GIAC
CERTIFICATIONS

Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Network Monitoring and Threat Detection In-Depth (Security 503)"
at http://www.giac.org/registration/gcia

PCAP Next Generation: Is Your Sniffer Up to Snuff?

GIAC (GCIA) Gold Certification

Author: Scott D. Fether, scott.d.fether.mil@mail.mil
Advisor: Johannes Ullrich
Accepted: February 22, 2018

Abstract

The PCAP file format is widely used for packet capture within the network and security industry, but it is not the only standard. The PCAP Next Generation (PCAPng) Capture File Format is a refreshing improvement that adds extensibility, portability, and the ability to merge and append data to a wire trace. While Wireshark has led the way in supporting the new format, other tools have been slow to follow. With advantages such as the ability to capture from multiple interfaces, improved time resolution, and the ability to add per-packet comments, support for the PCAPng format should be developing more quickly than it has. This paper describes the new standard, displays methods to take advantage of new features, introduces scripting that can make the format useable, and makes the argument that migration to PCAPng is necessary.

© 2018 The SANS Institute, Author Retains Full Rights

© 2018 The SANS Institute Author retains full rights.

1. Introduction

The PCAP file format has been the de facto packet capture format for many security tools for years, but there is a new format that is gaining support. The PCAP Next Generation (PCAPng) Capture File Format is currently defined as an Internet Engineering Task Force (IETF) Internet-Draft and is a "work in progress" (Tuexen, Risso, Bongertz, Combs, & Harris, 2017). The format’s stated goals are to improve upon the PCAP format by adding extensibility, portability, and the ability to merge and append data to the file. PCAPng introduces more granular timestamps, the ability to capture traffic from multiple interfaces, fields for useful metadata, and additional statistics pertaining to dropped packets. PCAPng’s GitHub contains a list of its current implementations. While most applications can read the new format, fewer can write it, and even fewer use the format as their default. Libpcap, the library that the popular tcpdump relies on, does not support the writing of PCAPng. Only one fork of Scapy supports the ability to read PCAPng, and it cannot write the file ("PCAPng/PCAPng," 2017). This information makes it apparent that more application development is necessary if the security community is going to support PCAPng.

Of the applications listed on the PCAPng GitHub page, Wireshark supports PCAPng more than any other application. PCAPng is currently the default format for Wireshark and TShark and is the standard for packet captures. The support also extends to several of the tools included with Wireshark. Wireshark started using PCAPng as its default format in 2012 with the introduction of Wireshark 1.8. With one of the most popular packet capture tools using PCAPng as its default, one would think that other applications would also take advantage of the new features that PCAPng introduces. While support has increased over the years, movement to the new standard has been slow. Much of the delayed progress is attributed to the fact that many tools rely on libpcap library, which will be discussed further in Section 2.1.

With or without backwards compatibility, the lack of standardized support for one format over another can present problems for analysts that make frequent changes to capture files. Compatibility issues are particularly evident when analysts transfer captures from one security analyst to the next. If analysts do not have access to a compatible

© 2018 The SANS Institute Author retains full rights.

© 2018 The SANS Institute, Author Retains Full Rights

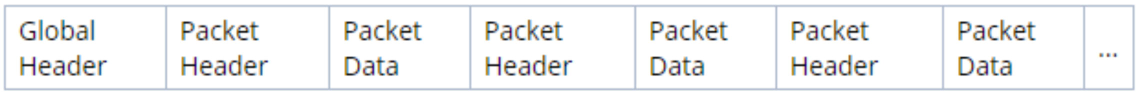
application, the additional features of a PCAPng file become useless. Nevertheless, conversion between formats is possible and is a common practice. Merging different formats can also be problematic, however. Important metadata included in PCAPng files can be lost or confused when merged with PCAP files. Undoubtedly, analysts have been in these situations before. Despite compatibility issues, developers aren't moving with urgency to update code to support the new format. It is possible that the community has not carefully considered the benefits of PCAPng. This paper will delve into features of the file format to show why it's necessary for the industry to move more aggressively towards supporting PCAPng. Tools that support the format will be used to demonstrate its practical use during real-world scenarios. Before those scenarios can be demonstrated, however, it's important to understand the differences between PCAP and PCAPng, and what added benefits PCAPng brings to the wire capture and analysis process.

2. File Format Comparison

2.1. PCAP

The PCAP file format is standardized with many popular networking tools such as tcpdump. This format is the most widely accepted capture file among network analysis tools. Libpcap is a portable C/C++ library for network traffic capture ("TCPDUMP & LIBPCAP", 2017). Packet capture files that often appear with the .pcap extension are referred to as either PCAP files or libpcap files. This section, will focus on the file format – not the portable library.

The general structure of a capture in the PCAP format is relatively simple. At the beginning of the capture file, there is a global header that contains information for the trace as a whole. This header only appears one time in the whole capture. It is followed by a packet header that contains information for the proceeding packet. Each frame has a separate packet header. The packet data that follows is a representation of the raw data that appeared on the network at the time of capture. Figure 1 shows the general structure of a PCAP file.



Author Name, email@addressmail.mil

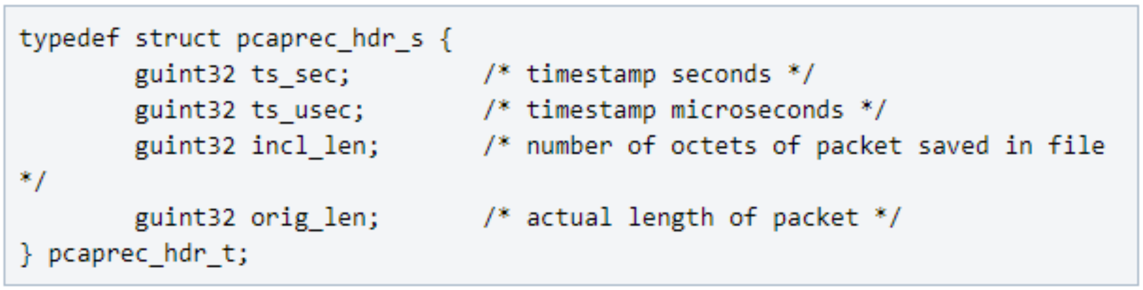
© 2018 The SANS Institute Author retains full rights.

© 2018 The SANS Institute Author Retains Full Rights

Figure 1 ("Development/LibpcapFileFormat", 2017) – PCAP Basic Structure

The global header provides a few useful pieces of information. The first four bytes of this field consist of the magic number, which identifies the file as a PCAP file. It also identifies the capture's "endianness." A Big Endian capture will use the value of 0xa1b2c3d4 while a Little Endian capture will use the swapped number of 0xd4c3b2a1. These bytes are important because the system must determine what byte order to expect for the remaining fields of the header. If the magic number is swapped, the fields that remain will be swapped as well ("Development/LibpcapFileFormat," 2017). The magic number is followed by the version number of the file format. Remaining fields include information about the local time zone for the capture, accuracy of timestamps, the length of the capture, and the link-layer header type.

Following the global header, each packet has a separate packet header. This header includes the following fields: ts_sec, ts_usec, incl_len, and orig_len. The ts_sec field identifies (in seconds) the date and time when the packet was captured. This is displayed in epoch time which is the value in seconds since January 1, 1970 00:00:00 GMT. The ts_usec field provides further granularity for packet capture time. Its value represents microseconds as an offset to the ts_sec field. The incl_len field represents the number of bytes that are saved in the file. The orig_len field is the length of the packet as it appeared on the network (not necessarily the amount of data that was saved). It's important to note that depending on the parameters set at the time of capture, the PCAP file may not save the entire packet. Data can be cut off if packets on the network are larger than the maximum size allowed at the time of capture ("Development/LibpcapFileFormat," 2017). Figure 2 provides a quick reference for the fields contained in the PCAP packet header.



```
typedef struct pcaprec_hdr_s {
        guint32 ts_sec;         /* timestamp seconds */
        guint32 ts_usec;        /* timestamp microseconds */
        guint32 incl_len;       /* number of octets of packet saved in file
*/
        guint32 orig_len;       /* actual length of packet */
} pcaprec_hdr_t;
```

Author Name, email@addressmail.mil

© 2018 The SANS Institute, Author Retains Full Rights

© 2018 The SANS Institute Author retains full rights.

