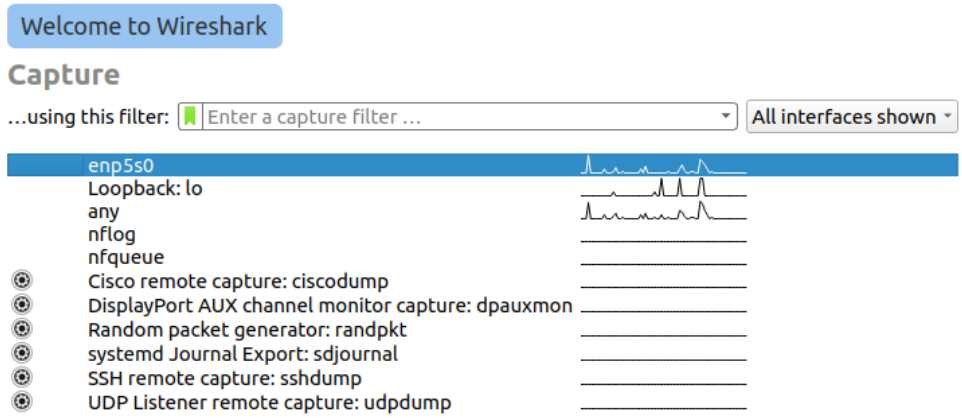

Figura 1: Eligiendo la interfaz de captura

Elegimos nuestra interfaz Ethernet o Wi-Fi, haciendo doble-clic sobre el nombre de la misma. Los nombres listados corresponden a los resultados que arroja el comando **ifconfig**. Tendremos ante nosotros la ventana típica de captura de paquetes de wireshark (ver Figura 2).

Probablemente, podamos observar en ella una gran cantidad de paquetes capturados. Están ordenados inicialmente en forma temporal (por la segunda columna, que está en segundos). Para cada paquete, también podemos ver su dirección IP fuente, su dirección IP destino, el protocolo, la longitud del paquete en bytes y

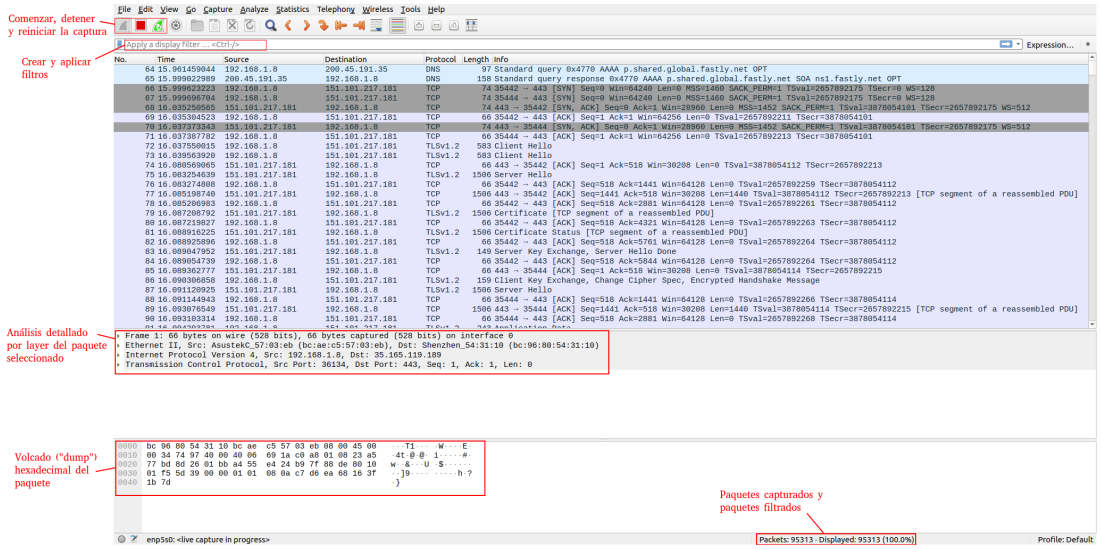

Figura 2: Ventana principal

un campo acerca del tipo de información que lleva ese paquete. En las secciones inferiores vemos un análisis detallado del paquete y de todos sus campos, ordenados por protocolo, desde la capa física hasta las capas superiores. Esto resulta de mucha utilidad para estudiar en detalle un protocolo en particular o para ver que está sucediendo "en el cable", ya que wireshark capturará todo lo que la interfaz vea, tenga interpretación en algún protocolo particular o no. Además, permite configurar los protocolos que analizará. En general, vienen todos habilitados por *default*.

