Universidad Nacional Autónoma de México
Facultad de Ciencias

Alumnos:
Axel Eduardo Becerril Nájera
Erick Carmona Jiménez

Materia: Análisis Forense en Redes de Computadoras

Profesora: María Teresa Canseco Rodríguez

Analizador de Protocolos

I.-Descripción

Un analizador de protocolos es un programa que por medio de instrucciones captura tod el tráfico de red, todos los paquetes los clasifica según el protocolo al que pertenezcan, para clasificarlos se hace uso de información como su la cabecera de cada protocolo y de cada elemento que la compone.
En clase hemos visto las cabeceras de cada protocolo asi que asumiremos que ya las conocen.

El programa está hecho en el lenguaje de programación Python, a su vez se respeta el requerimiento acerca de usar libpcap para la elaboración del programa.

Para facilitar el acceso al programa, se hizo un repositorio en
http://github.com
Dentro del cual se encuentra el programa, y los documentos.
https://github.com/eseca/afrc2014-1

¿Qué es Pcapy?

Pcapy es un modulo de Python que se conecta con la biblioteca libpcap, Pcapy permite hacer scripts para capturar paquetes de red. Y nos permite utilizar funciones definidas dentro de libpcap, Por ésta razón es posible hacer el analizador en Python.

II.-Funcionamiento

Una vez que ya hemos descargado el programa, debemos descomprimir la carpeta, Inmediatamente identificaremos el archivo afrc.py.

Cabe señalar que el programa recibe argumentos a la hora de ejecutarlo.

Primero debemos estar como root, para asi poder ejecutar el programa.

Usando la terminal, nos situamos en la carpeta donde este el archivo afrc.py. Una vez situados ahí, escribimos

