NAME

text2pcap − Generate a capture file from an ASCII hexdump of packets

SYNOPSIS

text2pcap [**−a**] [**−d**] [**−D**] [**−e** <l3pid>] [**−h**] [**−i** <proto>] [**−l** <typenum>] [**−n**]
[**−N** <intf−name>] [**−m** <max−packet>] [**−o** hex|oct|dec] [**−q**] [**−s** <srcport>,<destport>,<tag>]
[**−S** <srcport>,<destport>,<ppi>] [**−t** <timefmt>] [**−T** <srcport>,<destport>]
[**−u** <srcport>,<destport>] [**−v**] [**−4** <srcip>,<destip>] [**−6** <srcip>,<destip>] <*infile*>|− <*outfile*>|−

DESCRIPTION

Text2pcap is a program that reads in an ASCII hex dump and writes the data described into a **pcap** or **pcapng** capture file. **text2pcap** can read hexdumps with multiple packets in them, and build a capture file of multiple packets. **text2pcap** is also capable of generating dummy Ethernet, IP and UDP, TCP, or SCTP headers, in order to build fully processable packet dumps from hexdumps of application−level data only.

Text2pcap understands a hexdump of the form generated by *od −Ax −tx1 −v*. In other words, each byte is individually displayed, with spaces separating the bytes from each other. Each line begins with an offset describing the position in the packet, each new packet starts with an offset of 0 and there is a space separating the offset from the following bytes. The offset is a hex number (can also be octal or decimal − see **−o**), of more than two hex digits.

Here is a sample dump that **text2pcap** can recognize:

```
000000 00 0e b6 00 00 02 00 0e b6 00 00 01 08 00 45 00
000010 00 28 00 00 00 00 ff 01 37 d1 c0 00 02 01 c0 00
000020 02 02 08 00 a6 2f 00 01 00 01 48 65 6c 6c 6f 20
000030 57 6f 72 6c 64 21
000036
```

Note the last byte must either be followed by the expected next offset value as in the example above or a space or a line−end character(s).

There is no limit on the width or number of bytes per line. Also the text dump at the end of the line is ignored. Bytes/hex numbers can be uppercase or lowercase. Any text before the offset is ignored, including email forwarding characters '>'. Any lines of text between the bytestring lines is ignored. The offsets are used to track the bytes, so offsets must be correct. Any line which has only bytes without a leading offset is ignored. An offset is recognized as being a hex number longer than two characters. Any text after the bytes is ignored (e.g. the character dump). Any hex numbers in this text are also ignored. An offset of zero is indicative of starting a new packet, so a single text file with a series of hexdumps can be converted into a packet capture with multiple packets. Packets may be preceded by a timestamp. These are interpreted according to the format given on the command line (see **−t**). If not, the first packet is timestamped with the current time the conversion takes place. Multiple packets are written with timestamps differing by one microsecond each. In general, short of these restrictions, **text2pcap** is pretty liberal about reading in hexdumps and has been tested with a variety of mangled outputs (including being forwarded through email multiple times, with limited line wrap etc.)

There are a couple of other special features to note. Any line where the first non−whitespace character is '#' will be ignored as a comment. Any line beginning with #TEXT2PCAP is a directive and options can be inserted after this command to be processed by **text2pcap**. Currently there are no directives implemented; in the future, these may be used to give more fine grained control on the dump and the way it should be processed e.g. timestamps, encapsulation type etc.

Text2pcap also allows the user to read in dumps of application−level data, by inserting dummy L2, L3 and L4 headers before each packet. The user can elect to insert Ethernet headers, Ethernet and IP, or Ethernet, IP and UDP/TCP/SCTP headers before each packet. This allows Wireshark or any other full−packet decoder to handle these dumps.

OPTIONS

−a

Enables ASCII text dump identification. It allows one to identify the start of the ASCII text dump and not include it in the packet even if it looks like HEX.

NOTE: Do not enable it if the input file does not contain the ASCII text dump.

−d

Displays debugging information during the process. Can be used multiple times to generate more debugging information.

−D

The text before the packet starts either with an I or O indicating that the packet is inbound or outbound. This is used when generating dummy headers. The indication is only stored if the output format is pcapng.

−e <l3pid>

Include a dummy Ethernet header before each packet. Specify the L3PID for the Ethernet header in hex. Use this option if your dump has Layer 3 header and payload (e.g. IP header), but no Layer 2 encapsulation. Example: *−e 0x806* to specify an ARP packet.

For IP packets, instead of generating a fake Ethernet header you can also use *−l 101* to indicate a raw IP packet to Wireshark. Note that *−l 101* does not work for any non−IP Layer 3 packet (e.g. ARP), whereas generating a dummy Ethernet header with *−e* works for any sort of L3 packet.

−h

Displays a help message.

