Documentação - Packet Sniffer 1.0

Desenvolvimento

O software foi desenvolvido na linguagem **C**, em ambiente **Mac OS**, utilizando o editor de texto **TextMate**.

Utiliza-se da biblioteca *pcap* (pcap.h) para a captura de pacotes em si e da biblioteca *pthread* (pthread.h) para que possam existir 2 ou mais traços de execução paralelos. Além dessas bibliotecas, o programa se utiliza de alguns headers da família netinet (ip.h, ip6.h, in_systm.h, icmp6.h, ip_icmp.h, tcp.h, udp.h e if_ether.h), bem como headers para operações de internet (arpa/inet.h), funções envolvendo entrada/saída (stdio.h), conversões/alocação de memória (stdlib.h), manipulação de arrays (string.h) e com definições de constantes (unistd.h). Por fim, também são utilizados dois arquivos contendo structs (structs.c) e constantes (constants.c) definidas/criadas pelo desenvolvedor da aplicação.

Funções Existentes

- int main(int argc, char *argv[]) - Fluxo principal da aplicação. Parâmetros:
  - int argc - Número de argumentos passados ao programa;
  - char *argv[] - Array de ponteiros para as strings que são os argumentos em si

(OBS.: Os parâmetros acima estão presentes apenas por serem a assinatura padrão do main, já que o programa não recebe atributos na sua chamada).

- void* watching_keyboard(void *data) - Função que será utilizada pela pthread_init na chamada da thread auxiliar para monitoramento do teclado e eventual interrupção de captura (No caso específico do programa, quando os caracteres ‘q’ ou ‘Q’ forem pressionados). Parâmetros:
  - void *data - Parâmetro enviado pela pthread_init.

(OBS.: O parâmetro acima está presente apenas por ser a assinatura padrão do método de pthread_init, já que o método não se utiliza de parâmetro de entrada algum).

- void got_packet(u_char *args, const struct pcap_pkthdr *header, const u_char *packet) - Função que será utilizada pelo pcap_loop e chamada na entrada de cada novo pacote. Parâmetros:
  - u_char *args - Argumento passado pelo pcap_loop.
  - const struct pcap_pkthdr *header - header do pacote.
  - const u_char *packet - ponteiro para o primeiro byte do conteúdo do pacote em si.

- void process_ethernet(const u_char *packet) - Processa um pacote que possui protocolo da camada de enlace. Caso o pacote possua informações de camadas superiores (Neste caso, rede), o método encaminhará o pacote para o devido processamento. Parâmetros:
  - const u_char *packet: ponteiro para o primeiro byte do conteúdo do pacote em si.

- void process_IP(const u_char *packet) - Processa um pacote que possui o protocolo de camada de transporte IP. Caso o pacote possua informações de camadas superiores (Neste caso, transporte e aplicação), o método fará o processamento necessário para obtenção das devidas informações; ou encaminhará o pacote para o devido processamento, no caso do ICMP. Parâmetros:

    - const u_char *packet: ponteiro para o primeiro byte do conteúdo do pacote em si.

- void process_IPv6(const u_char *packet) - Processa um pacote que possui o protocolo de camada de transporte IPv6. Caso o pacote possua informações de camadas superiores (Neste caso, transporte e aplicação), o método fará o processamento necessário para obtenção das devidas informações; ou encaminhará o pacote para o devido processamento, no caso do ICMPv6. Parâmetros:
    - const u_char *packet: ponteiro para o primeiro byte do conteúdo do pacote em si.

- void process_ICMP(const u_char *packet) - Processa um pacote que possui o protocolo da camada de rede ICMP. Parâmetros:
    - const u_char *packet: ponteiro para o primeiro byte do conteúdo do pacote em si.

- void process_ICMPv6(const u_char *packet) - Processa um pacote que possui o protocolo da camada de rede ICMPv6. Parâmetros:
    - const u_char *packet: ponteiro para o primeiro byte do conteúdo do pacote em si.

- void get_ICMP_message(int icmpType, int icmpCode, char **icmpMessage) - Busca pela mensagem ICMP correspondente ao tipo e código passados por parâmetro e a atribui ao parâmetro icmpMessage. Parâmetros:
    - int icmpType - Tipo da mensagem ICMP;
    - int icmpCode - Código da mensagem ICMP;
    - char **icmpMessage) - Ponteiro para um ponteiro de char que irá armazenar a mensagem.