Figure 2 (“Development/LibpcapFileFormat”, 2017) – Pcap Packet Header

The captured packet data will follow the packet header for the number of bytes that were specified in the incl_len field. In essence, the PCAP file format is simply a container for the captured data, providing little more than timestamps and link-layer protocol data. For years, this method has worked well for network analysts in the way of troubleshooting and network forensics. With the combination of timestamps and byte-level data capture, an analyst can parse the data and understand what is happening at the network level. In combination with protocol parsers, this has assisted in upper-layer troubleshooting, incident response, and network forensics. As a simple network capture file format, it has been implemented into many tools, but it is not without its drawbacks.

Because of its simplicity, the PCAP file format leaves some features to be desired. Time resolution, for example, does not necessarily fit the needs of high-speed network connections that are available in the modern environment. Increased time resolution in nanoseconds is not possible with this format. Another limitation is that the format cannot store packets for multiple interfaces of different link-layer types. It is not possible for PCAP to capture on an Ethernet and on a Wireless interface in the same instance. This can lead to the need to merge PCAPs which can cause timestamp collisions. Another shortcoming of the format is the lack of statistics for packet drops. There are no mechanisms in the format that provide this information. Another current feature that is desired today is the ability to add comments to the header. PCAP files are often used in incident response, and they can be passed from one analyst to another. In the case of PCAP files, these transfers sometimes require separate documentation to help the new analyst identify interesting traffic. A desirable feature would be the capability to include notes directly in the packet header, so analysts can collaborate more easily.

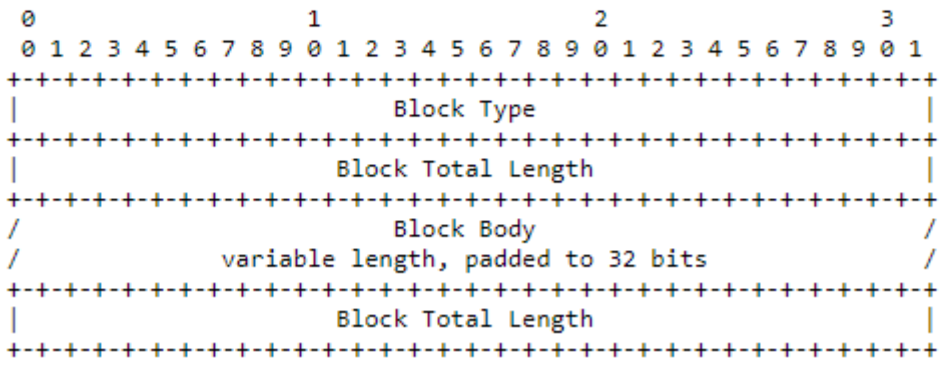
An answer to some of these drawbacks is the development of PCAPng. PCAPng provides a way to include more metadata in the packet capture itself, which can be useful to network and security analysts. Improved time resolution and the ability to capture from multiple interfaces are immediate improvements to the PCAP file format. In order to understand the capabilities of the new format, the specifics of the file structure must be considered.

© 2018 The SANS Institute Author retains full rights.

© 2018 The SANS Institute, Author Retains Full Rights

2.2. PCAP Next Generation

While the ultimate goal of capturing traffic on the wire is to view data at its most basic byte level, PCAPng places special importance on the reading and writing of a capture's metadata. Metadata is data that describes information about an aspect of the capture, and this is expressed in additional layers organized as blocks. PCAPng's Internet Draft (Tuexen, Risso, Bongertz, Combs, & Harris, 2017) describes the block types and their structures. There are four types of blocks: Enhanced Packet Block (EPB), Simple Packet Block (SPB), Name Resolution Block (NRB), and Custom Block (CB). Each block type follows a standard structure which allows for easy processing and provides a framework for the addition of new block types. Applications can skip blocks they may not know how to process or don't need. Figure 3 describes the basic block structure.



```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Block Type                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Block Total Length                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
/                          Block Body                           /
/              variable length, padded to 32 bits               /
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Block Total Length                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

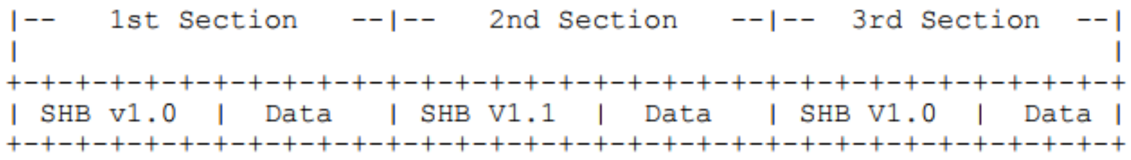
Figure 3 (Tuexen, Risso, Bongertz, Combs, & Harris, 2017) – Basic Block Structure

The Section Header Block (SHB) is mandatory and must appear at least once in each file. It identifies the beginning of a section in the capture file. Optional blocks that may appear in a file include the Interface Description Block (IDB), Enhanced Packet Block (EPB), Simple Packet Block (SPB), Name Resolution Block (NRB), Interface Statistics Block (ISB), and Custom Block (CB). This list is not all-inclusive, as there are several experimental blocks that are being considered by the PCAPng authors for future implementation (Tuexen, Risso, Bongertz, Combs, & Harris, 2017). Each of these blocks may contain a number of options. A file must begin with an SHB, but there may be

© 2018 The SANS Institute, Author Retains Full Rights

© 2018 The SANS Institute Author retains full rights.

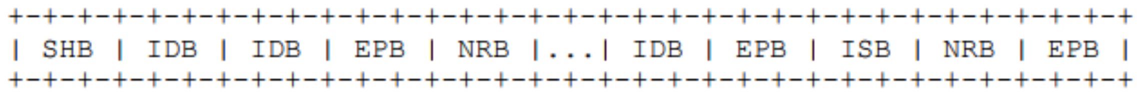
multiple SHBs in a capture. A "section" is defined as the data that is delimited by SHBs or the end of the file. It is likely that an analyst will see multiple SHB's when a file is merged. Figure 4 shows a graphical representation of how sections are identified.



```
|--   1st Section   --|--   2nd Section   --|--  3rd Section  --|
|                                                               |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| SHB v1.0  |  Data   | SHB V1.1  |  Data   | SHB V1.0  |  Data |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

Figure 4 (Tuexen, Risso, Bongertz, Combs, & Harris, 2017, p 7)

