Packet Sniffer Docs

version

Author Name

February 04, 2025

Contents

Welcome to Packet Sniffer Documentation!

packetSniffer

filters module

`filters.`**`apply_filters`**(`packets`, `filter_type`, `value`)

Apply filters to the list of packets based on the filter type and value.

Parameters:

- **packets** (*list*) – List of packet dictionaries.
- **filter_type** (*str*) – Type of filter (‘host’, ‘port’, ‘ip’, ‘protocol’, ‘net’).
- **value** (*str*) – Filter value to apply.

Returns: Filtered list of packets.

Return type: list

parser module

`parser.`**`parse_pcap`**(`filename`)

Parses a pcap (Packet Capture) file and extracts Ethernet, IPv4, and IPv6 header details along with transport layer information (TCP/UDP).

The function reads packets from a given pcap file, processes each packet’s Ethernet, IP (IPv4 or IPv6), and transport layer (TCP/UDP) headers, and returns a list of dictionaries with the parsed packet details.

Parameters:

filename : *str*

The name of the pcap file to be parsed. The file should be in binary format.

Returns:

List[dict]

A list of dictionaries, where each dictionary contains the following key-value pairs for a single packet:

- “packet_number” (int): The packet number in the pcap file.
- “timestamp” (float): The timestamp of the packet.
- “packet_size” (int): The size of the packet in bytes.
- “src_mac” (str): The source MAC address in the format “xx:xx:xx:xx:xx:xx”.
- “dst_mac” (str): The destination MAC address in the format “xx:xx:xx:xx:xx:xx”.
- “eth_type” (int): The Ethernet type field (e.g., 0x0800 for IPv4, 0x86dd for IPv6).
- “ip_version” (int or None): The IP version (4 for IPv4, 6 for IPv6), or None if not applicable.
- “header_length” (int or None): The IP header length in bytes, or None if not applicable.
- “tos” (int or None): The Type of Service field in the IPv4 header, or None if not applicable.
- “total_length” (int or None): The total length of the IP packet, or None if not applicable.
- “identification” (int or None): The IP identification field, or None if not applicable.
- “flags” (int or None): The flags field in the IP header, or None if not applicable.
- “fragment_offset” (int or None): The fragment offset field in the IP header, or None if not applicable.
- “ttl” (int or None): The Time to Live (TTL) field in the IP header, or None if not applicable.

- “protocol” (str): The protocol used at the transport layer (e.g., “TCP”, “UDP”, “ICMP”), or a descriptive string for unknown protocols.
- “header_checksum” (int or None): The checksum of the IP header, or None if not applicable.
- “src_ip” (str or None): The source IP address (in IPv4 or IPv6 format), or None if not applicable.
- “dst_ip” (str or None): The destination IP address (in IPv4 or IPv6 format), or None if not applicable.
- “src_port” (int or None): The source port for TCP/UDP packets, or None if not applicable.
- “dst_port” (int or None): The destination port for TCP/UDP packets, or None if not applicable.

Notes:

- For Ethernet frames, the function assumes that the packets are either IPv4 or IPv6 packets.
- For IPv4 packets, both TCP and UDP transport layer protocols are supported.
- For IPv6 packets, only the transport layer protocols that are commonly used (TCP/UDP) are considered.
- The packet data for IPv4 and IPv6 is processed and the associated transport layer data is extracted when applicable.
- If the packet is not an IPv4 or IPv6 packet, the relevant IP-specific fields (such as *ip_version*, *src_ip*, *dst_ip*, etc.) will be set to *None*.