- void get_ICMPv6_message(int icmpType, int icmpCode, char **icmpMessage) - Busca pela mensagem ICMPv6 correspondente ao tipo e código passados por parâmetro e a atribui ao parâmetro icmpMessage. Parâmetros:
    - int icmpType - Tipo da mensagem ICMPv6;
    - int icmpCode - Código da mensagem ICMPv6;
    - char **icmpMessage) - Ponteiro para um ponteiro de char que irá armazenar a mensagem.

- void get_IPv6_message(int ipPriority, char **icmpMessage) - Busca pela mensagem IPv6 correspondente ao código de prioridade passado por parâmetro e a atribui ao parâmetro icmpMessage. Parâmetros:
    - int ipPriority - Código de prioridade IPv6;
    - char **icmpMessage) - Ponteiro para um ponteiro de char que irá armazenar a mensagem.

- void print_packet_line(struct sniffer_packet *sp, int number) - Imprime na saída padrão o pacote passado por parâmetro. Parâmetros:
    - struct sniffer_packet *sp - Ponteiro para o pacote;
    - int number - Número do pacote a ser impresso.

- void print_packets(sniffer_packet *packets) - Imprime na saída padrão os pacotes contidos no array de pacotes passado por parâmetro. Utiliza-se da print_packet_line() citada anteriormente. Parâmetros:
    - sniffer_packet *packets - Ponteiro para o array de pacotes.

- void print_summary() - Imprime o resumo da captura realizada.

- void print_header() - Imprime o cabeçalho da lista de pacotes na saída padrão.

- void print_ipv6_graph(sniffer_packet **packets) - Imprime o gráfico de prioridade dos pacotes IPv6. Parâmetros:
  - sniffer_packet **packets - Ponteiro para o ponteiro do array de pacotes.

- void print_ipv4_ipv6_graph(sniffer_packet **packets) - Imprime o gráfico comparativo de pacotes IPv4 e IPv6. Parâmetros:
  - sniffer_packet **packets - Ponteiro para o ponteiro do array de pacotes.

- void print_icmp6_graph(sniffer_packet **packets) - Imprime o gráfico dos tipos de mensagens dos pacotes ICMPv6. Parâmetros:
  - sniffer_packet **packets - Ponteiro para o ponteiro do array de pacotes.

- int add_to_sniffer_array (sniffer_packet *item) - Adiciona um pacote ao array de pacotes. Parâmetros:
  - sniffer_packet *packets - Ponteiro para o pacote a ser adicionado ao array.

- hash_t *hash_init(int size) - Inicializa um hash. Parâmetros:
  - int size - Tamanho do hash.

void hash_increment(hash_t *hash, int key) - Incrementa um dos valores do hash. Parâmetros:
- hash_t *hash - Hash que terá um dos valores incrementado;
- int key - Valor da chave a ser incrementada.

void free_hash(hash_t *hash) - Libera a memória ocupada pelo hash. Parâmetros:
- hash_t *hash - Hash que terá a memória desalocada.

Manual de utilização

1) Abra o Terminal;
2) Dirija-se ao diretório aonde se encontra o sniffer (cd).
3) Adquira privilégios de superusuário (sudo).
4) Digite ./packet_sniffer.
5) O menu principal será mostrado.
    1) Na opção 1, selecione o dispositivo a ser “sniffado” (Padrão: en1);
    2) Na opção 2, configure o filtro de captura desejado (Padrão: vazio);
    3) Na opção 3, configure a quantidade de pacotes a ser capturada, ou -1 para captura infinita (Padrão: -1)
    4) Na opção 4, a captura em si poderá ser iniciada;
        1) Uma vez em captura, pressione ‘q’ ou ‘Q’ a qualquer momento para interromper a captura.
        2) O menu pós-captura será mostrado.
            1) Na opção 1, o comparativo gráfico entre pacotes IPv4 / IPv6 será exibido.
            2) Na opção 2, o comparativo gráfico entre as prioridades dos pacotes IPv6 será exibido.
            3) Na opção 3, o comparativo gráfico entre as mensagens dos pacotes ICMPv6 será exibido.
            4) Na opção 4, o programa será encerrado.

5) Na opção 5, o programa será encerrado.

Manual de compilação

O Packet Sniffer utiliza da seguinte linha de comando para a compilação:

gcc -Wall packet_sniffer.c -o packet_sniffer -lpcap -lpthread