This graphic is a great example of the portability that PCAPng can bring to the table. This type of packet capture is likely the concatenation of three PCAPng files that came from different sources. Notice that two sections include a v1.0 SHB, but the middle section includes v1.1. This indicates that the capture occurred on different devices. The built-in portability features of the file format would permit an application that is only compatible with SHB v1.0 to read the first section, skip the SHB v1.1 section that it does not understand, and continue reading the third section. With so many blocks, the format is much more robust than its PCAP predecessor. The added complexity comes with capability, flexibility, and portability, however. A more complex block may look something like Figure 5 at its basic level. This graphic represents packets captured from three interfaces, the third of which begins after packets have arrived on other interfaces. It also includes Name Resolution Blocks (NSB) and Interface Statistics Block (ISB) (Tuexen, Risso, Bongertz, Combs, & Harris, 2017).



```
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| SHB | IDB | IDB | EPB | NRB |...| IDB | EPB | ISB | NRB | EPB |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

Figure 5 (Tuexen, Risso, Bongertz, Combs, & Harris, 2017, p 8)

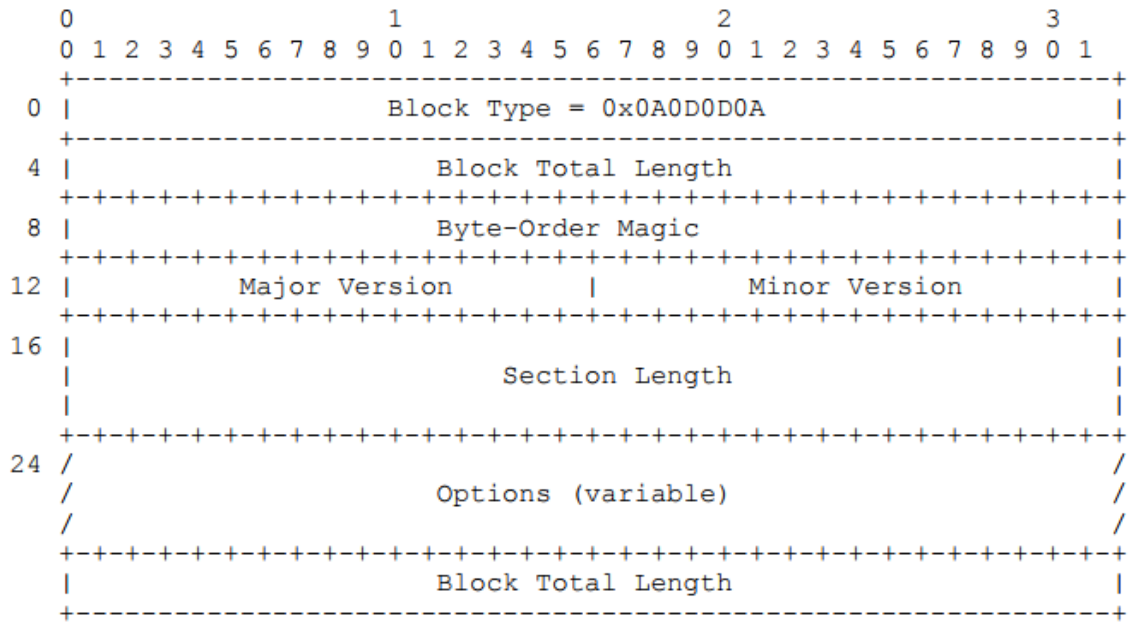
Each of the blocks may contain a number of options. While some blocks have options designed specifically for their block type, other options can be present in any option field. These universally acceptable options are opt_endofopt (code 0),

© 2018 The SANS Institute Author retains full rights.

© 2018 The SANS Institute, Author retains Full Rights

opt_comment (code 1), and opt_custom (code 2988/2989/19372/19373). The opt_endofopt option identifies the end of the optional fields and must be used only once within the list of options (Tuexen, Risso, Bongertz, Combs, & Harris, 2017). The opt_custom option is available for vendor-specific data. The opt_comment option, a UTF-8 string which contains human-readable text, is extremely useful because it describes something about that block. For example, an opt_comment option might be modified for a section that appears interesting to an analyst. The analyst might insert the text "This packet contains the text that triggered our snort signature! Investigate further!" With tools that can write to PCAPng files, this field can be modified and saved as part of the overall capture.

Now that the different types of blocks and options are understood, it's important to delve more deeply into the commonly seen blocks. This overview is not a complete description, however. To fully understand the structure of each, block, it is best to reference the internet draft in its original document. The Section Header Block (SHB) appears in every capture, so it's important to describe it in detail. Figure 6 shows the SHB Format.



