Assignment 3: Packet Generation

The goal of this assignment is to generate a `.pcap` file containing 12 packets based on the given `.pcap` in Assignment 2. You must use Scapy in Python to construct and append each packet, then write the full set of packets to a `.pcap` file that can be analyzed using TShark or Wireshark. The `.pcap` file that you generate must contain the same number of packets in the same order with identical fields, identical values for each field (including payloads where necessary) and identical timestamps for each packet as seen in the `.pcap` file that you were given in Assignment 2.

> Tip: Scapy can also be used to analyse a given `.pcap` file. However, you might find it easier to use TShark or Wireshark for this.

Requirements

The `.pcap` file that you produce using Scapy must contain the following features, which make it identical to the `.pcap` file that you were given in Assignment 2:

- 12 total packets.
- 1 ICMPv6 Time Exceeded error packet, with code 0 or 1, enclosing a UDP packet.
- 1 ICMPv6 Echo Request.
- 1 ICMPv6 Echo Reply.
- Remaining 9 are split between TCP/UDP and have identical hexadecimal string payloads as seen in Assignment 2 pcap. And one of the 9 packets will be DNS which is either TCP/UDP.

Sample Example

The following is a sample example of how to use Scapy to generate a `.pcap` file with a single packet.

```
 # Import Scapy
from scapy.all import Ether, IPv6, UDP, rdpcap, wrpcap

# Create a new packet stacked with Ether, IPv6, UDP, and a payload
packet = Ether(src='...', dst='...')
        /IPv6(src='...', dst='...')
        /UDP(sport=53, dport=80)

# Write the .pcap file
wrpcap('<username>.pcap', packet)
```

Use `ls()` to list all the available layers and `ls(IP)` to list all the available fields for the IP layer. For instance:

```
 # List available Scapy layers
ls()

# Sample output
 Layer2        :: Ether, Dot3, LLC, SNAP
 Layer3        :: IP, IPerror, IPy, GRE
 Transport     :: TCP, UDP
 Packet        :: Packet, bind_layers
...
```

More information on how to use Scapy can be found in the Scapy Documentation.

Important:

- Packets must be constructed in order defined by requirements. Analyse the `.pcap` file from Assignment 2 to determine the order.
- Payloads and other parameters should NOT be randomized. Use the given `.pcap` file to determine the values for each packet. Use `-T fields -e data` to extract the payloads from the `.pcap` file that you were given.
- When defining non-DNS packet payloads, refer to the layer documentation on Raw byte string formatting. The load field should be populated with a byte string representing the payload data. Review the syntax guide here: Scapy Raw
- Any raw byte string payloads you construct should be prefixed with b to indicate it is a byte string rather than a regular string. For example:

```
    payload = b"abc123" # Bytes
   payload_string = "abc123" # String
```

Testing

- Use TShark/Wireshark to validate the contents of generated `.pcap` -- and thus check the output of your Scapy program.
- Check for the correct number of packets and the field values.
- Verify addressing, ports, and protocols.
- Check a few packet payloads at random to validate contents.

Submission

Submit your `.py` script and the generated `.pcap` file in a `.zip` file through Blackboard by the due date. Ensure that your `.py` script is well-documented with relevant comments and follows the PEP 8 style guide for Python. The `.zip` file should be named in the format `A3_<username>.zip` before submission.