Detectie Binare In Traficul De Retea

libpcap permite capturarea si trimiterea de pachete pe retea,este o librarie opern source

Pentru instalare:

sudo apt-get install libpcap-dev

Odata ce a fost instalat , puteti compila programe pcap cu urmatoarea comanda:

gcc <filename> -lpcap ( programul trebuie rulat cu drepturi de root)

In proiectul nostru , trebuie sa interceptam traficul , sa verificam daca fisierul este binar ,verificand payload-ul si daca fisierul este ok sa il readucem in starea initiala.

The first thing to understand is the general layout of a pcap sniffer. The flow of code is as follows:

1. We begin by determining which interface we want to sniff on. In Linux this may be something like eth0, in BSD it may be xl1, etc. We can either define this device in a string, or we can ask pcap to provide us with the name of an interface that will do the job.
2. Initialize pcap. This is where we actually tell pcap what device we are sniffing on. We can, if we want to, sniff on multiple devices. How do we differentiate between them? Using file handles. Just like opening a file for reading or writing, we must name our sniffing "session" so we can tell it apart from other such sessions.
3. In the event that we only want to sniff specific traffic (e.g.: only TCP/IP packets, only packets going to port 23, etc) we must create a rule set, "compile" it, and apply it. This is a three phase process, all of which is closely related. The rule set is kept in a string, and is converted into a format that pcap can read (hence compiling it.) The compilation is actually just done by calling a function within our program; it does not involve the use of an external application. Then we tell pcap to apply it to whichever session we wish for it to filter.
4. Finally, we tell pcap to enter it's primary execution loop. In this state, pcap waits until it has received however many packets we want it to. Every time it gets a new packet in, it calls another function that we have already defined. The function that it calls can do anything we want; it can dissect the packet and print it to the user, it can save it in a file, or it can do nothing at all.
5. After our sniffing needs are satisfied, we close our session and are complete.

This is actually a very simple process. Five steps total, one of which is optional (step 3, in case you were wondering.) Let's take a look at each of the steps and how to implement them.



