CS 577 Cybersecurity Lab

Lab 8 – due 11/19/15 11:59pm

Subject: Packet sniffing and analysis

All network intrusion detection systems start with sniffing network packets and then analyzing the captured packets using the methods we went over in the class. In this lab, you are going to develop a rudimentary packet sniffer that can apply a set of simple analyses on the captured packets and report some data back to the administrator.

A popular library for sniffing network packets is *libpcap*. To make things easier for you, you are not going to actually sniff a live network interface, but you are going to be using the library to read packets from a file, where packets were previously captured using *pcap* and the *tcpdump* utility. The main operations you should learn to perform through the library include obtaining the various fields of the encapsulated protocols (e.g., Ethernet, IP, and TCP) and selecting packets based on them, correlating packets (e.g., which TCP packets belong to the same stream), and examining the content of the packets (aka deep-packet inspection).

Your sniffer should support two command line arguments. The first should specify the pcap file to open, and the optional second the analysis module to use. When processing finishes your program should print the number of processed packets per protocol (IP, TCP, UDP), the time it took to finish processing the packets, and then, if an analysis module was specified, it can print its own information.

You do not have to do any analyses on Ethernet frames, so you need to take the Ethernet frames in the pcap file, extract IP packets from them, and support at least the TCP and UDP protocols (encapsulated) over IP .

It's really pretty simple. Just take the ethernet frames that you get from pcap and extract the IP packets from them, reassembling any that were fragmented. Then, reorder the TCP segments from the IP packets, according to the sequence numbers, paying attention that you discard any duplicate data. Then, process the stream as an HTTP stream. Of course, HTTP doesn't come in packets; it is an application layer protocol, but I'm sure this will be obvious once you've done all this other work. Pay attention as you do all these things to checksum the IP headers and TCP segments, to ensure that your data is correct. Also, if pcap happens to miss any packets, make sure you deal with this appropriately

Pcap file:

We are going to use the following file http://download.netresec.com/pcap/maccdc-2012/maccdc2012_00016.pcap.gz, from http://www.netresec.com/?page=MACCDC

Tutorials:

http://www.tcpdump.org/pcap.htm

Deliverables

A packet sniffer and analysis tool with the following functionality [80%]:

1. **Base sniffer [30%]**
   Open and process a pcap file, printing packet statistics.
2. **Context search module [25%]**
   This module receives a third argument, which specifies a sequence of bytes that should be searched in the logged packets. E.g.,
   `sniffer data.pcap search "0A EB FF 00"`
   When processing finishes, it should print how many packets matched the given byte string, along with the connections they belong. For example:
   `(TCP|UDP) source_ip:source_tcp_port dest_ip:dest_tcp_port packets_number`
   or
   `IP source_ip dest_ip packets number`
   for other protocols.
3. **Encrypted flows module [25%]**
   This module locates all TCP connections/flows that are using SSL or TLS in the captured file. When processing finishes print out all the TCP flows that were found using the following format:
   `source_ip:source_tcp_port dest_ip:dest_tcp_port packets_number`
   You can reconstruct TCP streams before looking for indication of SSL/TLS setup (obviously, you will only see encrypted data after the channel is set up). You may also be able to accomplish your goal, without doing so (maybe as an easier first step). What could go wrong in this case?
   To reconstruct TCP streams, you need to reorder the TCP segments contained in IP packets according to their sequence numbers. You need to be careful to discard duplicate packets (again using the packet sequence number). The application protocol (SSL/TLS in this case) is in the reassembled data stream contained in the TCP packets.

Include a report.txt file explaining how sniffer works and identify any limitations it may have [20%].

Submission information

You can develop for this lab on the SEED VM that include the development files for libpcap. Submit all your files as a tar.gz archive through Canvas.