**Assignment 2: Passive network monitoring application**

SBU ID: 111481387

This is a passive network monitoring application implemented in C language that uses the libpcap library. The program captures the network traffic in the promiscuous mode and also supports reading from a .pcap trace file and prints record for each packet in its standard output.

**Features supported:**

1. -i option: This is used to specify the device we want to sniff. It could be the default interface, in which case we need not use this option, or a user specified interface.

2. -r option : This reads a pcap file. This option is used when we don’t want to sniff an interface but read from a .pcap trace file and prints the packets.

3. -s option : This option looks for a string pattern in the payload and only prints the packet details if a string match found.

4. BPF filter: This is a filter used to capture a subset of the traffic.

**Implementation details:**

* The program starts with scanning the command line options using the getopt function, which parses each of the arguments mentioned above and their respective arguments and stores in them in respective variables which are handled later in the program.
* The program takes input either from the file or the interface. The file is obtained from the -r option argument and the interface through the -i option. The interface if not specified, is the default interface. In case of default interface, we need look up for the device using the pcap\_lookupdev() function.
* Once we know the input source, we get the handle for the device(file or the interface) using the pcap\_open\_offline() in case of file and pcap\_open\_live() in case of the interface.
* We compile the filter expression specified by the user using the pcap\_compile() function and then apply this compiled filter using the pcap\_setfilter() function.
* The pcap\_loop function sets the call back function got\_packet
* got\_packet() is the callback function that gets called for each incoming packet. This function dissects the packet to get its IP header and then extracts the transport layer headers and processes the payload .
* Once the pointer to payload is obtained, the program checks if the -s option is set. If yes, then the program checks for the string argument in the obtained payload, and prints the packet only if a match is found using the print\_payload() function.