```
                        1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +---------------------------------------------------------------+
 0 |                   Block Type = 0x0A0D0D0A                     |
   +---------------------------------------------------------------+
 4 |                      Block Total Length                       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 8 |                      Byte-Order Magic                         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
12 |          Major Version        |         Minor Version         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
16 |                                                               |
   |                          Section Length                       |
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
24 /                                                               /
   /                      Options (variable)                       /
   /                                                               /
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                      Block Total Length                       |
   +---------------------------------------------------------------+
```

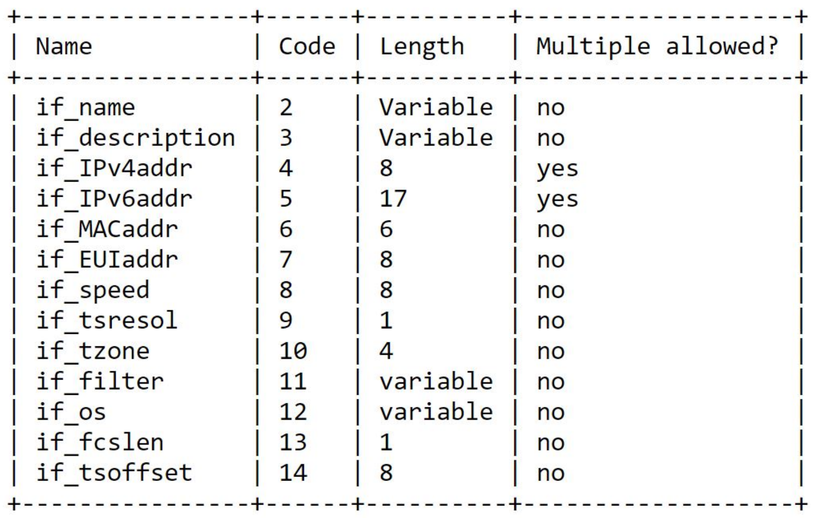
Figure 6 (Tuexen, Risso, Bongertz, Combs, & Harris, 2017, p 14) – SHB Format

© 2018 The SANS Institute Author retains full rights.

© 2018 SANS Institute Author Retains Full Rights

The SHB Block Type field is always expected to equal 0x0A0D0D0A for a file that is not corrupted. This can be used to identify SHBs when reading files in a hex editor. The Block Total Length field is self-explanatory. Byte-Order Magic is the magic number for PCAPng, and it is used to identify the "endianness" of the file. The value of this field is 0x1A2B3C4D. Major and minor version fields assist applications with identifying possible compatibility issues. The Section Length field identifies the length in octets of the section that follows. This value does not include the length of the SHB itself, and the field is padded to a 32-bit alignment. The options block is variable and can include any of the standard options that were previously mentioned. In addition to standard options, this field can also include options that identify the hardware name, operating system name, and the name of the application used to create the section. These SHB-specific options are identified as shb_hardware (code 2), shb_os (code 3), and shb_userappl (code 4).

Another common block is the Interface Description Block (IDB). This block is mandatory for most captures, and it contains information about the interface where the capture occurred. This is one of the components that allows PCAPng to capture on multiple interfaces at the same time. The IDB provides a mechanism to distinguish between interfaces. There are a number of options for the IDB that are useful to other blocks within the file including IP addresses, MAC addresses, speed of the interface, time resolution, capture filters, and operating system information. These option fields can also be valuable for the application that is reading the file. Figure 7 shows the options that are valid within the IDB.



Name	Code	Length	Multiple allowed?
if_name	2	Variable	no
if_description	3	Variable	no
if_IPv4addr	4	8	yes
if_IPv6addr	5	17	yes
if_MACaddr	6	6	no
if_EUIaddr	7	8	no
if_speed	8	8	no
if_tsresol	9	1	no
if_tzone	10	4	no
if_filter	11	variable	no
if_os	12	variable	no
if_fcslen	13	1	no
if_tsoffset	14	8	no

© 2018 The SANS Institute Author retains full rights.

© 2018 The SANS Institute, Author Retains Full Rights

Figure 7 (Tuexen, Risso, Bongertz, Combs, & Harris, 2017, p 18) – IDB Options

The if_name option shows the name of the device that captured the information. Since one PCAPng file can consist of captures from multiple interfaces, the if_name option is important for distinguishing between interfaces. The next few options also contain identifying information such as if_description, if_IPv4addr, if_IPv6addr, if_MACaddr, and if_EUIaddr. The speed of the interface is identified in the if_speed option which consists of a 64-bit decimal number representing speed in bits per second. Notably, the if_speed option only shows the speed of the interface performing the capture. It does not define the time resolution of timestamps. Time resolution is contained in the next option, if_tsresol.

The if_tsresol option is significant because adjusting this option can provide an important improvement over PCAP's current capability. PCAP can only provide time resolution in microseconds (10^-6 seconds) which results in 999,999 packets per second. This can make it difficult for PCAP to distinguish true timing between frames captured even on a basic 1Gbps interface (Walls, 2012). As described by Walls:

> The PCAPng format overcomes PCAP's time resolution limitation by defining a flexible format that can be used to adjust the resolution. Timestamps are now expressed as a single 64-bit time unit, representing the number of time units since 1/1/1970. An associated resolution field (if_tsresol) specifies what the time units mean. (Walls, 2012)

The default value for this option is still 10^-6 seconds, but it can be adjusted to a much deeper resolution. As this option can express timestamps in nanoseconds, a much-needed improvement for capturing on high-speed links is provided.

Another option available in this block is the if_timezone option, which identifies the timezone for the capture. The if_filter option describes the filter that might have been used to capture the traffic. If a capture filter was used to define only interesting traffic, that filter information is defined directly in this optional field. The if_os option identifies the operating system of the interface that performed the capture. Interestingly enough, this can be different than the OS information provided in the SHB because the capture could have been performed on a remote machine. Finally, the if_fcslen provides the

© 2018 The SANS Institute, Author Retains Full Rights

© 2018 The SANS Institute Author retains full rights.

length of the Frame Check Sequence for this interface, and the if_tsoffset provides an offset for obtaining an absolute timestamp if one is necessary (Tuexen, Risso, Bongertz, Combs, & Harris, 2017, p 20).

This is a great deal of information to take in, but just in this one block, PCAPng provides some drastic improvements over the PCAP file format. It provides a way for an analyst to capture on multiple interfaces and identify unique captures after they have been merged. It also allows the analyst to increase timestamp resolution for capture on high-speed interfaces. The ability to identify whether the capture was performed remotely is quite possible when comparing OS information between the SHB and the IDB. Address information is scalable between both IPv4 and IPv6 interfaces. The additional data can assist an analyst in discovering the source of a capture.

As the basic structure of a capture in PCAPng and a description of some of the important blocks for capturing metadata has been presented, the next sections will focus on the block that contains the captured data. There are two blocks that contain the data coming from the network: the Enhanced Packet Block (EPB) and the Simple Packet Block (SPB). The EPB is a fully-featured block and will provide the most information about the captured data. The SPB is intended for use in a resource-constrained environment because it is lightweight and easier to process. It is possible to see both types of blocks in a capture because a tool could be configured to switch between the two based on available resources (Tuexen, Risso, Bongertz, Combs, & Harris, 2017, p 25). More specifically, the following examples will focus on the EPB. The structure of the EPB is displayed in Figure 8 below.

© 2018 The SANS Institute, Author Retains Full Rights

© 2018 The SANS Institute Author retains full rights.

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +---------------------------------------------------------------+
 0 |                    Block Type = 0x00000006                    |
   +---------------------------------------------------------------+
 4 |                      Block Total Length                       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 8 |                         Interface ID                          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
12 |                        Timestamp (High)                       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
16 |                        Timestamp (Low)                        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
20 |                    Captured Packet Length                     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
24 |                    Original Packet Length                     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
28 /                                                               /
   /                          Packet Data                          /
   /              variable length, padded to 32 bits               /
   /                                                               /
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   /                                                               /
   /                      Options (variable)                       /
   /                                                               /
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                      Block Total Length                       |
   +---------------------------------------------------------------+
```

Figure 8 (Tuexen, Risso, Bongertz, Combs, & Harris, 2017, p 22) – EPB Format

The field for Interface ID must match the number provided in the IDB within the current section of the capture. This is how a frame can be tied to a particular capture interface. The Packet Data field will include the captured data from the network including link-layer headers. Options available include epb_flags, epb_hash, and epb_dropcount. The epb_hash is provided for integrity as data is being transferred from device to device. The epb_dropcount option specifies the number of packets lost between this particular packet and the one that was captured before it. This is another statistic that can be very useful for an analyst since the epb_flags option provides link layer information such as whether the packet was inbound or outbound, broadcast or multicast, or whether there are any link-layer errors identified (Tuexen, Risso, Bongertz, Combs, & Harris, 2017, p 25).

The fields and options that have been described this far are the most common makings of the PCAPng file format. There are many more options available, but the

© 2018 The SANS Institute Author retains full rights.

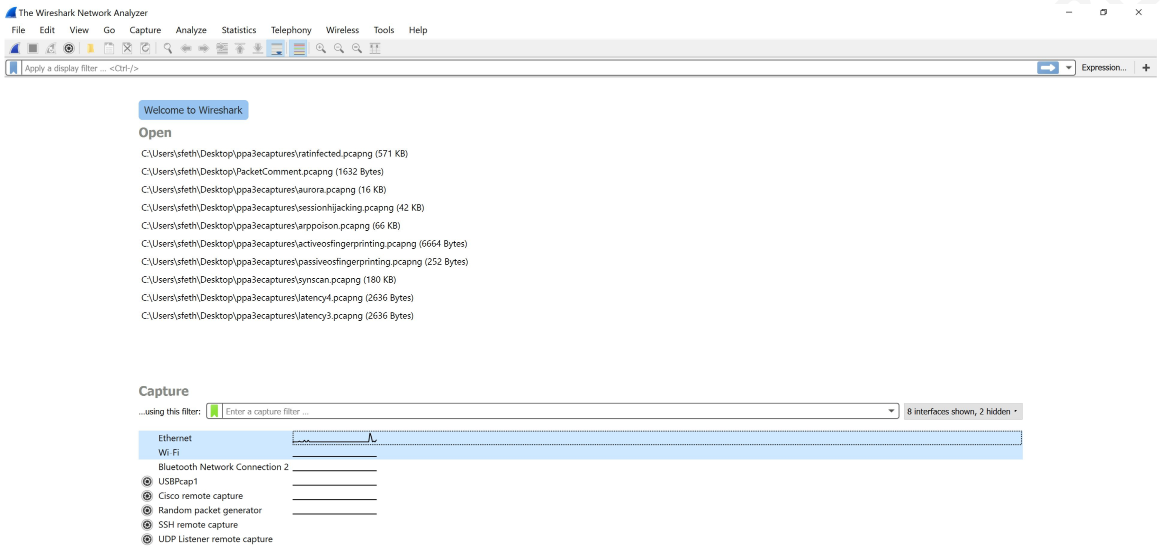
SHB, IDB, EPB blocks make up the nuts and bolts of the new standard. These blocks alone provide the ability to capture on multiple interfaces, increase time resolution on timestamps, and add comments on a per-packet basis, which is a drastic improvement on the PCAP format. Additional blocks provide even more capability such as better processing for name resolution (Name Resolution Block), metadata for capture statistics (Interface Statistics Block), information about compression (Compression Block), encryption information (Encryption Block), or even information about events or security alerts (Event/Security Block). The file format's capability increases daily, and its added benefits will surely impact the network and security communities in a positive way. Tools used to read and write the file format must improve, but for now, Wireshark leads the way.

