Laboratorio: scapy

Cátedra de Redes y Sistemas Distribuidos
Original 2011, Rafael Carrascosa

Descargo de responsabilidad

En este laboratorio se va a demostrar cómo capturar contenido enviado por red a la computadora en la que estamos trabajando. Si se conservan copias de archivos cuyo contenido esta protegido por derecho de autor se esta comentiendo un delito a menos que uno cuente con la autorización del tenedor de estos derechos.

Asimismo, si se captura y observa información dirigida a *otra* computadora (lo cual no se explica en este laboratorio) también se corre el riesgo de estar incurriendo en un delito puesto que, por ejemplo, leer correspondencia ajena esta penado con entre 15 días y 6 meses de prisión en Argentina.

Ni la cátedra de Redes y Sistemas Distribuídos, ni FaMAF, ni la UNC se hacen responsables del uso inapropiado del material dado para estas clases.

Objetivos

- Aprender a usar una **herramienta de análisis de red.**
- Entender mínimamente el **funcionamiento de TCP**.
- Poder **capturar información** enviada en una conexión TCP.

Motivación

Existen numerosas aplicaciónes web que de alguna forma nos *esconden* u *ofuscan* su funcionamiento interno. Un ejemplo de una aplicación web *opaca* que recibe datos "a nuestras espaldas" es el reproductor de música de last.fm, o el viejo reproductor de grooveshark, porque de alguna forma reciben de un servidor la música que escuchamos pero el mecanísmo con el cúal lo hacen es mantenido oculto para reforzar la protección de los derechos de autor.

Resulta interesante y educativo poder observar y entender qué datos son transmitidos hacia nuestra computadora por alguna aplicación web opaca (cómo por ejemplo aquellas escritas en flash) teniendo la opción de conservar los datos recibidos por estas aplicaciones(siempre y cuando no se violen derechos de autor ni licencias de uso).

Introducción

TCP es *el* protocolo sobre el cual la mayor parte de las aplicaciónes que dependen de internet están construidas. Entender su funcionamiento no es solo un buen entrenamiento en la prácticas comúnes del diseño de protocolos sino que además puede resultar útil para capturar información que se transmite por la red.

Cuando se esta desarrollando un nuevo protocolo de red, o cuando se están buscando fallas de seguridad en un protocolo es de enorme utilidad poder observar (desde el punto de vista de un tercero) la información que intercambia el protocolo a travez de la red. Las herramientas que permiten observar este tipo de actividad se llaman **herramientas de análisis de red**.

De entre estas la herramienta más básica es el **sniffer**, un programa que permite capturar los paquetes de datos tal cual son enviados por la placa de red. Los datos capturados por este programa pueden ser luego alimentados a otros que cumplen funciones de analisis específicas (estadísticas, reconstrucción de contenido, busqueda de ataques, etc.).

En este laboratorio presentaremos a un sniffer clásico de varios *nix: **tcpdump**. Además utilizaremos **scapy**, una biblioteca python para el análisis de paquetes capturados con tcpdump.

Tcpdump

Tcpdump lee paquetes de una interfaz de red y los puede presentarlos al usuario de varias formas. Por ejemplo:

```
rafael@tiber:~$ sudo tcpdump -i wlan2 -n -q
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on wlan2, link-type EN10MB (Ethernet), capture size 96 bytes
12:09:34.478995 IP 192.168.1.108.22 > 192.168.1.110.34657: tcp 192
12:09:34.480783 IP 192.168.1.110.34657 > 192.168.1.108.22: tcp 0
12:09:34.488981 IP 192.168.1.108.22 > 192.168.1.110.34657: tcp 176
12:09:34.490766 IP 192.168.1.110.34657 > 192.168.1.108.22: tcp 0
12:09:34.495459 IP 122.107.108.110.16191 > 192.168.1.108.34909: tcp 0
12:09:34.495480 IP 192.168.1.108.34909 > 122.107.108.110.16191: tcp 1448
...
12:09:34.894421 IP 192.168.1.108.22 > 192.168.1.110.34657: tcp 240
^C
75 packets captured
75 packets received by filter
0 packets dropped by kernel
rafael@tiber:~$
```

Aqui se muestra a tcpdump leyendo paquetes desde wlan2 (una interfaz wireless) y mostrando un resumen por salida estandar. Notar que tcpdump necesita ejecutarse con privilegios de root.

```
rufus@gavilan:~$ sudo tcpdump -i eth1 -n -s 0 -w textfile.pcap
tcpdump: listening on eth1, link-type EN10MB (Ethernet), capture size 65535 bytes
^C179 packets captured
179 packets received by filter
0 packets dropped by kernel
rufus@gavilan:~$
```

Este otro ejemplo hace algo más interesante: Guarda una copia de cada paquete que paso por la interfaz wlan2 en el archivo textfile.pcap. Pcap es un formato de archivo más o menos estandar para almacenar **p**aquetes **cap**turados.