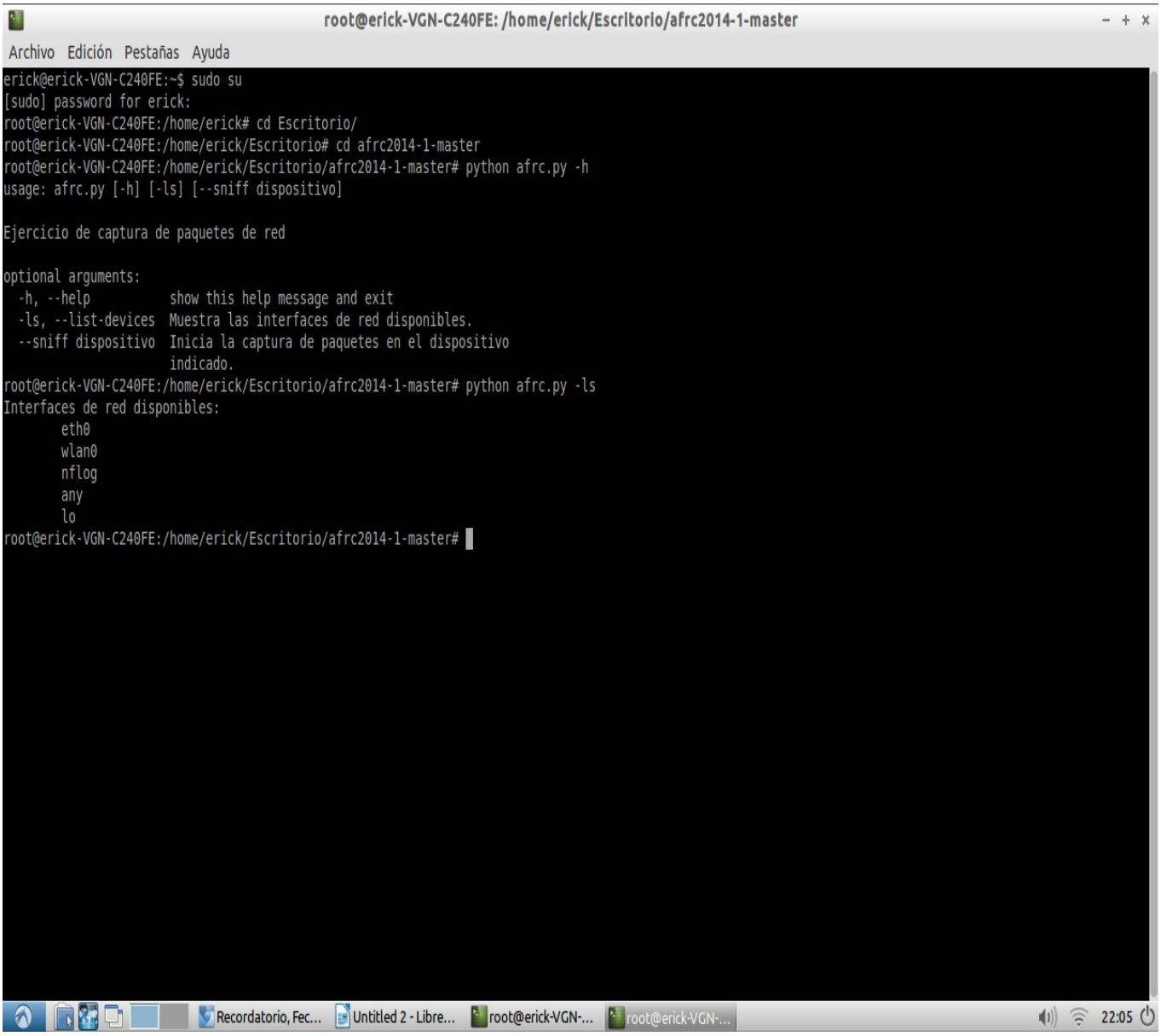
```
python  afrc.py  -h
```

Este comando nos mostrara los parametros que podemos pasarle a afrc.py en terminal.

A continuación se describen los parametros.

-h : muestra los parametros que podemos usar
-ls : muestra la lista de interfaces de res disponibles
--sniff dispositivo : inicia la captura de paquetes en el dispositivo indicado, donde los dispositivos pueden ser eth0,wlan0,nflog,any,lo.

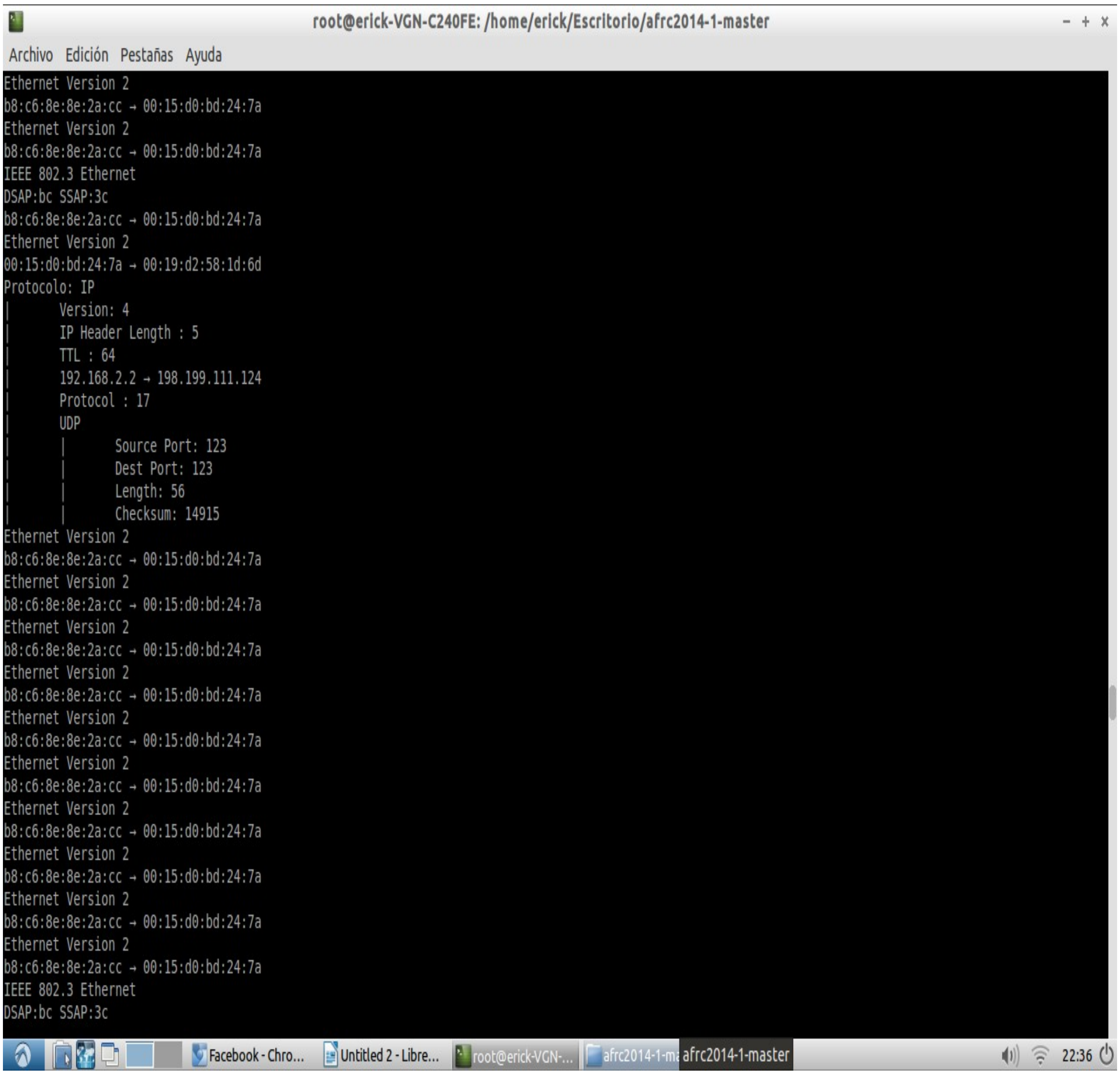
Eth0 si estamos conectados a cable de red
Wlan0 si estamos coenctados sin cable de red.