Example:

```
>>> parse_pcap("capture.pcap")
[{'packet_number': 1, 'timestamp': 1619190374.123456, 'packet_size': 84, 'src_mac': '00:14
    'dst_mac': '00:14:22:67:89:ab', 'eth_type': 2048, 'ip_version': 4, 'header_length': 20
    'tos': 0, 'total_length': 60, 'identification': 12345, 'flags': 2, 'fragment_offset': 
    'ttl': 64, 'protocol': 'TCP', 'header_checksum': 0x1234, 'src_ip': '192.168.1.1',
    'dst_ip': '192.168.1.2', 'src_port': 80, 'dst_port': 12345},
... ]
```

pktsniffer module

`pktsniffer.`**`main`**`()`

`pktsniffer.`**`print_packet_details`**`(packet)`

This function takes a packet (represented as a dictionary) and displays its details in a human-readable format, including Ethernet, IP, and transport layer information (TCP/UDP/ICMP). The packet’s number, timestamp, MAC addresses, protocol details, and other header information are printed for easy inspection.

Parameters:

packet : *dict*

A dictionary containing the parsed details of a network packet, including the following keys:

- “packet_number” (int): The packet number in the pcap file.
- “timestamp” (float): The timestamp of the packet.
- “packet_size” (int): The size of the packet in bytes.
- “src_mac” (str): The source MAC address in the format “xx:xx:xx:xx:xx:xx”.
- “dst_mac” (str): The destination MAC address in the format “xx:xx:xx:xx:xx:xx”.
- “eth_type” (int): The Ethernet type field (e.g., 0x0800 for IPv4, 0x86dd for IPv6).
- “ip_version” (int or None): The IP version (4 for IPv4, 6 for IPv6), or None if not applicable.
- “header_length” (int or None): The IP header length in bytes, or None if not applicable.

- “tos” (int or None): The Type of Service field in the IPv4 header, or None if not applicable.
- “total_length” (int or None): The total length of the IP packet, or None if not applicable.
- “identification” (int or None): The IP identification field, or None if not applicable.
- “flags” (int or None): The flags field in the IP header, or None if not applicable.
- “fragment_offset” (int or None): The fragment offset field in the IP header, or None if not applicable.
- “ttl” (int or None): The Time to Live (TTL) field in the IP header, or None if not applicable.
- “protocol” (str): The protocol used at the transport layer (e.g., “TCP”, “UDP”, “ICMP”).
- “header_checksum” (int or None): The checksum of the IP header, or None if not applicable.
- “src_ip” (str or None): The source IP address (in IPv4 or IPv6 format), or None if not applicable.
- “dst_ip” (str or None): The destination IP address (in IPv4 or IPv6 format), or None if not applicable.
- “src_port” (int or None): The source port for TCP/UDP packets, or None if not applicable.
- “dst_port” (int or None): The destination port for TCP/UDP packets, or None if not applicable.

Example:

```
>>> packet = {
        'packet_number': 1, 'timestamp': 1619190374.123456, 'packet_size': 84,
        'src_mac': '00:14:22:01:23:45', 'dst_mac': '00:14:22:67:89:ab', 'eth_type': 2048,
        'ip_version': 4, 'header_length': 20, 'tos': 0, 'total_length': 60, 'identificatio
        'flags': 2, 'fragment_offset': 0, 'ttl': 64, 'protocol': 'TCP', 'header_checksum':
        'src_ip': '192.168.1.1', 'dst_ip': '192.168.1.2', 'src_port': 80, 'dst_port': 1234
    }
>>> print_packet_details(packet)
```

utils module

utils.**format_ip**(address)

Convert an IP address to a readable format.

This function takes an IP address as a byte sequence (in IPv4 format) and returns a string representation of the address in the standard “xxx.xxx.xxx.xxx” format.

Parameters:

address : *bytes*

The IP address as a sequence of 4 bytes.

Returns:

str

The IP address as a formatted string, e.g., “192.168.1.1”.

Example:

```
>>> format_ip(b'À¨■■')
'192.168.1.1'
```

utils.**format_mac**(address)

Convert a MAC address to a readable format.

This function takes a MAC address as a byte sequence and returns a string representation of the address in the standard “xx:xx:xx:xx:xx:xx” format, where “xx” represents a two-digit hexadecimal value for each byte.

Parameters:

address : *bytes*

The MAC address as a sequence of 6 bytes.

Returns:

str

The MAC address as a formatted string, e.g., “00:1a:2b:3c:4d:5e”.