Scapy

Scapy es una biblioteca para python que permite (entre otras cosas) manipular paquetes capturados en archivos pcap. En el kickstart se entrega un pequeño código (`tryme.py`) que carga en el interprete interactivo dos listas:

- **`xs`**: Una lista con todos los paquetes que hay en textfile.pcap

- `ys`: Una lista con todos los paquetes de xs que son TCP y que tienen un payload

```
rufus@gavilan:~/trabajo/docencia/redes/Lab4/kickstart$ ipython tryme.py
Python 2.6.5 (r265:79063, Apr 16 2010, 13:57:41)
Type "copyright", "credits" or "license" for more information.

IPython 0.10 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object'. ?object also works, ?? prints more.

In [1]: len(xs)
Out[1]: 179

In [2]: len(ys)
Out[2]: 3

In [3]: ys[0].show()
###[ Ethernet ]###
  dst= 94:0c:6d:bd:3e:f6
  src= 00:18:f3:54:c5:1c
  type= 0x800
###[ IP ]###
     version= 4L
     ihl= 5L
     tos= 0x0
     len= 195
     id= 7725
     flags= DF
     frag= 0L
     ttl= 64
     proto= tcp
     chksum= 0x80aa
     src= 192.168.1.110
     dst= 200.16.17.55
     options= ''
###[ TCP ]###
        sport= 49543
        dport= www
        seq= 802911102
        ack= 846891830
        dataofs= 8L
        reserved= 0L
        flags= PA
        window= 92
        chksum= 0x35d5
        urgptr= 0
        options= [('NOP', None), ('NOP', None), ('Timestamp', (1076572, 155969451))]
###[ Raw ]###
           load= 'GET /~carrascosa/loremipsum.txt HTTP/1.0\r\nUser-Agent: Wget/1.12 (linux-gnu)\r\nAccept: */*\r\nHost: cs.famaf.unc.edu.ar\r\nConnection: Keep-Alive\r\n\r\n'
```

Como se puede ver, solo hay 3 paquetes con payload y se pertenecen a un pedido HTTP de un archivo de texto: `loremipsum.txt`

Tarea

Se deberá modificar un programa provisto (en el kickstart) de manera que pueda **capturar archivos** enviados por protocolo HTTP sobre TCP/IP.

El programa implementado tiene que (al menos) poder guardar archivos de tipo `.jpg` (imagen), `.ogg` (audio) y `.ogg` (video) enviados en una conexión HTTP sobre TCP a partir de los paquetes capturados en un archivo pcap (este archivo lo proveemos nosotros, ver en ayudas). Además debería ser capaz de capturar cualquier archivo que tenga un tipo MIME declarado en el encabezado del pedido HTTP a partir de cualquier archivo de captura.

Para ello deberán:

- Completar la clase Connection_status desde el esqueleto (docstring y prototipos) para que pueda indentificar cuando una conexión fue establecida y cuando una conexión se cerro.
- Completar la clase Reassembler desde el esqueleto (docstring y prototipos) para que pueda reconstruir el stream de datos enviado en una conexión TCP.

Reensamblar TCP/HTTP correctamente todas las veces es una tarea compleja, por lo que para este laboratorio **no** es necesario que su programa pueda capturar todos los archivos todas las veces. Es suficiente con que su programa se comporte bien en las situaciones más comunes (este no es software crítico).

Ayudas

- Jugar con `tryme.py` un rato les va a resultar muy provechoso
- El RFC que especifica TCP:
  - La seccion 4.3.6 especifica 7 estados en los que se puede encontrar una conexión y como se transiciona entre ellos.
  - Figura 1 en la página 48, transiciones entre estados.
  - Figura 2.2b-1 en la página 51, cómo procesar la entrada.
  - Figura 2.3-1 en la página 52, cómo reensamblar el payload.
- El kickstart es ejecutable, tiene las clases con implementaciones de jueguete. Cada paquete se considera una conexión propia y su payload se considera el payload de toda la conexión, o sea que va a haber un archivo por cada payload HTTP OK en el archivo de capturas. Esto les permite ver cuáles archivos se transmitieron en una captura.
- Aquí hay un archivo pcap de ejemplo conteniendo descargas de `.jpg`, `.ogg` y `.ogg`.

Requisitos del código a entregar

- La fecha máxima de entrega es el día míercoles de 25 de Mayo, a las 23:59.
- Las entregas serán a través del repositorio SVN provisto por la Facultad para la Cátedra.
- Junto con el código, se deberá entregar un **brevisimo informe** en el cual se explique los casos especiales en los que no funciona el código, las dificultades con las que se encontraron, y cómo las resolvieron. El formato de dicho informe queda a elección de ustedes, siempre y cuando sea un formato libre.
- Se deberá mantener el estilo de codificación PEP8.
- También se exigirán las metodologías de programación defensiva aprendidas en las materias correlativas.
- El trabajo es grupal. Todos los integrantes del grupo deberán poder explicar el código presentado.
- No está permitido compartir código entre grupos.