Entonces una vez que tenemos claro los parametros que podemos usar, procedemos a escribir

python afrc.py –sniff eth0

Con dicho comando empezara a capturar el tráfico de red, pero lo va a ir capturando al mismo tiempo que lo va clasificando, a su vez irá mostrando toda la información respecto a cada paquete.



Nos muestra información de cada paquete, nos muestra direcciones MAC, por ejemplo analizemos el siguiente paquete capturado por el programa.

Protocolo: IP
| Version: 4
| IP Header Length : 5
| TTL : 64
| 192.168.2.2 → 198.199.111.124
| Protocol : 17
| UDP
| | Source Port: 123
| | Dest Port: 123
| | Length: 56
| | Checksum: 14915

Podemos ver que muestra el protocolo asociado al paquete, la versión, la longitud de la cabecera , el tiempo de viaje redondo (ttl), la Ip tanto origen como destino, el puerto origen y el puerto destino, el checksum,etc. Donde estos datos van siendo clasificados de acuerdo a lo descrito dentro del código.

III.- Detalles

Algunos detalles, fueron al hacer la representación de los paquetes, por ejemplo con la función “unpack”.

Con el modulo de Python “Struct” podemos hacer uso de la función unpack.
Este módulo nos permite hacer conversiones entre valores de Python y estructuras de C representadas como cadenas de Python.
Struct tiene varias funciones y excepciones que podemos usar, de entre ella está, unpack, la firma del método es :

unpack(fmt, string)

la cual desempaca la cadena de acuerdo con el formato dado (fmt es el formato, string es la cadena).
Detalles como saber el formato exacto fueron problemas lo causaron algunas confusiones.

Cabe mencionar que las funciones que se tuvieron que programar son:
def devs: la cual muestra las interfaces de red disponibles
def sniff: con la que usa un ciclo while(true) para capturar todo el tráfico hasta que se le de control-c.
def pretty-mac : la cual da formato a la dirección MAC
def parse: esta es sobre todo la función más importante, pues con ella , clasificamos los paquetes de acuerdo al protocolo al que pertenecen.
def main: la cual muestra las opciones de los diferentes argumentos que podemos pasarle al programa cuando vamos a ejecutarlo

IV.-Conclusiones

Este programa, nos enseña mucho acerca de como se elabora un analizador de protocolos sencillo, puesto que otros analizadores como wireshark integran muchas más opciones, y por ende son más programas más pulidos, tanto a nivel gráfico como a nivel del código usado.

Además checar e informarnos de que no sólo están lenguajes como C o Java para poder hacer un proyecto. Python en este caso ayudo mucho, además es bastante más compacto y limpio en el sentido de que es sencillo identificar que se esta haciendo, y que con comentarios la comprensión del código se vuelve más digerible, y con un reporte como éste esperamos que sea aún más sencillo, poder comprender lo que se hizo.

Además Python viene ya instalado dentro de los sistemas linux, entonces es una comodidad más.

V.- Bibliografía

http://docs.python.org/2/library/struct.html
//Documentacion del módulo Struct

http://www.python.org/
//página oficial de Python