DOCUMENTAÇÃO DE IMPLEMENTAÇÃO DO IPv6

Universidade de Caxias do Sul (UCS)
Disciplina: Redes de Computadores
Aluno: Matheus Elias da Silva

https://github.com/buckyroberts/Python-Packet-Sniffer (original)
https://github.com/sirvisconde/Python-Packet-Sniffer (fork)

Foi empregado o uso de conceitos de padrões de projeto para realizar a fatoração. Através da utilização da interface "Protocol", que é implementada por todos os protocolos abordados, foi viabilizada a simplificação da expansão do projeto. Os protocolos recebem os dados binários do pacote em seu construtor e montam seus cabeçalhos por meio da manipulação de bits utilizando a biblioteca struct do Python. Um objeto Protocol possui a função print_data, que além de exibir os cabeçalhos do pacote na tela, também invocará a mesma função no protocolo encapsulado, caso ele exista.

O projeto original possuía cabeçalhos de Ethernet, ICMP, IPv4, TCP, UDP e HTTP. Os protocolos implementados com o fork foram:

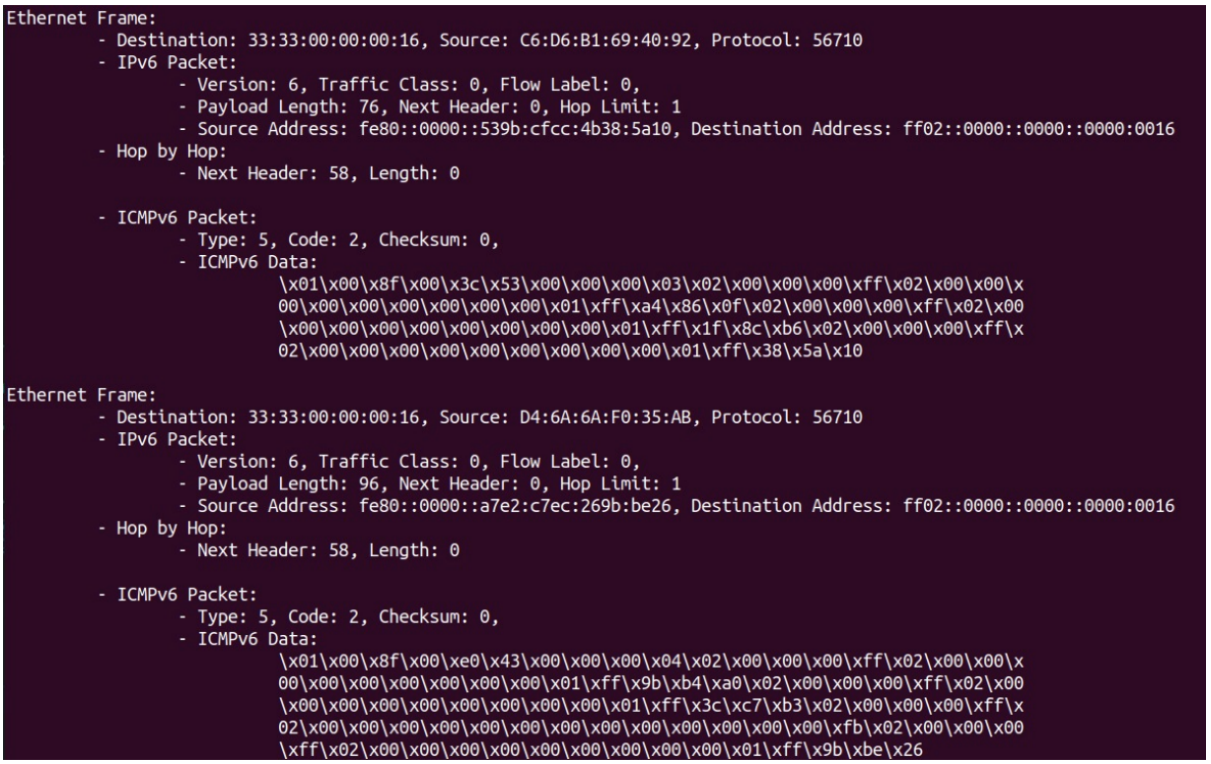
IPv6:

- Version
- Traffic Class
- Flow Label
- Payload Length
- Next Header
- Hop Limit
- Source
- Target

ICMPv6:

- Type
- Type Name
- Code
- Code Name
- Checksum

Segue exemplo de pacotes capturados pelo programa:



```
Ethernet Frame:
        - Destination: 33:33:00:00:00:16, Source: C6:D6:B1:69:40:92, Protocol: 56710
        - IPv6 Packet:
                - Version: 6, Traffic Class: 0, Flow Label: 0,
                - Payload Length: 76, Next Header: 0, Hop Limit: 1
                - Source Address: fe80::0000::539b:cfcc:4b38:5a10, Destination Address: ff02::0000::0000::0000:0016
        - Hop by Hop:
                - Next Header: 58, Length: 0

        - ICMPv6 Packet:
                - Type: 5, Code: 2, Checksum: 0,
                - ICMPv6 Data:
                        \x01\x00\x8f\x00\x3c\x53\x00\x00\x00\x03\x02\x00\x00\x00\xff\x02\x00\x00\x
                        00\x00\x00\x00\x00\x00\x00\x01\xff\xa4\x86\x0f\x02\x00\x00\x00\xff\x02\x00
                        \x00\x00\x00\x00\x00\x00\x00\x00\x01\xff\x1f\x8c\xb6\x02\x00\x00\x00\xff\x
                        02\x00\x00\x00\x00\x00\x00\x00\x00\x00\x01\xff\x38\x5a\x10

Ethernet Frame:
        - Destination: 33:33:00:00:00:16, Source: D4:6A:6A:F0:35:AB, Protocol: 56710
        - IPv6 Packet:
                - Version: 6, Traffic Class: 0, Flow Label: 0,
                - Payload Length: 96, Next Header: 0, Hop Limit: 1
                - Source Address: fe80::0000::a7e2:c7ec:269b:be26, Destination Address: ff02::0000::0000::0000:0016
        - Hop by Hop:
                - Next Header: 58, Length: 0

        - ICMPv6 Packet:
                - Type: 5, Code: 2, Checksum: 0,
                - ICMPv6 Data:
                        \x01\x00\x8f\x00\xe0\x43\x00\x00\x00\x04\x02\x00\x00\x00\xff\x02\x00\x00\x
                        00\x00\x00\x00\x00\x00\x00\x01\xff\x9b\xb4\xa0\x02\x00\x00\x00\xff\x02\x00
                        \x00\x00\x00\x00\x00\x00\x00\x00\x01\xff\x3c\xc7\xb3\x02\x00\x00\x00\xff\x
                        02\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xfb\x02\x00\x00\x00
                        \xff\x02\x00\x00\x00\x00\x00\x00\x00\x00\x00\x01\xff\x9b\xbe\x26
```

Structs e parâmetros:

No construtor __init__, o cabeçalho IPv6 é desempacotado usando a função struct.unpack. Os valores dos campos relevantes são extraídos e atribuídos aos respectivos atributos da classe.

A função ipv6 converte o endereço IPv6 de bytes para uma representação legível, no formato de grupos hexadecimais separados por ":"

O campo data armazena os dados adicionais do pacote IPv6 (payload).

A função struct.unpack (formato, buffer) é usada para desempacotar dados binários de acordo com o formato especificado. No caso desse trecho de código, o formato é '!IHBB16s16s'.

Aqui a explicação de cada parte do formato:

- '!': Indica a formatação de rede, que garante a ordem correta dos bytes.

- I: Especifica um valor de 4 bytes (integer sem sinal).
- H: Especifica um valor de 2 bytes (integer sem sinal).
- B: Especifica um valor de 1 byte (integer sem sinal).
- 16s: Especifica uma sequência de 16 bytes (string de tamanho fixo).
- 16s: Especifica outra sequência de 16 bytes (string de tamanho fixo).

Parâmetro raw_data[:40]:

- raw_data: É o buffer de dados brutos que será desempacotado.
- [:40]: Indica que foram selecionados os primeiros 40 bytes do buffer raw_data para serem desempacotados.

O resultado do struct.unpack será uma tupla contendo os valores desempacotados de acordo com o formato especificado. A ordem dos valores na tupla corresponde à ordem definida no formato.

Por exemplo, se raw_data contiver um pacote IPv6 completo de 40 bytes, o struct.unpack retornará uma tupla com os seguintes valores:

O primeiro valor será um inteiro sem sinal de 4 bytes (I).
O segundo valor será um inteiro sem sinal de 2 bytes (H).
O terceiro valor será um inteiro sem sinal de 1 byte (B).
O quarto valor será outro inteiro sem sinal de 1 byte (B).
O quinto valor será uma sequência de 16 bytes (16s).
O sexto valor será outra sequência de 16 bytes (16s).

Na classe ICMPv6, está sendo feito um struct.unpack('! B B H', raw_data[:4]) que é usado para desempacotar os primeiros 4 bytes dos dados brutos (raw_data) usando o formato ! B B H.

Aqui está o que cada parte significa:

- !: indica a formatação de rede, que garante a ordem correta dos bytes.
- B B H: especifica os tipos e a ordem dos valores esperados.
- B significa unsigned char (1 byte) e H significa unsigned short (2 bytes).