Wireshark posee un avanzado sistema de filtrado de paquetes. ¿Cuál es la idea? Separar los paquetes que nos interesan del resto. Por ejemplo, para ver los paquetes de ARP, se puede escribir "arp" como se muestra en la figura y luego pulsar "Enter" para ver el resultado. (ver Figura 3).

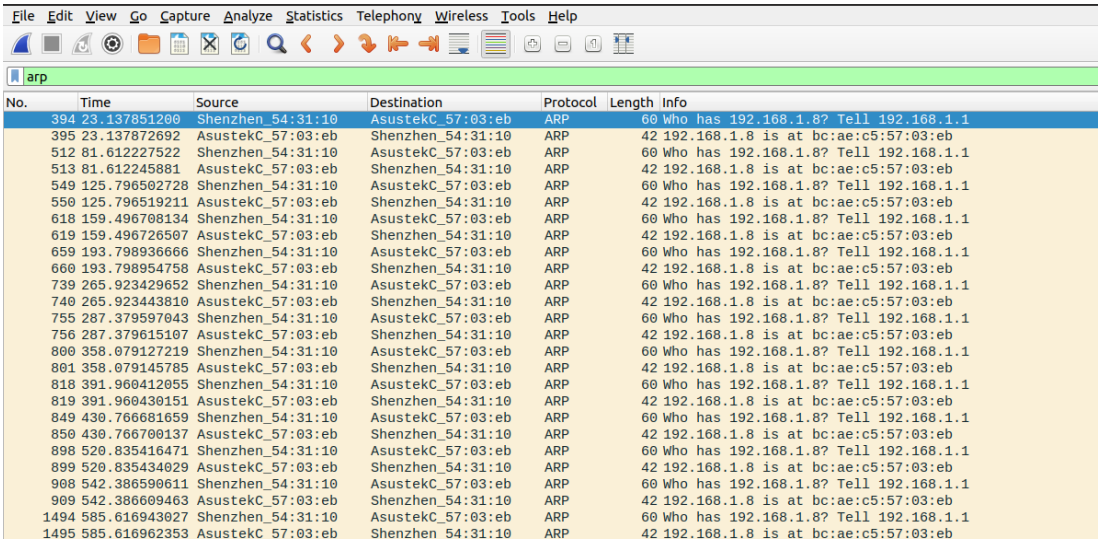

Figura 3: Captura de paquetes ARP

3.3. Primera consigna: Modelando tráfico usando fuentes de información

Sean $p_1..p_n$ las tramas de capa 2 que se capturan en una red local. Se pueden modelar las tramas capturadas como una fuente de información de memoria nula S1 = $\{s_1, s_2, ..., s_q\}$, donde cada s_i está formado por la combinación entre el tipo de destino de la trama (*Unicast* o *Broadcast*) y el protocolo de la capa inmediata superior encapsulado en la misma. Por ejemplo, $s_i =< Broadcast, ARP >$. A continuación se presenta un código Python de ejemplo para capturar de paquetes y calcula las probabilidades de cada uno de los símbolos de la fuente de información S1 en una red con la que tenemos conexión:

```
#!/usr/bin/python

from scapy.all import *

S1 = {}

def mostrar_fuente(S):
    N = sum(S.values())
    simbolos = sorted(S.iteritems(), key=lambda x: -x[1])
    print "\n".join([ "%s : %.5f" % (d,k/N) for d,k in simbolos ])
    print

def callback(pkt):
    if pkt.haslayer(Ether):
        dire = "BROADCAST" if pkt[Ether].dst=="ff:ff:ff:ff:ff:ff" else "UNICAST"
        proto = pkt[Ether].type # El campo type del frame tiene el protocolo
        s_i = (dire, proto) # Aca se define el simbolo de la fuente
        if s_i not in S1: S1[s_i] = 0.0
        S1[s_i] += 1.0

    mostrar_fuente(S1)

sniff(prn=callback)
```

Si ejecutamos el código anterior y esperamos a capturar suficientes paquetes podemos obtener una salida como la siguiente:

```
('UNICAST', 34525) : 0.48902
('UNICAST', 2048) : 0.42611
('UNICAST', 2054) : 0.06825
('BROADCAST', 2054) : 0.01424
('UNICAST', 34958) : 0.00237
```

En dicha salida se muestran los símbolos de la fuente de información S1 y sus respectivas probabilidades. Cada símbolo es una tupla que indica si se trata de paquetes Broadcast o Unicast y el protocolo de capa superior al que corresponde cada paquete capturado. Por ejemplo, 2048 es IP, de los que se capturaron sólo paquete Broadcast, 2054 es ARP para los cuales hay tanto Broadcast como Unicast, etc.