3. Wireshark Application

Wireshark started as a project called Ethereal which was released in 1998 under the GNU Public License by Gerald Combs. Ethereal was rebranded in 2006 under the name Wireshark, and today it has more than 500 developers who actively contribute to the project (Sanders, 2017, pp. 37-38). Although Wireshark is popular for its easy-to-navigate GUI, a typical install includes TShark which is a full-featured command line version of Wireshark. TShark has all the same capabilities that Wireshark does, as it can take advantage of the ability to read and write PCAPng. It also comes with tools such as EditCap, MergeCap, ReorderCap, which assist in editing and manipulating captures. Wireshark's popular GUI makes it easy to take advantage of the features PCAPng introduces. For example, Figure 9 shows the ability to capture from multiple interfaces. An analyst can choose one of the active interfaces on a machine, hold CTRL, and click on a second interface. Wireshark will capture traffic from both interfaces at the same time – a feature not possible without PCAPng's ability to distinguish between interface ID's.

© 2018 The SANS Institute Author retains full rights.

© 2018 The SANS Institute, Author Retains Full Rights


The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
Welcome to Wireshark

Open

C:\Users\sfeth\Desktop\ppa3ecaptures\ratinfected.pcapng (571 KB)
C:\Users\sfeth\Desktop\PacketComment.pcapng (1632 Bytes)
C:\Users\sfeth\Desktop\ppa3ecaptures\aurora.pcapng (16 KB)
C:\Users\sfeth\Desktop\ppa3ecaptures\sessionhijacking.pcapng (42 KB)
C:\Users\sfeth\Desktop\ppa3ecaptures\arppoison.pcapng (66 KB)
C:\Users\sfeth\Desktop\ppa3ecaptures\activeosfingerprinting.pcapng (6664 Bytes)
C:\Users\sfeth\Desktop\ppa3ecaptures\passiveosfingerprinting.pcapng (252 Bytes)
C:\Users\sfeth\Desktop\ppa3ecaptures\synscan.pcapng (180 KB)
C:\Users\sfeth\Desktop\ppa3ecaptures\latency4.pcapng (2636 Bytes)
C:\Users\sfeth\Desktop\ppa3ecaptures\latency3.pcapng (2636 Bytes)

Capture

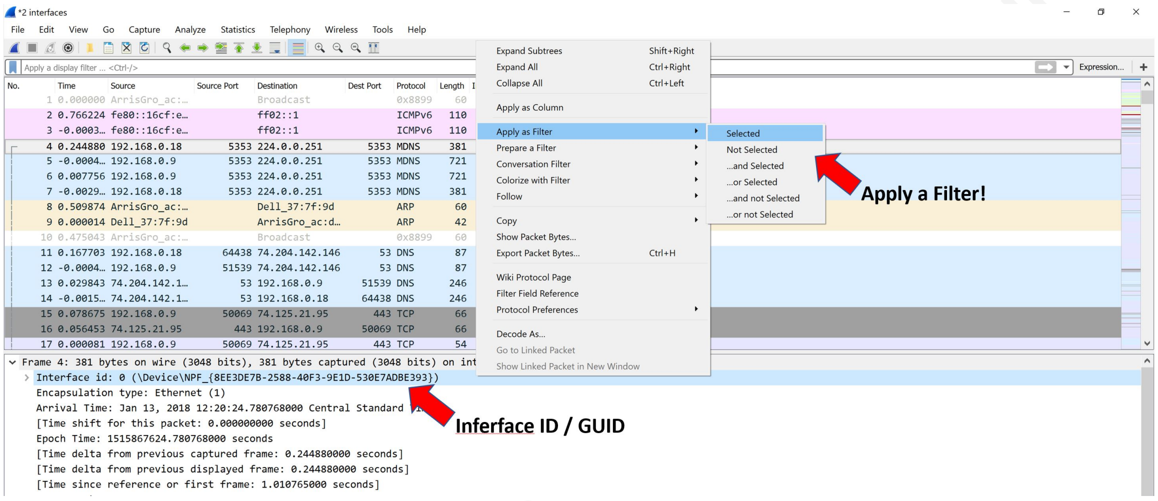
...using this filter: Enter a capture filter ... 8 interfaces shown, 2 hidden

Ethernet
Wi-Fi
Bluetooth Network Connection 2
USBPcap1
Cisco remote capture
Random packet generator
SSH remote capture
UDP Listener remote capture

Figure 9 – Perform a Capture

In this case, the analyst can capture on the wired and wireless interfaces at the same time which saves a significant amount of time. These examples are executed on a home computer, it is evident how easily this concept could be applied in a situation where the behavior of two networks, separated by physical interfaces, might need to be captured at the same time. Distinguishing the traffic is as simple as applying a display filter when the capture is complete. Figure 10 shows that each packet is given an interface ID – a direct benefit of the fields available in the Interface Description Block (IDB) and Enhanced Packet Block (EPB) provided by PCAPng. Because this information exists in the capture, analysts can distinguish between the two interfaces and capture at the same time. Timestamps can be compared with more accuracy using this method, and merging becomes less necessary. Filtering the capture to display information from only one interface is simple. As displayed in Figure 10, an analyst can simply right-click on the interface ID and choose to filter on that interface alone. This is considered a "display filter," and can be defined at any time after a capture is complete. Wireshark also provides the ability to apply a "capture filter," which tells the application to save only the information allowed by the filter. For example, one could sniff a wireless network, but choose to save only http traffic to the PCAPng produced from the sniffing session.

© 2018 The SANS Institute Author retains full rights.

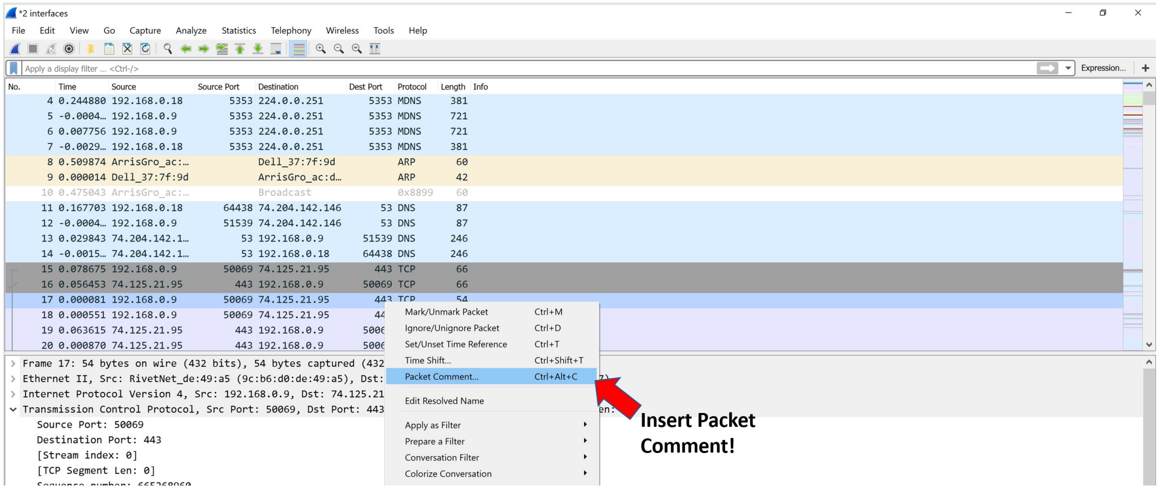


```
No.   Time     Source          Source Port Destination      Dest Port Protocol Length
    1 0.000000 ArrisGro_ac:…               Broadcast                  0x8899     60
    2 0.766224 fe80::16cf:e…               ff02::1                    ICMPv6    110
    3 -0.0003… fe80::16cf:e…               ff02::1                    ICMPv6    110
    4 0.244880 192.168.0.18           5353 224.0.0.251           5353 MDNS      381
    5 -0.0004… 192.168.0.9            5353 224.0.0.251           5353 MDNS      721
    6 0.007756 192.168.0.9            5353 224.0.0.251           5353 MDNS      721
    7 -0.0029… 192.168.0.18           5353 224.0.0.251           5353 MDNS      381
    8 0.509874 ArrisGro_ac:…               Dell_37:7f:9d              ARP        60
    9 0.000014 Dell_37:7f:9d               ArrisGro_ac:d…             ARP        42
   10 0.475043 ArrisGro_ac:…               Broadcast                  0x8899     60
   11 0.167703 192.168.0.18          64438 74.204.142.146          53 DNS        87
   12 -0.0004… 192.168.0.9           51539 74.204.142.146          53 DNS        87
   13 0.029843 74.204.142.1…            53 192.168.0.9          51539 DNS       246
   14 -0.0015… 74.204.142.1…            53 192.168.0.18         64438 DNS       246
   15 0.078675 192.168.0.9           50069 74.125.21.95           443 TCP        66
   16 0.056453 74.125.21.95            443 192.168.0.9          50069 TCP        66
   17 0.000081 192.168.0.9           50069 74.125.21.95           443 TCP        54
```

```
v Frame 4: 381 bytes on wire (3048 bits), 381 bytes captured (3048 bits) on int
  > Interface id: 0 (\Device\NPF_{8EE3DE7B-2588-40F3-9E1D-530E7ADBE393})
    Encapsulation type: Ethernet (1)
    Arrival Time: Jan 13, 2018 12:20:24.780768000 Central Standard Ti
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1515867624.780768000 seconds
    [Time delta from previous captured frame: 0.244880000 seconds]
    [Time delta from previous displayed frame: 0.244880000 seconds]
    [Time since reference or first frame: 1.010765000 seconds]
```

