Trabalho 2 de laboratório de redes

Douglas Silva Martins

RGA: 201711901010

Resumo

Trabalho consiste na leitura de um arquivo pcap e gravação de um também, nesse arquivo deve ter cabeçalhos de protocolos de comunicação da rede , a primeira coisa a se fazer e um menu de opções para escolher entre ler ou gravar um arquivo pcap sendo foi usado a comparação dos elementos passado pelo argv para ver qual o comando recebido no argv[1] se for -r leitura de arquivo e se -w gravação de arquivo.

Comando de execução leitura de arquivos pcap

sudo ./executavel –r nomeArquivo.pcap

Abertura de Arquivo

Para conseguirmos ler o arquivo e preciso ter um ponteiro que aponte para o arquivo então a forma de definir um é a seguinte:

```
FILE *fp = fopen(argv[2], "rb");
```

Estrutura pcap_header

Para que consigamos percorrer todo o arquivo e que saibamos os dados que estão definidos la corretamente e necessário que se saiba a estrutura utilizada para gravar ou seja estruturas dos dados lá presente a estrutura pcap header e responsável pelos dados do arquivo pcap ou seja todo cabeçalho pcap tem quer ter ela no inicio.

Comando fread

Esse Comando foi utilizado para ler a estrutura do cabeçalho ou seja ele faz a copia dos dados presente no arquivo para a estrutura pheader.

Estrtura pcap_packethdr

Responsável por fazer a leitura do cabeçalho de cada pacote ou seja todo arquivo pcap tem antes de cada pacote essa estrutura.

memset(buffer, 0, MAX_LENGTH);

Responsável por atribuir zero ao buffer após cada interação do recebimento de pacotes

fread(buffer, pkthdr.caplen, 1, fp);

Responsável por fazer leitura de cabeçalho pcap de cada pacote

struct eth_hdr *eth = (struct eth_hdr*) buffer;

Responsável por movimentar o ponteiro de leitura do arquivo para a posição do protocolo ethernet

Em seguida temos a exibição dos dados do protoco ethernet

struct ip_hdr *ip = (struct ip_hdr*) (buffer + sizeof(struct eth_hdr));

Responsável por movimentar o ponteiro de leitura do arquivo para a posição do protocolo ip

Em seguida temos a exibição dos dados do protocolo ip

if(ip->protocol == 6)

Aqui temos o teste se a conexão foi feita por TCP ou UDP se for 6 temos uma conexão TCP

struct tcp_hdr *tcp = (struct tcp_hdr*) (buffer + sizeof(struct eth_hdr) + ip->ihl*4);

Responsável por movimentar o ponteiro de leitura do arquivo para a posição do protocolo TCP

Em seguida temos a exibição dos dados do protocolo TCP

if(ip->protocol == 17)

Aqui temos o teste se a conexão foi feita por TCP ou UDP se for 17 temos uma conexão UDP

struct udp_hdr *udp = (struct udp_hdr*) (buffer + sizeof(struct eth_hdr) + ip->ihl*4);

Responsável por movimentar o ponteiro de leitura do arquivo para a posição do protocolo UDP

Em seguida temos a exibição dos dados do protocolo UDP

Resumo

Para se abrir um arquivo e necessário ter um ponteiro que a ponte para o mesmo e para navegarmos pelo dados e necessário saber se o que esta buscando no arquivo e quais estruturas foram utilizadas na sua gravação, aqui temos nessa parte leitura primeiro precisamos saber qual a estrutura de criação de uma arquivo pcap ao sabermos essa estrutura teremos também seu tamanho sendo assim basta chamar o comando fread para fazermos a leitura dos dados para estrutura logo depois e preciso usar o um while para que continue lendo todo arquivo, temos uso da função memset que coloca zero em todos os campo do nosso buffer para não reste vestígio do pacote anterior, também e necessário saber a estrutura pcap de cabeçalho de cada pacote e de novo chamaremos fread para fazemos a leitura dos dados, e logo depois fazemos a movimentação do ponteiro de leitura para os dados do ethernet para que seja possível fazer casting do buffer em uma struct ethernet e necessário que se saiba como e a struct ao fazer isso temos acesso aos dados do protocolo ethernet e funciona da mesma forma para o protocolo ip tcp e udp.

Comando de execução captura de pacotes na interface

```
sudo ./executavel -w nomeArquivo.pcap -i interface
```

Escrita de arquivo pcap com pacotes capturados

A parte de escrita de arquivo pcap consiste na captura de pacotes utilizando o sock_raw dos pacotes capturados serão selecionados apenas os TCP e o UDP para serem escritos no arquivo pcap, nesse código esta sendo armazenado somente a parte do cabeçalho dos arquivos sendo feito o descarte dos dados.

