SMV Code Documentation

Introduction

The Sampled Measured Values (SMV) protocol is a part of the IEC 61850 standard and is used for transmitting digitized measurements between devices within electrical substations in real time. This documentation provides detailed instructions for using the SMV protocol code, which supports packet manipulation, live capture, and parsing of SMV messages.

The SMV protocol implementation is divided into separate files for better management:

- `sv.py`: Contains the definitions for SMV packets.
- `sv_pdu.py`: Contains the ASN.1 definitions for SMV PDU (Protocol Data Unit).

System Requirements

Python Version:

- Python 3.x

Required Libraries:

- `pyasn1`: For ASN.1 encoding/decoding of SMV messages.
- `Scapy`: For crafting and capturing network packets.
- `PyShark`: For live packet capture.

Install the required libraries using the following command:

```
pip install pyasn1 scapy pyshark
```

Additionally, ensure the local `smv` folder is located in the parent directory of your project, as it contains the necessary `sv.py` and `sv_pdu.py` files.

Usage Instructions

The SMV protocol code provides two options for capturing and processing packets: **live capture** from a network interface and **offline processing** of pre-recorded `.pcap` files.

Command Line Options

You can use the following commands to run the code:

1. **Live Capture**:
   Capture SMV messages directly from a live network interface:

   ```
   python3 script_name.py --livecapture --store <filename.pcap>
   ```

   Replace `script_name.py` with the name of your script. The captured packets will be stored in the specified file.

2. **Offline Processing**:
   Process packets from an existing `.pcap` file:

   ```
   python3 script_name.py --pcapfile <input_filename.pcap> --output <output_filename.pcap>
   ```

3. **Help :**

   ```
   python3 script_name.py -h
   ```

Network Interface and Filter Setup

To capture only SMV packets, the script applies a filter based on EtherType:

- **Network Interface**: `eth1` (can be changed based on your system).
- **EtherType Filter**: SMV EtherType is `0x88ba`.

```
networkInterface = "eth1"
filter_string = "ether proto 0x88ba"
```

```
SV_TYPE = 0x88ba
```

You can modify the `networkInterface` variable in the code to capture from a different interface.

Manual Code Modifications

The core packet processing logic and modification of SMV packets are handled within the code. Specific sections are highlighted below:

- The code supports modifying the `svID`, `smpCnt`, and `seqData` fields in the SMV packet's ASDU (Application Service Data Unit) section. This modification is done within the workflow between `packets_to_modify` and `filtered_packets`.

Modification Logic

The core modification logic is implemented in the section where the following fields are updated:

- **svID**: Set by the user for injection.
- **smpCnt**: Sequentially modified in the provided range.
- **seqData**: Modified to alter the first three bytes for testing purposes.

This part of the logic allows for injecting the modified packets back into the network or saving them to a `.pcap` file using the `wrpcap` function.

Sample Workflow

1. **Live Capture**:
    - The script captures packets from the `eth1` interface, filters for SMV packets using `ether proto 0x88ba`, and stores them in a `.pcap` file.
2. **Filtering SMV Packets**:
    - The captured packets are filtered to retain only the SMV packets, which are then decoded.
3. **Modifying Packets**:

- Users can choose specific packets for modification. The script allows for changing the `svID`, `smpCnt`, and `seqData` fields.

4. **Injecting Packets**:
   - The modified packets can be injected back into the network (optional), or saved in a `.pcap` file for further analysis.

Notes

- Before performing **DOS attack** check the traffic frequency and choose a method accordingly

  → Refer to my documentation on "***Tools for sending packets***"

- **Wireshark**: To analyze the SMV packets, you can use Wireshark and apply the display filter `ether proto 0x88ba` to isolate SMV traffic.
- **For DEBUG** if any error occurred:
  - Use this before call the decoder function

    ```
    from pyasn1 import debug
    debug.setLogger(debug.Debug('all'))
    ```

  - Also check the `sv_pdu` parameters in `sv_pdu.py` and match them with the packet using wireshark
  - Asnate library