Figure 10 - Filter on Interface ID

Another capability that is beneficial to the PCAPng file format is making per-packet comments, as previously mentioned. The opt_comment option allows the file format to accept a UTF-8 human-readable string. Since each packet has a header capable of being edited by an upper-level application, it's easy to place analytical notes directly into a PCAPng file. Wireshark takes advantage of this capability, and comments can be added directly to a packet. By simply right-clicking on any packet the user will be given the option to add a comment. Figure 11 shows the simple act of adding comments to packet number 17, the completion of a TCP 3-Way Handshake.


*2 interfaces
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
Mark/Unmark Packet Ctrl+M
Ignore/Unignore Packet Ctrl+D
Set/Unset Time Reference Ctrl+T
Time Shift... Ctrl+Shift+T
Packet Comment... Ctrl+Alt+C
Edit Resolved Name
Apply as Filter
Prepare a Filter
Conversation Filter
Colorize Conversation
Insert Packet Comment!

```
No.   Time     Source          Source Port Destination      Dest Port Protocol Length Info
    4 0.244880 192.168.0.18           5353 224.0.0.251           5353 MDNS      381
    5 -0.0004… 192.168.0.9            5353 224.0.0.251           5353 MDNS      721
    6 0.007756 192.168.0.9            5353 224.0.0.251           5353 MDNS      721
    7 -0.0029… 192.168.0.18           5353 224.0.0.251           5353 MDNS      381
    8 0.509874 ArrisGro_ac:…               Dell_37:7f:9d              ARP        60
    9 0.000014 Dell_37:7f:9d               ArrisGro_ac:d…             ARP        42
   10 0.475043 ArrisGro_ac:…               Broadcast                  0x8899     60
   11 0.167703 192.168.0.18          64438 74.204.142.146          53 DNS        87
   12 -0.0004… 192.168.0.9           51539 74.204.142.146          53 DNS        87
   13 0.029843 74.204.142.1…            53 192.168.0.9          51539 DNS       246
   14 -0.0015… 74.204.142.1…            53 192.168.0.18         64438 DNS       246
   15 0.078675 192.168.0.9           50069 74.125.21.95           443 TCP        66
   16 0.056453 74.125.21.95            443 192.168.0.9          50069 TCP        66
   17 0.000081 192.168.0.9           50069 74.125.21.95           443 TCP        54
   18 0.000551 192.168.0.9           50069 74.125.21.95           44
   19 0.063615 74.125.21.95            443 192.168.0.9           5006
   20 0.000870 74.125.21.95            443 192.168.0.9           5006
```

```
> Frame 17: 54 bytes on wire (432 bits), 54 bytes captured (432
> Ethernet II, Src: RivetNet_de:49:a5 (9c:b6:d0:de:49:a5), Dst:
> Internet Protocol Version 4, Src: 192.168.0.9, Dst: 74.125.21
v Transmission Control Protocol, Src Port: 50069, Dst Port: 443
    Source Port: 50069
    Destination Port: 443
    [Stream index: 0]
    [TCP Segment Len: 0]
```

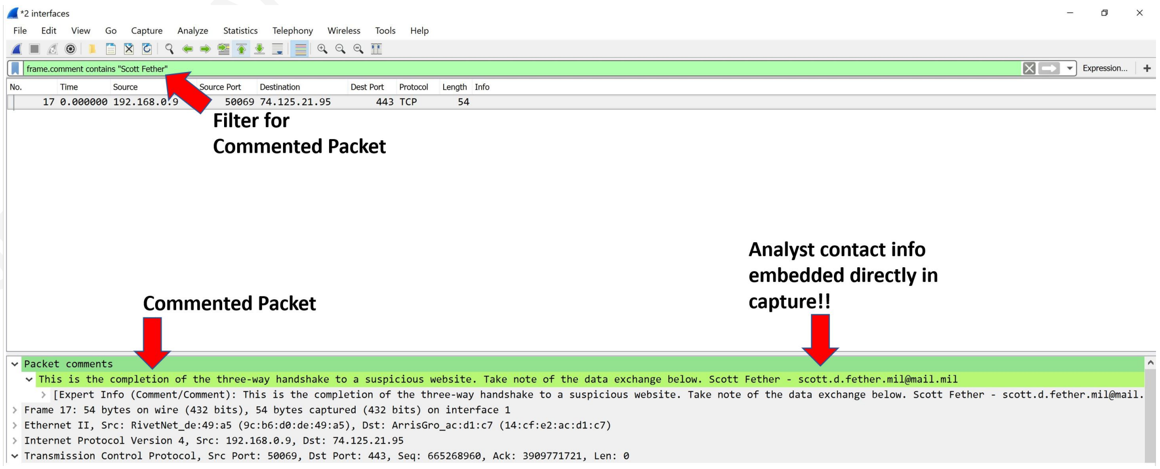
Figure 11 - Packet Comment

Author Name, email@addressmail.mil

© 2018 The SANS Institute, Author Retains Full Rights

© 2018 SANS Institute, Author Retains Full Rights

For this example, a comment has been added to packet 17 that includes a name and email address. This shows how an analyst would take advantage of the commenting ability that PCAPng provides. If an analyst expects that other analysts will view their comments, he could make it standard practice to add contact information to those comments. One might even desire to add comments to each packet to identify who captured the packet and when the capture took place. This would ensure that even when the packet capture gets split up into multiple files, each packet shows the time, date, and circumstance of the capture. This would contribute to shared information and the ability for analysts to collaborate. Once a comment has been applied, a great benefit is the ability to filter on those comments. In this example, since this particular comment contains a name, the analyst can just filter for any packet that contains a frame comment with the name he or she seeks. This is shown in Figure 12.