Example:

```
>>> format_mac(b'■+<M^')
'00:1a:2b:3c:4d:5e'
```

Module Documentation

`pktsniffer.`**`print_packet_details`**`(packet)`

This function takes a packet (represented as a dictionary) and displays its details in a human-readable format, including Ethernet, IP, and transport layer information (TCP/UDP/ICMP). The packet’s number, timestamp, MAC addresses, protocol details, and other header information are printed for easy inspection.

Parameters:

packet : *dict*

A dictionary containing the parsed details of a network packet, including the following keys:

- “packet_number” (int): The packet number in the pcap file.
- “timestamp” (float): The timestamp of the packet.
- “packet_size” (int): The size of the packet in bytes.
- “src_mac” (str): The source MAC address in the format “xx:xx:xx:xx:xx:xx”.
- “dst_mac” (str): The destination MAC address in the format “xx:xx:xx:xx:xx:xx”.
- “eth_type” (int): The Ethernet type field (e.g., 0x0800 for IPv4, 0x86dd for IPv6).
- “ip_version” (int or None): The IP version (4 for IPv4, 6 for IPv6), or None if not applicable.
- “header_length” (int or None): The IP header length in bytes, or None if not applicable.
- “tos” (int or None): The Type of Service field in the IPv4 header, or None if not applicable.
- “total_length” (int or None): The total length of the IP packet, or None if not applicable.
- “identification” (int or None): The IP identification field, or None if not applicable.
- “flags” (int or None): The flags field in the IP header, or None if not applicable.
- “fragment_offset” (int or None): The fragment offset field in the IP header, or None if not applicable.
- “ttl” (int or None): The Time to Live (TTL) field in the IP header, or None if not applicable.
- “protocol” (str): The protocol used at the transport layer (e.g., “TCP”, “UDP”, “ICMP”).
- “header_checksum” (int or None): The checksum of the IP header, or None if not applicable.
- “src_ip” (str or None): The source IP address (in IPv4 or IPv6 format), or None if not applicable.
- “dst_ip” (str or None): The destination IP address (in IPv4 or IPv6 format), or None if not applicable.
- “src_port” (int or None): The source port for TCP/UDP packets, or None if not applicable.
- “dst_port” (int or None): The destination port for TCP/UDP packets, or None if not applicable.

Example:

Example:

```
>>> packet = {
        'packet_number': 1, 'timestamp': 1619190374.123456, 'packet_size': 84,
        'src_mac': '00:14:22:01:23:45', 'dst_mac': '00:14:22:67:89:ab', 'eth_type': 2048,
        'ip_version': 4, 'header_length': 20, 'tos': 0, 'total_length': 60, 'identificatio
        'flags': 2, 'fragment_offset': 0, 'ttl': 64, 'protocol': 'TCP', 'header_checksum':
        'src_ip': '192.168.1.1', 'dst_ip': '192.168.1.2', 'src_port': 80, 'dst_port': 1234
    }
>>> print_packet_details(packet)
```

`parser.`**`parse_pcap`**`(filename)`

Parses a pcap (Packet Capture) file and extracts Ethernet, IPv4, and IPv6 header details along with transport layer information (TCP/UDP).

The function reads packets from a given pcap file, processes each packet's Ethernet, IP (IPv4 or IPv6), and transport layer (TCP/UDP) headers, and returns a list of dictionaries with the parsed packet details.

Parameters:

filename : *str*

The name of the pcap file to be parsed. The file should be in binary format.

Returns:

List[dict]

A list of dictionaries, where each dictionary contains the following key-value pairs for a single packet:

- "packet_number" (int): The packet number in the pcap file.
- "timestamp" (float): The timestamp of the packet.
- "packet_size" (int): The size of the packet in bytes.
- "src_mac" (str): The source MAC address in the format "xx:xx:xx:xx:xx:xx".
- "dst_mac" (str): The destination MAC address in the format "xx:xx:xx:xx:xx:xx".
- "eth_type" (int): The Ethernet type field (e.g., 0x0800 for IPv4, 0x86dd for IPv6).
- "ip_version" (int or None): The IP version (4 for IPv4, 6 for IPv6), or None if not applicable.
- "header_length" (int or None): The IP header length in bytes, or None if not applicable.
- "tos" (int or None): The Type of Service field in the IPv4 header, or None if not applicable.
- "total_length" (int or None): The total length of the IP packet, or None if not applicable.
- "identification" (int or None): The IP identification field, or None if not applicable.
- "flags" (int or None): The flags field in the IP header, or None if not applicable.
- "fragment_offset" (int or None): The fragment offset field in the IP header, or None if not applicable.
- "ttl" (int or None): The Time to Live (TTL) field in the IP header, or None if not applicable.
- "protocol" (str): The protocol used at the transport layer (e.g., "TCP", "UDP", "ICMP"), or a descriptive string for unknown protocols.
- "header_checksum" (int or None): The checksum of the IP header, or None if not applicable.
- "src_ip" (str or None): The source IP address (in IPv4 or IPv6 format), or None if not applicable.
- "dst_ip" (str or None): The destination IP address (in IPv4 or IPv6 format), or None if not applicable.
- "src_port" (int or None): The source port for TCP/UDP packets, or None if not applicable.
- "dst_port" (int or None): The destination port for TCP/UDP packets, or None if not applicable.

Notes:

- For Ethernet frames, the function assumes that the packets are either IPv4 or IPv6 packets.
- For IPv4 packets, both TCP and UDP transport layer protocols are supported.
- For IPv6 packets, only the transport layer protocols that are commonly used (TCP/UDP) are considered.
- The packet data for IPv4 and IPv6 is processed and the associated transport layer data is extracted when applicable.
- If the packet is not an IPv4 or IPv6 packet, the relevant IP-specific fields (such as *ip_version*, *src_ip*, *dst_ip*, etc.) will be set to *None*.

Example:

```
>>> parse_pcap("capture.pcap")
[{'packet_number': 1, 'timestamp': 1619190374.123456, 'packet_size': 84, 'src_mac': '00:14
    'dst_mac': '00:14:22:67:89:ab', 'eth_type': 2048, 'ip_version': 4, 'header_length': 20
    'tos': 0, 'total_length': 60, 'identification': 12345, 'flags': 2, 'fragment_offset': (
    'ttl': 64, 'protocol': 'TCP', 'header_checksum': 0x1234, 'src_ip': '192.168.1.1',
    'dst_ip': '192.168.1.2', 'src_port': 80, 'dst_port': 12345},
... ]
```

`filters.`**`apply_filters`**(`packets`, `filter_type`, `value`)

Apply filters to the list of packets based on the filter type and value.

Parameters:
- **packets** (*list*) – List of packet dictionaries.
- **filter_type** (*str*) – Type of filter (‘host’, ‘port’, ‘ip’, ‘protocol’, ‘net’).
- **value** (*str*) – Filter value to apply.

Returns: Filtered list of packets.

Return type: list

`utils.`**`format_ip`**(`address`)

Convert an IP address to a readable format.

This function takes an IP address as a byte sequence (in IPv4 format) and returns a string representation of the address in the standard “xxx.xxx.xxx.xxx” format.

Parameters:

address : *bytes*

The IP address as a sequence of 4 bytes.

Returns:

str

The IP address as a formatted string, e.g., “192.168.1.1”.

Example:

```
>>> format_ip(b'À¨■■')
'192.168.1.1'
```

`utils.`**`format_mac`**(`address`)

Convert a MAC address to a readable format.

This function takes a MAC address as a byte sequence and returns a string representation of the address in the standard “xx:xx:xx:xx:xx:xx” format, where “xx” represents a two-digit hexadecimal value for each byte.

Parameters:

Parameters:

address : *bytes*

The MAC address as a sequence of 6 bytes.

Returns:

str

The MAC address as a formatted string, e.g., “00:1a:2b:3c:4d:5e”.

Example:

```
>>> format_mac(b'■+<M^')
'00:1a:2b:3c:4d:5e'
```

Index

Python Module Index

f

filters

p

parser

pktsniffer

u

utils