Como foi escrito esse código primeiro e feito o teste dos argumento informados pelo usuário se estiver correto é criado um file pointer que será responsável em apontar para o arquivo pcap essefile pointer recebe a função fopen() responsável por criar nosso arquivo texto segundo os parâmetros passados a ela nesse caso os parâmetros são nome do arquivo e o tipo de instrução que é gravação de arquivo binário, logo em seguida temos a criação da estrutura pheader e preenchimento responsável pelo cabeçalho do arquivo pcap e o define se o arquivo e pcap ou não logo em seguida temos a função fwrite() que recebe pheader e grava no arquivo pcap, em seguidas testamos se o usuário fez a passagem de 5 argumentos e posteriormente chamamos a função socket() passando a ela o parâmetro de PF_PACKET utilizado para conseguimos especificar uma interface, SOCK_RAW responsável por capturarmos os pacotes na camada de transporte, por fim temos htons(ETH_P_ALL) que permite o socket ter acesso aos protocolos antes dos protocolos do kernel, a variável sockfd recebe o retorno da função socket a mesma e testado para ver se a criação do socket ocorreu de forma correta, logo abaixo temos a função strncpy fazendo a mudança do nome da interface para a que o usuário informo ao abrir o código logo abaixo e feito a chamada de ioctl() responsável por Defina a interface para o modo promíscuo , a diante temos setsockopt() nos permite configurar o socket há duas chamadas dessa função a primeira com objetivo de Permite que outros sockets se liguem () a esta porta, a menos que já exista um socket de escuta ativo ligado à porta. Isso

permite que você contorne as mensagens de erro "Endereço já em uso" ao tentar reiniciar o servidor após uma falha, a segunda para Vincule este soquete a um nome de dispositivo simbólico como eth0 em vez de usar bind () para vinculá-lo a um endereço IP, ate aqui vimos a parte de configuração do socket de agora em diante será a parte de receber os dados e escrita

Temos a criação da estrutura hpcap que é responsável pelo cabeçalho de cada pacote

Ela conte informações de te hora de captura e tamanho do pacote, logo abaixo temos um laço de repetição que e responsável por manter o recebimentos de dados pois dentro dele se encontra a função recv() resposanvel por receber mensagem do socket e guarda na variável buffer, recebendo o retorno da função recv() temos n logo abaixo e testado esse valor de retorno para verifica se aconteceu algum erro, se n for maior que 3 é dado inicio ao processor de escrita dos dados do pacote no arquivo pcap, primeiro a passo a ser feito é preencher a estrutura timeval que esta dentro da estrutura hpcap essa estrutura precisa receber os dados de tempo de recebimento e para preencher a mesma vamos fazer uso da função gettimeofday() que recebe como primeiro parâmetro estrutura timeval e segundo parâmetro temos o fuso horário vamos deixar como NULL a chamada da função já faz preenchimento da estrutura timeval, logo abaixo fazemos o cast da estrutura ip para conseguirmos se mover no buffer para a parte dos dados do ip necessário para verificar se o protocolo e tcp ou udp, daqui em diante iniciasse o processo de escrita dos protocolos e ethernet IP TCP ou UDP, se o pacote recebido for TCP entraremos no if(ip->protocol==6) e teremos o seguintes códigos comentados e suas funções :

```
unsigned short ip_hdrlen;
struct ip_hdr *ip_h = (struct ip_hdr *)( buffer  + sizeof(struct eth_hdr) );
```

Essa parte responsável por fazer cast do buffer utilizando a estrutura IP para utilizamos para obter os dados protocolo IP .

```
ip_hdrlen = ip_h->ihl*4;
```

desse modo obtemos o tamanho do cabeçalho ip porem não ficamos restringido a 20 bytes podendo variar

```
hpcap.caplen = sizeof(struct eth_hdr)+ip_hdrlen +sizeof(struct tcp_hdr);
hpcap.len = sizeof(struct eth_hdr)+ip_hdrlen +sizeof(struct tcp_hdr);
```

Aqui temos o preenchimento dos campos de tamanho dos cabeçalhos do pacotes da estrutura hpcap ou seja temos a soma de todos os cabeçalhos aqui temos uma diferença entre o TCP e o UDP no TCP deve se colocar o tamanho do cabeçalho TCP e no UDP coloca se o tamanho do cabeçalho UDP.

```
fwrite(&hpcap, sizeof(struct pcap_packethdr),1,fp);
```

Aqui temos o comando de escrita no arquivo pcap

```
struct eth_hdr *eth = (struct eth_hdr *)buffer;
```