**How to execute the program:**

1. make

2. ./mydump [-i interface] [-r file] [-s string] expression

**Contents of the tarball:**

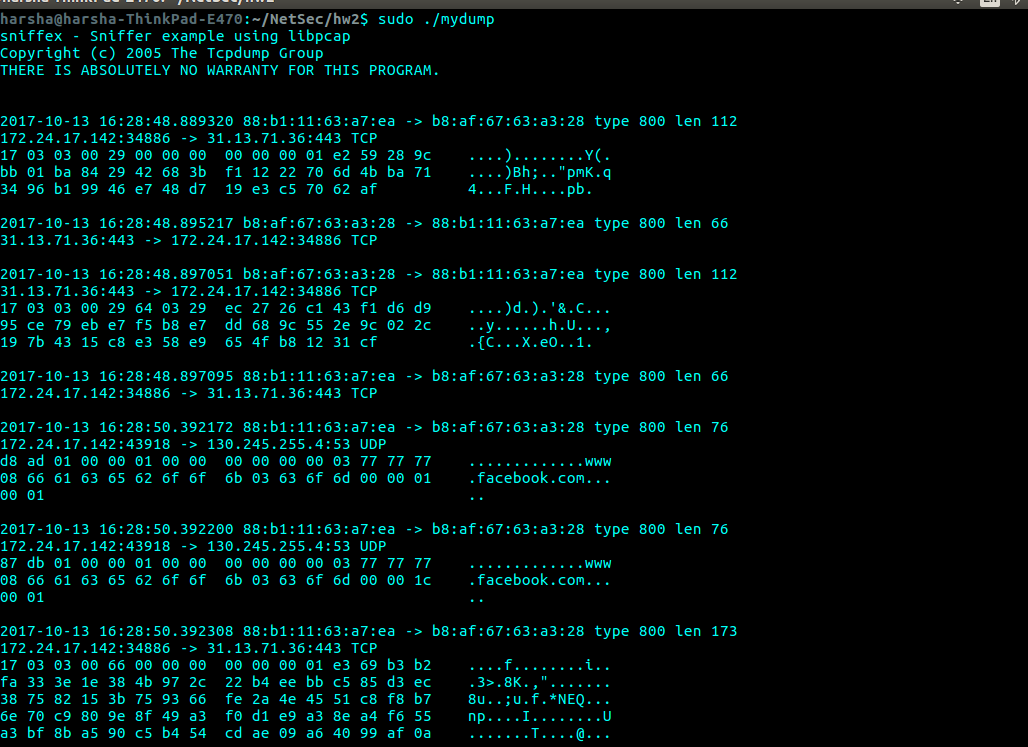
1. mydump.c

2. makefile

3. Report “hw2.docx”

**Sample outputs of the program:**

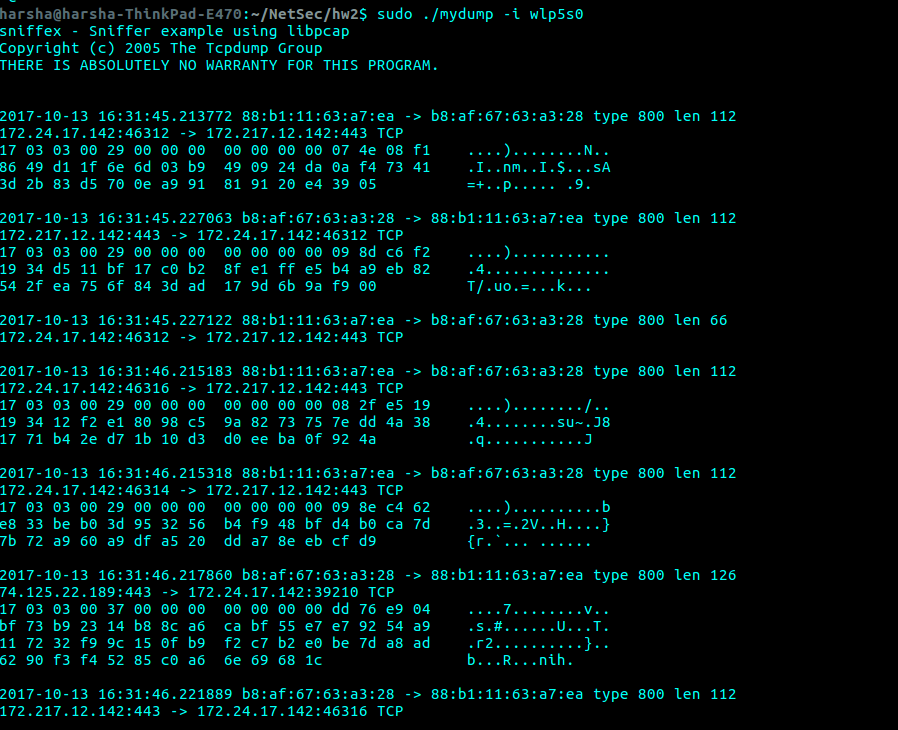
1. Capturing packets on the default interface



sudo ./mydump

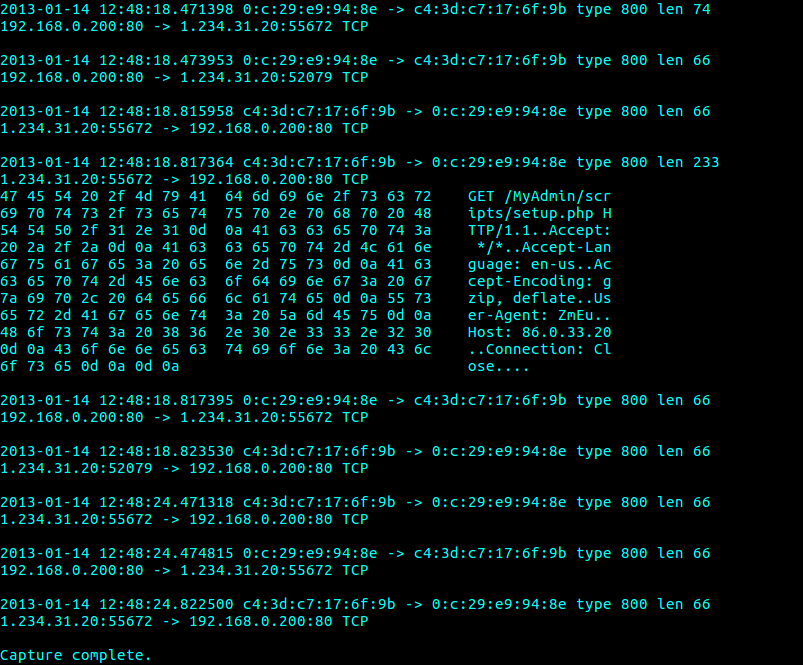
2. Capturing packets on the user specified interface

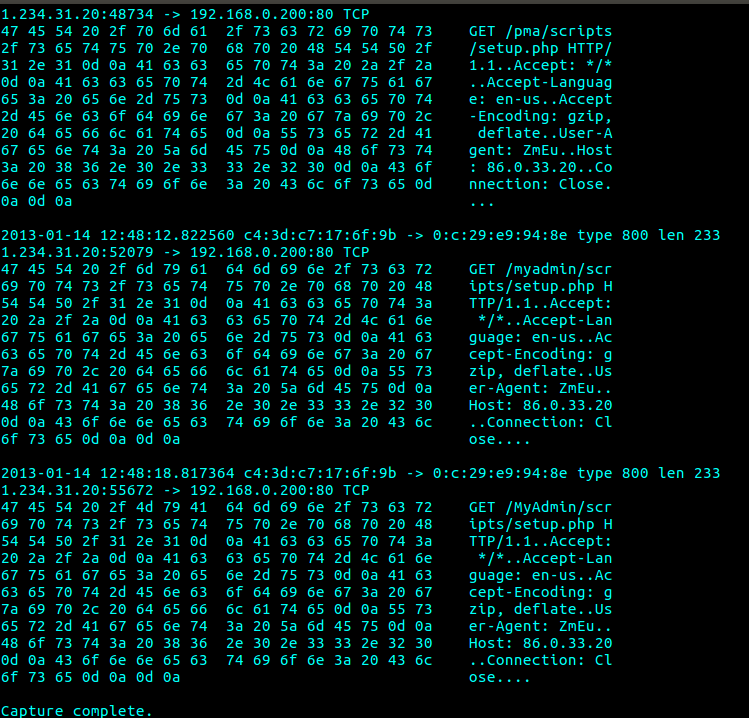
sudo ./mydump -i wlp5s0



3. Capturing packets on the file interface with tcp as the bpf filter

sudo ./mydump -r hw1.pcap tcp



4 Capturing the packets with TCP as the BPF filter and “GET” as the string expression

sudo ./mydump -r hw1.pcap tcp -s GET