Probar la captura de paquetes usando el código presentado anteriormente (notar que debe ser ejecutado con permisos de superusuario), que captura tráfico en una red local y muestra representativamente la fuente modelada S1. La salida consiste en una tabla que muestra la probabilidad de cada símbolo de la fuente. Luego, extender el código para que calcule la información de cada símbolo y la entropía de la fuente. Finalmente, realizar una captura de tráfico utilizando el código extendido anteriormente. La captura deben ser lo más extensa posibles (por ejemplo de más de 10.000 tramas).

3.4. Segunda consigna: Experimentación e Informe

Utilizando la herramienta desarrollada, que permite analizar el tráfico en una red usando fuentes de información, realizar experimentos para analizar una red distinta por cada integrante del grupo.

El informe debe seguir la siguiente estructura, intentando cumplir con los límites de palabras sugeridos:

- **Introducción (máximo 200 palabras):** Breve explicación de los experimentos que se van a realizar.

- **Métodos y condiciones de los experimentos (máximo 400 palabras):** Explicación del código implementado y descripciones de las redes sobre las cuales se realizan las capturas. Se debe detallar en la descripción de la red el tipo de tecnología y tamaño y si las características de la muestra -tamaño, horario, día de la semana, etc.- y la justificación de la elección del modelo de la fuente S2.
- **Resultados de los experimentos (máximo 600 palabras):** En esta sección deben presentarse figuras y/o tablas que muestren de manera integral los resultados observados. **Sugerencias:**
  - Porcentaje de tráfico Broadcast/Unicast sobre el tráfico total.
  - Porcentaje de aparición de cada protocolo encontrado.
  - Entropía de cada red analizada.
  - Cantidad de información de cada símbolo comparado con la entropía de la red.
- **Conclusiones (máximo 200 palabras):** Breve reseña que sintetize las principales dificultades y descubrimientos.

A continuación se sugieren preguntas que se pueden intentar responder una vez realizadas las capturas. No hace falta transcribirlas en el informe y se valorará significativamente el planteo de nuevas preguntas.

- ¿Considera que las muestras obtenidas analizadas son representativas del comportamiento general de la red?
- ¿Hay alguna relación entre la entropía de las redes y alguna característica de las mismas (ej.: tamaño, tecnología, etc)?
- ¿Considera significativa la cantidad de tráfico broadcast sobre el tráfico total?
- ¿Cuál es la función de cada uno de los protocolos encontrados?
- ¿Cuáles son protocolos de control y cuáles transportan datos de usuario?
- ¿En alguna red la entropía de la fuente alcanza la entropía máxima teórica?
- ¿Ha encontrado protocolos no esperados? ¿Puede describirlos?

3.5. Tercera consigna (OPCIONAL): Nodos distinguidos

Extender el informe con una propuesta de modelo de fuente de información de memoria nula S2 con el objeto de *distinguir* los hosts de cada red. La distinción de S2 debe estar basada únicamente en las direcciones IP dentro de los paquetes ARP [3]. El criterio para el modelado lo deberá establecer cada grupo utilizando las herramientas teóricas provistas por la teoría de la información. Se puede pensar que un símbolo es *distinguido* cuando sobresale del resto en términos de la información que provee. Algunas preguntas que se pueden intentar responder serían:

- ¿La entropía de la fuente es máxima? ¿Qué sugiere esto acerca de la red?
- ¿Se pueden *distinguir* nodos? ¿Se les puede adjudicar alguna función específica?
- ¿Hay evidencia parcial que sugiera que algún nodo funciona de forma anómala y/o no esperada?
- ¿Existe una correspondencia entre lo que se conoce de la red y los nodos distinguidos detectados por la herramienta?
- ¿Ha encontrado paquetes ARP no esperados? ¿Se puede determinar para que sirven?

Referencias

[1] Wireshark http://www.wireshark.org

[2] Scapy http://www.secdev.org/projects/scapy

[3] RFC 826 (ARP) http://tools.ietf.org/html/rfc826

[4] RFC 2131 (DHCP) http://tools.ietf.org/html/rfc2131