Essa parte responsável por fazer cast do buffer utilizando a estrutura Ethernet para utilizamos para obter os dados protocolo Ethernet .

```
fwrite(eth, sizeof(struct eth_hdr),1,fp);
```

Aqui temos o comando de escrita da dos dados do protocolo ethernet no arquivo pcap.

```
struct ip_hdr *ip = (struct ip_hdr *)(buffer  + sizeof(struct eth_hdr));
```

Essa parte responsável por fazer cast do buffer utilizando a estrutura IP para utilizamos para obter os dados protocolo IP .

```
fwrite(ip, ip_hdrlen,1,fp);
```

Aqui temos o comando de escrita da dos dados do protocolo IP no arquivo pcap.

```
struct tcp_hdr *tcp = (struct tcp_hdr*)(buffer + ip_hdrlen + sizeof(struct eth_hdr));
```

Essa parte responsável por fazer cast do buffer utilizando a estrutura TCP para utilizamos para obter os dados protocolo TCP .

```
fwrite(tcp, sizeof(struct tcp_hdr),1,fp);
```

Aqui temos o comando de escrita da dos dados do protocolo TCP no arquivo pcap.

Logo abaixo temos a else if(ip->protocol==17) que faz o teste para saber se o protocolo e UDP abaixo dele se repete os mesmo comando com alteração da estrutura TCP para UDP da seguinte forma

```
unsigned short ip_hdrlen;
struct ip_hdr *ip_h = (struct ip_hdr *)( buffer  + sizeof(struct eth_hdr) );
```

Essa parte responsável por fazer cast do buffer utilizando a estrutura IP para utilizamos para obter os dados protocolo IP .

```
ip_hdrlen = ip_h->ihl*4;
```

desse modo obtemos o tamanho do cabeçalho ip porem não ficamos restringido a 20 bytes podendo variar

```
hpcap.caplen = sizeof(struct eth_hdr)+ip_hdrlen +sizeof(struct udp_hdr);
hpcap.len = sizeof(struct eth_hdr)+ip_hdrlen +sizeof(struct udp_hdr);
```

Aqui temos o preenchimento dos campos de tamanho dos cabeçalhos do pacotes da estrutura hpcap ou seja temos a soma de todos os cabeçalhos aqui temos uma diferença entre o TCP e o UDP no TCP deve se colocar o tamanho do cabeçalho TCP e no UDP coloca se o tamanho do cabeçalho UDP.

```
fwrite(&hpcap, sizeof(struct pcap_packethdr),1,fp);
```

Aqui temos o comando de escrita no arquivo pcap

```
struct eth_hdr *eth = (struct eth_hdr *)buffer;
```

Essa parte responsável por fazer cast do buffer utilizando a estrutura Ethernet para utilizamos para obter os dados protocolo Ethernet .

```
fwrite(eth, sizeof(struct eth_hdr),1,fp);
```

Aqui temos o comando de escrita da dos dados do protocolo ethernet no arquivo pcap.

```
struct ip_hdr *ip = (struct ip_hdr *)(buffer  + sizeof(struct eth_hdr));
```

Essa parte responsável por fazer cast do buffer utilizando a estrutura IP para utilizamos para obter os dados protocolo IP .

```
fwrite(ip, ip_hdrlen,1,fp);
```

Aqui temos o comando de escrita da dos dados do protocolo IP no arquivo pcap.

```
struct udp_hdr *udp = (struct udp_hdr*)(buffer + ip_hdrlen + sizeof(struct eth_hdr));
```

Essa parte responsável por fazer cast do buffer utilizando a estrutura TCP para utilizamos para obter os dados protocolo TCP .

```
fwrite(udp, sizeof(struct udp_hdr),1,fp);
```

Aqui temos o comando de escrita da dos dados do protocolo UDP no arquivo pcap.

Assim finalizamos a parte de escrita, nosso while e controlado pela variável recebendo que por sua vez por valor default esta com o valor 1 mas a mesma variável pode ter seu estado alterado pela função signal(SIGINT,meutratadordesinal) quando o usuário aperta Ctrl + c.

Função que capturar Ip da interface em uso para o sock_raw

Essa função foi utilizada com a finalidade de se obter o ip da interface para podermos fazer a verificação se o ip esta sendo enviado ou recebido por nossa interface

```
void ip(char interface[]);
```

Função para obter o tempo
Para que possamos calcular velocidade de envio e recebimento de dados

```
double tempo();
```

Função responsável pela interrupção da escrita

Essa função e responsável por fazer a chamada da função que gerar a mudança de estado do laço de repetição para poder parar o recebimento de dados, sendo assim e utilizado a função signal() para receber o sinal de interrupção.

```
signal(SIGINT,meutratadordesinal);
```

que chama

```
void  meutratadordesinal(int sig);
```