−i <proto>

Include dummy IP headers before each packet. Specify the IP protocol for the packet in decimal. Use this option if your dump is the payload of an IP packet (i.e. has complete L4 information) but does not have an IP header with each packet. Note that an appropriate Ethernet header is automatically included with each packet as well. Example: *−i 46* to specify an RSVP packet (IP protocol 46). See https://www.iana.org/assignments/protocol−numbers/protocol−numbers.xhtml for the complete list of assigned internet protocol numbers.

−l

Specify the link−layer header type of this packet. Default is Ethernet (1). See https://www.tcpdump.org/linktypes.html for the complete list of possible encapsulations. Note that this option should be used if your dump is a complete hex dump of an encapsulated packet and you wish to specify the exact type of encapsulation. Example: *−l 7* for ARCNet packets encapsulated BSD−style.

−m <max−packet>

Set the maximum packet length, default is 262144. Useful for testing various packet boundaries when only an application level datastream is available. Example:

od −Ax −tx1 −v stream | text2pcap −m1460 −T1234,1234 − stream.pcap

will convert from plain datastream format to a sequence of Ethernet TCP packets.

−n

Write the file in pcapng format rather than pcap format.

−N <intf−name>

Specify a name for the interface included when writing a pcapng format file. By default no name is defined.

−o hex|oct|dec

Specify the radix for the offsets (hex, octal or decimal). Defaults to hex. This corresponds to the −A option for *od*.

−q

Be completely quiet during the process.

−s <srcport>,<destport>,<tag>

Include dummy SCTP headers before each packet. Specify, in decimal, the source and destination SCTP ports, and verification tag, for the packet. Use this option if your dump is the SCTP payload of a packet but does not include any SCTP, IP or Ethernet headers. Note that appropriate Ethernet and IP headers are automatically also included with each packet. A CRC32C checksum will be put into the SCTP header.

−S <srcport>,<destport>,<ppi>

Include dummy SCTP headers before each packet. Specify, in decimal, the source and destination SCTP ports, and a verification tag of 0, for the packet, and prepend a dummy SCTP DATA chunk header with a payload protocol identifier if *ppi*. Use this option if your dump is the SCTP payload of a packet but does not include any SCTP, IP or Ethernet headers. Note that appropriate Ethernet and IP headers are automatically included with each packet. A CRC32C checksum will be put into the SCTP header.

−t <timefmt>

Treats the text before the packet as a date/time code; *timefmt* is a format string of the sort supported by strptime(3). Example: The time "10:15:14.5476" has the format code "%H:%M:%S."

NOTE: The subsecond component delimiter must be specified (.) but no pattern is required; the remaining number is assumed to be fractions of a second.

NOTE: Date/time fields from the current date/time are used as the default for unspecified fields.

−T <srcport>,<destport>

Include dummy TCP headers before each packet. Specify the source and destination TCP ports for the packet in decimal. Use this option if your dump is the TCP payload of a packet but does not include any TCP, IP or Ethernet headers. Note that appropriate Ethernet and IP headers are automatically also

included with each packet. Sequence numbers will start at 0.

−u <srcport>,<destport>

Include dummy UDP headers before each packet. Specify the source and destination UDP ports for the packet in decimal. Use this option if your dump is the UDP payload of a packet but does not include any UDP, IP or Ethernet headers. Note that appropriate Ethernet and IP headers are automatically also included with each packet. Example: *−u1000,69* to make the packets look like TFTP/UDP packets.

−v

Print the version and exit.

−4 <srcip>,<destip>

Prepend dummy IP header with specified IPv4 dest and source address. This option should be accompanied by one of the following options: −i, −s, −S, −T, −u Use this option to apply "custom" IP addresses. Example: *−4 10.0.0.1,10.0.0.2* to use 10.0.0.1 and 10.0.0.2 for all IP packets.

−6 <srcip>,<destip>

Prepend dummy IP header with specified IPv6 dest and source address. This option should be accompanied by one of the following options: −i, −s, −S, −T, −u Use this option to apply "custom" IP addresses. Example: *−6 fe80::202:b3ff:fe1e:8329,2001:0db8:85a3::8a2e:0370:7334* to use fe80::202:b3ff:fe1e:8329 and 2001:0db8:85a3::8a2e:0370:7334 for all IP packets.

SEE ALSO

od(1), pcap(3), wireshark(1), tshark(1), dumpcap(1), mergecap(1), editcap(1), strptime(3), pcap−filter(7) or tcpdump(8)

NOTES

This is the manual page for **Text2pcap** 3.6.2. **Text2pcap** is part of the **Wireshark** distribution. The latest version of **Wireshark** can be found at https://www.wireshark.org.

AUTHORS

Original Author

Ashok Narayanan <ashokn[AT]cisco.com>