*2 interfaces
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Expression...
Filter for Commented Packet
Commented Packet
Analyst contact info embedded directly in capture!!

```
frame.comment contains "Scott Fether"

No.    Time     Source          Source Port  Destination   Dest Port  Protocol  Length  Info
    17 0.000000 192.168.0.9         50069 74.125.21.95          443 TCP           54
```

```
⌄ Packet comments
  ⌄ This is the completion of the three-way handshake to a suspicious website. Take note of the data exchange below. Scott Fether - scott.d.fether.mil@mail.mil
    › [Expert Info (Comment/Comment): This is the completion of the three-way handshake to a suspicious website. Take note of the data exchange below. Scott Fether - scott.d.fether.mil@mail.
› Frame 17: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 1
› Ethernet II, Src: RivetNet_de:49:a5 (9c:b6:d0:de:49:a5), Dst: ArrisGro_ac:d1:c7 (14:cf:e2:ac:d1:c7)
› Internet Protocol Version 4, Src: 192.168.0.9, Dst: 74.125.21.95
⌄ Transmission Control Protocol, Src Port: 50069, Dst Port: 443, Seq: 665268960, Ack: 3909771721, Len: 0
```

Figure 12 – Filter for Commented Packet

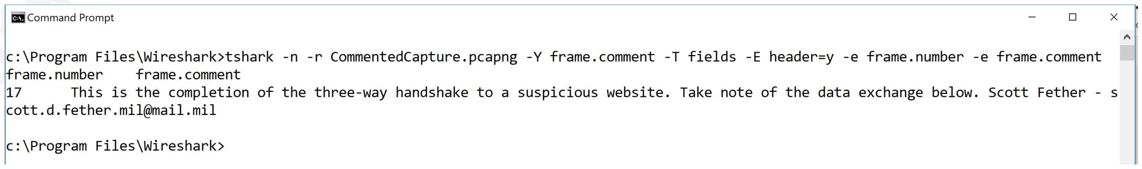
The impressive part about these features is that they are contained within the file format. These features are portable, and they are carried along as a PCAPng file is transported from application to application. If the application supports the reading of the PCAPng file format, comments will carry over as part of that file. TShark can be implemented to further display how analysts can take advantage of

© 2018 The SANS Institute Author retains full rights.

these new features. TShark is Wireshark's command line version of Wireshark. Because it is CLI, users of tcpdump will feel comfortable learning TShark. TShark has the added advantage of being able to read and write to the PCAPng file format which brings all the additional features to the table.

3.1 TShark and other Command Line Tools

TShark has many options and display capabilities that can manipulate the display of a PCAPng file based on an analyst's need. The intent of this section is not to teach the reader how to use TShark, but to show the portability and usefulness of PCAPng. If the reader feels the need to understand the options used in these commands, it is recommended to read the TShark man-page available on Wireshark's website. The capture used in the previous section was saved to a file called CommentedCaputre.pcapng and was moved into the directory where TShark is installed. A simple TShark command that displays commented packets shows that the packet comment used in this example was transported along with the PCAPng file. It is readable by TShark and would be readable to any application that can read the file format, as Figure 13 shows. As expected, Frame 17 contains the comment created in the Wireshark GUI.



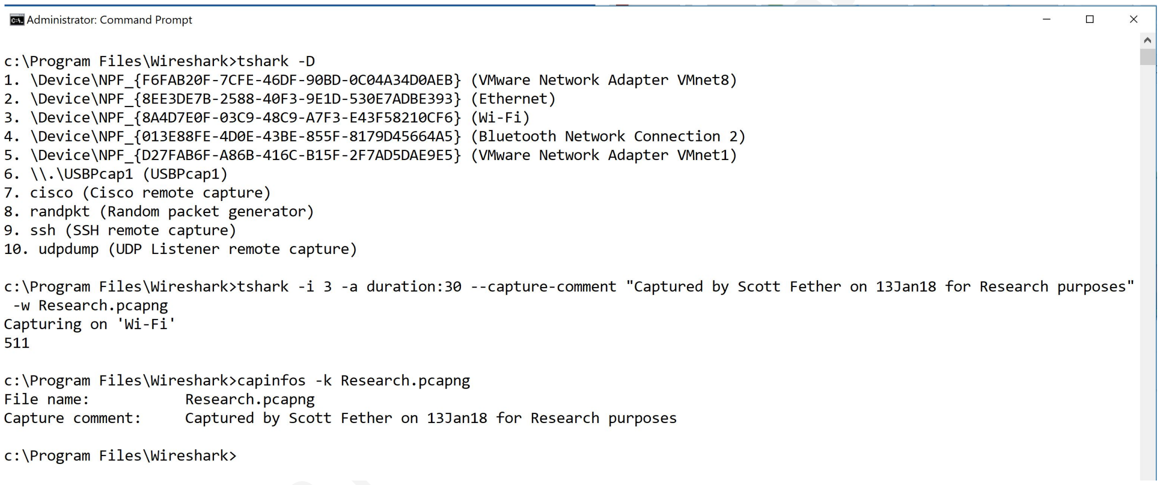
```
c:\Program Files\Wireshark>tshark -n -r CommentedCapture.pcapng -Y frame.comment -T fields -E header=y -e frame.number -e frame.comment
frame.number    frame.comment
17      This is the completion of the three-way handshake to a suspicious website. Take note of the data exchange below. Scott Fether - s
cott.d.fether.mil@mail.mil

c:\Program Files\Wireshark>
```

Figure 13 – Portable Comments

As demonstrated, TShark can read PCAPng files with comments. TShark can also can write a comment to an entire capture when TShark is the tool being used to perform the capture. This is different than a per-packet comment, however, and if a capture gets split into two files at a later time, this metadata might be lost. As an example of TShark's implementation of a capture comment, Figure 14 shows a typical capture process. The first command uses the "-D" option which displays all the interfaces available on which to capture. In this case, interface 3 is used which is

© 2018 The SANS Institute Author retains full rights.

WiFi. Using a typical capture command along with the "—capture-comment" option, TShark can take advantage of the ability to comment on a capture as a whole. An analyst then has the opportunity to add his or her name, date, and purpose of capture. Finally, using another tool that is provided with Wireshark called CapInfos, the comment that was created during the capture is displayed.

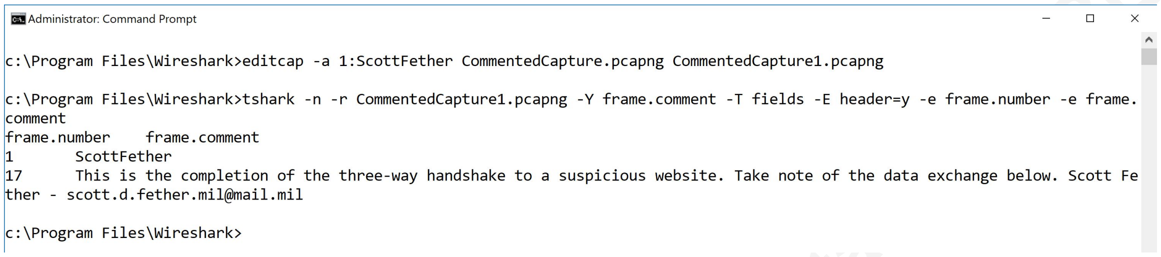


```
c:\Program Files\Wireshark>tshark -D
1. \Device\NPF_{F6FAB20F-7CFE-46DF-90BD-0C04A34D0AEB} (VMware Network Adapter VMnet8)
2. \Device\NPF_{8EE3DE7B-2588-40F3-9E1D-530E7ADBE393} (Ethernet)
3. \Device\NPF_{8A4D7E0F-03C9-48C9-A7F3-E43F58210CF6} (Wi-Fi)
4. \Device\NPF_{013E88FE-4D0E-43BE-855F-8179D45664A5} (Bluetooth Network Connection 2)
5. \Device\NPF_{D27FAB6F-A86B-416C-B15F-2F7AD5DAE9E5} (VMware Network Adapter VMnet1)
6. \\.\USBPcap1 (USBPcap1)
7. cisco (Cisco remote capture)
8. randpkt (Random packet generator)
9. ssh (SSH remote capture)
10. udpdump (UDP Listener remote capture)

c:\Program Files\Wireshark>tshark -i 3 -a duration:30 --capture-comment "Captured by Scott Fether on 13Jan18 for Research purposes"
 -w Research.pcapng
Capturing on 'Wi-Fi'
511

c:\Program Files\Wireshark>capinfos -k Research.pcapng
File name:           Research.pcapng
Capture comment:     Captured by Scott Fether on 13Jan18 for Research purposes

c:\Program Files\Wireshark>
```

