pyshark

https://pypi.org/project/pyshark/

Installation

All Platforms

Simply run the following to install the latest from pypi

```
pip install pyshark
```

Or install from the git repository:

```
git clone https://github.com/KimiNewt/pyshark.git
cd pyshark/src
python setup.py install
```

Obs. – Anaconda:

C:\Users\....>conda activate C:\Users\.....\Anaconda3
C:\Users\.....>pip install --upgrade --force-reinstall pyshark

Mac OS X

You may have to install libxml which can be unexpected. If you receive an error from clang or an error message about libxml, run the following:

```
xcode-select --install
pip install libxml
```

You will probably have to accept a EULA for XCode so be ready to click an "Accept" dialog in the GUI.

Usage

Reading from a capture file:

```
>>> import pyshark
>>> cap = pyshark.FileCapture('/tmp/mycapture.cap')
```

```
>>> cap
<FileCapture /tmp/mycapture.cap (589 packets)>
>>> print cap[0]
Packet (Length: 698)
Layer ETH:
        Destination: BLANKED
        Source: BLANKED
        Type: IP (0x0800)
Layer IP:
        Version: 4
        Header Length: 20 bytes
        Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00:
Not-ECT (Not ECN-Capable Transport))
        Total Length: 684
        Identification: 0x254f (9551)
        Flags: 0x00
        Fragment offset: 0
        Time to live: 1
        Protocol: UDP (17)
        Header checksum: 0xe148 [correct]
        Source: BLANKED
        Destination: BLANKED
  ...
```

Other options

- **param keep_packets**: Whether to keep packets after reading them via next(). Used to conserve memory when reading large caps.
- **param input_file**: Either a path or a file-like object containing either a packet capture file (PCAP, PCAP-NG..) or a TShark xml.
- **param display_filter**: A display (wireshark) filter to apply on the cap before reading it.
- **param only_summaries**: Only produce packet summaries, much faster but includes very little information
- **param disable_protocol**: Disable detection of a protocol (tshark > version 2)
- **param decryption_key**: Key used to encrypt and decrypt captured traffic.
- **param encryption_type**: Standard of encryption used in captured traffic (must be either 'WEP', 'WPA-PWD', or 'WPA-PWK'. Defaults to WPA-PWK.
- **param tshark_path**: Path of the tshark binary

Reading from a live interface:

```
>>> capture = pyshark.LiveCapture(interface='eth0')
>>> capture.sniff(timeout=50)
>>> capture
<LiveCapture (5 packets)>
>>> capture[3]
<UDP/HTTP Packet>

for packet in capture.sniff_continuously(packet_count=5):
    print 'Just arrived:', packet
```

Other options

- **param interface**: Name of the interface to sniff on. If not given, takes the first available.
- **param bpf_filter**: BPF filter to use on packets.
- **param display_filter**: Display (wireshark) filter to use.
- **param only_summaries**: Only produce packet summaries, much faster but includes very little information
- **param disable_protocol**: Disable detection of a protocol (tshark > version 2)
- **param decryption_key**: Key used to encrypt and decrypt captured traffic.
- **param encryption_type**: Standard of encryption used in captured traffic (must be either 'WEP', 'WPA-PWD', or 'WPA-PWK'. Defaults to WPA-PWK).
- **param tshark_path**: Path of the tshark binary
- **param output_file**: Additionally save captured packets to this file.

Reading from a live interface using a ring buffer

```
>>> capture = pyshark.LiveRingCapture(interface='eth0')
>>> capture.sniff(timeout=50)
>>> capture
<LiveCapture (5 packets)>
>>> capture[3]
<UDP/HTTP Packet>

for packet in capture.sniff_continuously(packet_count=5):
    print 'Just arrived:', packet
```

Other options

- **param ring_file_size**: Size of the ring file in kB, default is 1024
- **param num_ring_files**: Number of ring files to keep, default is 1
- **param ring_file_name**: Name of the ring file, default is /tmp/pyshark.pcap
- **param interface**: Name of the interface to sniff on. If not given, takes the first available.
- **param bpf_filter**: BPF filter to use on packets.
- **param display_filter**: Display (wireshark) filter to use.
- **param only_summaries**: Only produce packet summaries, much faster but includes very little information
- **param disable_protocol**: Disable detection of a protocol (tshark > version 2)
- **param decryption_key**: Key used to encrypt and decrypt captured traffic.
- **param encryption_type**: Standard of encryption used in captured traffic (must be either 'WEP', 'WPA-PWD', or 'WPA-PWK'. Defaults to WPA-PWK).
- **param tshark_path**: Path of the tshark binary
- **param output_file**: Additionally save captured packets to this file.

Reading from a live remote interface:

```
>>> capture = pyshark.RemoteCapture('192.168.1.101', 'eth0')
>>> capture.sniff(timeout=50)
>>> capture
```

Other options

- **param remote_host**: The remote host to capture on (IP or hostname). Should be running rpcapd.
- **param remote_interface**: The remote interface on the remote machine to capture on. Note that on windows it is not the device display name but the true interface name (i.e. \Device\NPF_..).
- **param remote_port**: The remote port the rpcapd service is listening on
- **param bpf_filter**: A BPF (tcpdump) filter to apply on the cap before reading.
- **param only_summaries**: Only produce packet summaries, much faster but includes very little information
- **param disable_protocol**: Disable detection of a protocol (tshark > version 2)
- **param decryption_key**: Key used to encrypt and decrypt captured traffic.
- **param encryption_type**: Standard of encryption used in captured traffic (must be either 'WEP', 'WPA-PWD', or 'WPA-PWK'. Defaults to WPA-PWK).
- **param tshark_path**: Path of the tshark binary

Accessing packet data:

Data can be accessed in multiple ways. Packets are divided into layers, first you have to reach the appropriate layer and then you can select your field.

All of the following work:

```
>>> packet['ip'].dst
192.168.0.1
>>> packet.ip.src
192.168.0.100
>>> packet[2].src
192.168.0.100
```

To test whether a layer is in a packet, you can use its name:

```
>>> 'IP' in packet
True
```

To see all possible field names, use the `packet.layer.field_names` attribute (i.e. `packet.ip.field_names`) or the autocomplete function on your interpreter.

You can also get the original binary data of a field, or a pretty description of it:

```
>>> p.ip.addr.showname
Source or Destination Address: 10.0.0.10 (10.0.0.10)
# And some new attributes as well:
>>> p.ip.addr.int_value
167772170
>>> p.ip.addr.binary_value
'\n\x00\x00\n'
```

Decrypting packet captures

Pyshark supports automatic decryption of traces using the WEP, WPA-PWD, and WPA-PSK standards (WPA-PWD is the default).

```
>>> cap1 = pyshark.FileCapture('/tmp/capture1.cap',
decryption_key='password')
>>> cap2 = pyshark.LiveCapture(interface='wi0', decryption_key='password',
encryption_type='wpa-psk')
```

A tuple of supported encryption standards, SUPPORTED_ENCRYPTION_STANDARDS, exists in each capture class.

```
>>> pyshark.FileCapture.SUPPORTED_ENCRYPTION_STANDARDS
('wep', 'wpa-pwd', 'wpa-psk')
>>> pyshark.LiveCapture.SUPPORTED_ENCRYPTION_STANDARDS
('wep', 'wpa-pwd', 'wpa-psk')
```

License