To tackle this assignment, we need to write a C program that reads a pcap file, parses SMBv2 packets, extracts file attachments, and gathers metadata. Here's a step-by-step approach:

**Steps to Follow:**

1. **Setup the Project**:
   * Create a GitHub repository for version control.
   * Set up a basic C project structure.
2. **Read the PCAP File**:
   * Use a library like libpcap to read the pcap file.
3. **Parse Ethernet, IP, and TCP Headers**:
   * Extract the necessary information from the Ethernet, IP, and TCP headers.
4. **Parse SMBv2 Packets**:
   * Identify SMBv2 packets within the TCP payload.
   * Handle SMBv2 Write and Read Requests and Responses.
5. **Extract Attachments and Metadata**:
   * Extract files from the SMB Write Request/Response packets.
   * Extract metadata such as file name, file size, source IP, source port, destination IP, and destination port.
6. **Output the Extracted Files and Metadata**:
   * Save the extracted files.
   * Print or save the metadata in a structured format.
7. **Dockerize the Application**:
   * Create a Dockerfile to containerize the application for ease of deployment.

**Code Example:**

Below is an example structure of the C program:

c

Copy code

#include <pcap.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <netinet/ip.h>

#include <netinet/tcp.h>

// Ethernet header length

#define SIZE\_ETHERNET 14

// Define constants for SMBv2

#define SMB2\_HEADER\_LENGTH 64

#define SMB2\_WRITE\_REQUEST 0x0009

#define SMB2\_READ\_REQUEST 0x0008

#define SMB2\_WRITE\_RESPONSE 0x0009

#define SMB2\_READ\_RESPONSE 0x0008

// Metadata structure

struct metadata {

char filename[256];

uint32\_t filesize;

char src\_ip[16];

uint16\_t src\_port;

char dst\_ip[16];

uint16\_t dst\_port;

};

// Function prototypes

void parse\_pcap(const char \*filename);

void handle\_packet(const struct pcap\_pkthdr \*header, const u\_char \*packet);

int main(int argc, char \*argv[]) {

if (argc != 2) {

fprintf(stderr, "Usage: %s <pcap file>\n", argv[0]);

exit(EXIT\_FAILURE);

}

parse\_pcap(argv[1]);

return 0;

}

void parse\_pcap(const char \*filename) {

char errbuf[PCAP\_ERRBUF\_SIZE];

pcap\_t \*handle = pcap\_open\_offline(filename, errbuf);

if (handle == NULL) {

fprintf(stderr, "Could not open pcap file %s: %s\n", filename, errbuf);

exit(EXIT\_FAILURE);

}

struct pcap\_pkthdr header;

const u\_char \*packet;

while ((packet = pcap\_next(handle, &header)) != NULL) {

handle\_packet(&header, packet);

}

pcap\_close(handle);

}

void handle\_packet(const struct pcap\_pkthdr \*header, const u\_char \*packet) {

// Parse Ethernet header

struct ether\_header \*eth\_header = (struct ether\_header \*)packet;

if (ntohs(eth\_header->ether\_type) != ETHERTYPE\_IP) {

return;

}

// Parse IP header

struct ip \*ip\_header = (struct ip \*)(packet + SIZE\_ETHERNET);

int ip\_header\_length = ip\_header->ip\_hl \* 4;

// Parse TCP header

struct tcphdr \*tcp\_header = (struct tcphdr \*)(packet + SIZE\_ETHERNET + ip\_header\_length);

int tcp\_header\_length = tcp\_header->th\_off \* 4;

// Parse SMBv2 packet

const u\_char \*payload = packet + SIZE\_ETHERNET + ip\_header\_length + tcp\_header\_length;

int payload\_length = header->caplen - (SIZE\_ETHERNET + ip\_header\_length + tcp\_header\_length);

// Handle SMBv2 packets

if (payload\_length >= SMB2\_HEADER\_LENGTH) {

// Extract information and process based on the SMB2 command type

uint16\_t command = ntohs(\*(uint16\_t \*)(payload + 12));

if (command == SMB2\_WRITE\_REQUEST || command == SMB2\_WRITE\_RESPONSE) {

// Extract file attachment and metadata

} else if (command == SMB2\_READ\_REQUEST || command == SMB2\_READ\_RESPONSE) {

// Extract file attachment and metadata

}

}

}

**Dockerfile Example:**

dockerfile

Copy code

FROM debian:latest

RUN apt-get update && \

apt-get install -y libpcap-dev gcc make

WORKDIR /app

COPY . .

RUN make

CMD ["./network\_security"]

**Steps to Test:**

1. **Clone the Repository**:

bash

Copy code

git clone <repository\_url>

cd <repository>

1. **Build the Project**:

bash

Copy code

make

1. **Run the Program**:

bash

Copy code

./network\_security smb.pcap

1. **Build and Run with Docker**:

bash

Copy code

docker build -t network\_security .

docker run -v $(pwd):/data network\_security /data/smb.pcap

This structure provides a robust foundation to parse and handle SMBv2 packets, extract files, and gather metadata. Further development will involve detailed parsing of SMBv2 commands and managing file write/read operations.