Figure 14 – TShark Capture with Capture Comment

In order to add per-packet comments to a capture, one must use EditCap, which is another tool that comes with a basic Wireshark installation. EditCap can add, delete or modify information on a previously saved PCAPng file. For example, if an analyst wants to comment his name on the very first packet in the capture that was edited in Wireshark, he can use EditCap as shown in Figure 15. With the "-a" option in EditCap, an analyst can add a comment to one frame. Now instead of having just one comment on Frame 17, there is also a comment on Frame 1. Unfortunately, there is no option to add the same comment to a range of frames. For example, it may be desirable to add capture information to each frame, which would ensure that the information would survive most variations done in EditCap. Currently, there is no easy way to do this.

© 2018 The SANS Institute Author retains full rights.



```
c:\Program Files\Wireshark>editcap -a 1:ScottFether CommentedCapture.pcapng CommentedCapture1.pcapng

c:\Program Files\Wireshark>tshark -n -r CommentedCapture1.pcapng -Y frame.comment -T fields -E header=y -e frame.number -e frame.
comment
frame.number    frame.comment
1       ScottFether
17      This is the completion of the three-way handshake to a suspicious website. Take note of the data exchange below. Scott Fe
ther - scott.d.fether.mil@mail.mil

c:\Program Files\Wireshark>
```

Figure 15 – Comments with EditCap

The TShark and EditCap tools both have major limitations. TShark, for example, can only write comments on a per-capture basis. Even this "per-capture" commenting capability is limited because it can only be used at the time of capture. The proper use of this ability was displayed in Figure 14. TShark cannot add comments to a previously saved file. TShark is not able to write per-packet comments to a PCAPng file. These limitations require the use of multiple tools within a command-line environment to take advantage of the commenting features provided by PCAPng.

EditCap has the ability to add comments to a packet, but there is no way to add comments to a range of packets. The “-a” option depicted in Figure 15 only works one frame at a time. EditCap also requires an input file which means that comments cannot be added during a capture. The PCAPng file must be fully written before it can be adjusted by EditCap. Clearly, the tools provided in Wireshark provide an analyst the most comprehensive ability to take advantage of additional options and fields that the PCAPng file format introduces. As a developing format, Wireshark could be improved to support some more advanced operations, however. The ability to add per-packet comments at the time of capture would be a desirable improvement, for example. This capability applied to specific packets within a capture could help Wireshark take full advantage of commenting capabilities. In the meantime, analysts resort to scripting to take advantage of PCAPng's new features.

3.2 Scripting It Out

Although PCAPng has some clear advantages over its predecessor, the challenge for analysts becomes how to use these features while their favorite tools

© 2018 The SANS Institute Author retains full rights.

© 2018 The SANS Institute, Author Retains Full Rights

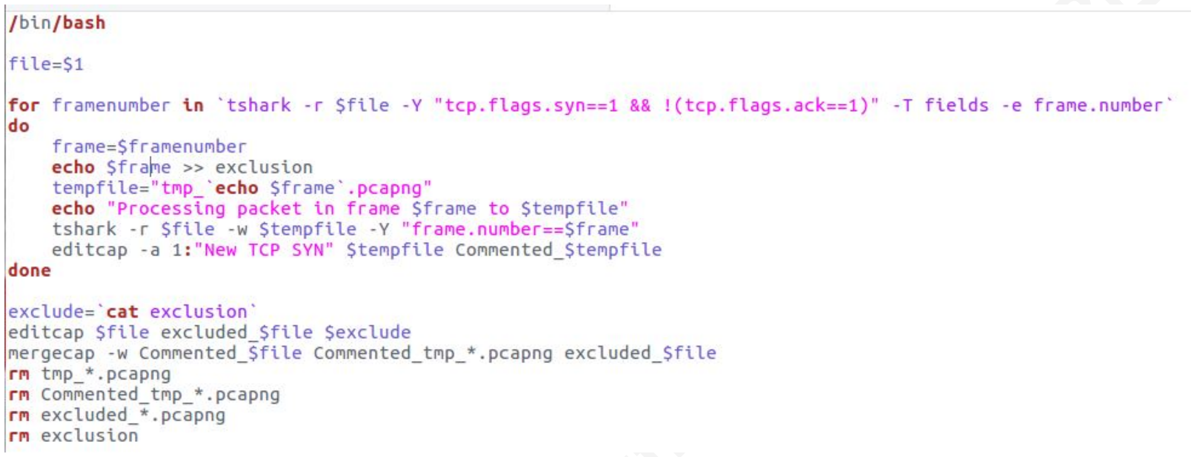
are slow to advance their capabilities. This section will pivot over to a Linux system to use the features of bash scripting. Since the ability to write PCAPng files is not fully supported in some languages, scripting combined with Wireshark's tools can be used to take advantage of PCAPng. This example will show how a script can be used to filter interesting traffic from any capture, and how comments can be added to each of those packets. As stated previously, tools like EditCap rely on a fully written PCAPng file, so this is a post-capture task. It can save time and help the analyst comment on interesting traffic through automation.

During script development, a Linux distribution that had Wireshark installed along with TShark, EditCap, and MergeCap was used. The script will also work on a Windows system with Bash installed. Bash provided the easiest way to manipulate the data and pass it between Wireshark's different tools. The SIFT Workstation from SANS was downloaded for this. Since it has already been demonstrated how to capture traffic from TShark, this demonstration uses a previously captured PCAPng file that had a large amount of web traffic. Chris Sanders has a multitude of PCAPng files posted on his GitHub. I chose one called lotsofweb.pcapng (Sanders, 2017).

When analyzing new captures, I often spend time looking for new TCP connections. This is especially interesting when analyzing web traffic. One could also be looking for connections to IP addresses that might be untrusted or adversarial. For captures that have a lot of TCP connections, it can be helpful to add a comment to each new connection. Because of this, I decided to write the script so it filters on new TCP connections and creates a new PCAPng file that includes comments on the first SYN packet for new connections. In Wireshark, the filter for this type of traffic could be tcp.flags.syn==1 && !(tcp.flags.ack==1). This will filter out only the initial SYN packet from a TCP three-way handshake. The script is posted in Figure 16.

© 2018 The SANS Institute Author retains full rights.

© 2018 The SANS Institute, Author Retains Full Rights



```
/bin/bash

file=$1

for framenumber in `tshark -r $file -Y "tcp.flags.syn==1 && !(tcp.flags.ack==1)" -T fields -e frame.number`
do
    frame=$framenumber
    echo $frame >> exclusion
    tempfile="tmp_`echo $frame`.pcapng"
    echo "Processing packet in frame $frame to $tempfile"
    tshark -r $file -w $tempfile -Y "frame.number==$frame"
    editcap -a 1:"New TCP SYN" $tempfile Commented_$tempfile
done

exclude=`cat exclusion`
editcap $file excluded_$file $exclude
mergecap -w Commented_$file Commented_tmp_*.pcapng excluded_$file
rm tmp_*.pcapng
rm Commented_tmp_*.pcapng
rm excluded_*.pcapng
rm exclusion
```

Figure 16 – PCAPng Comment Script

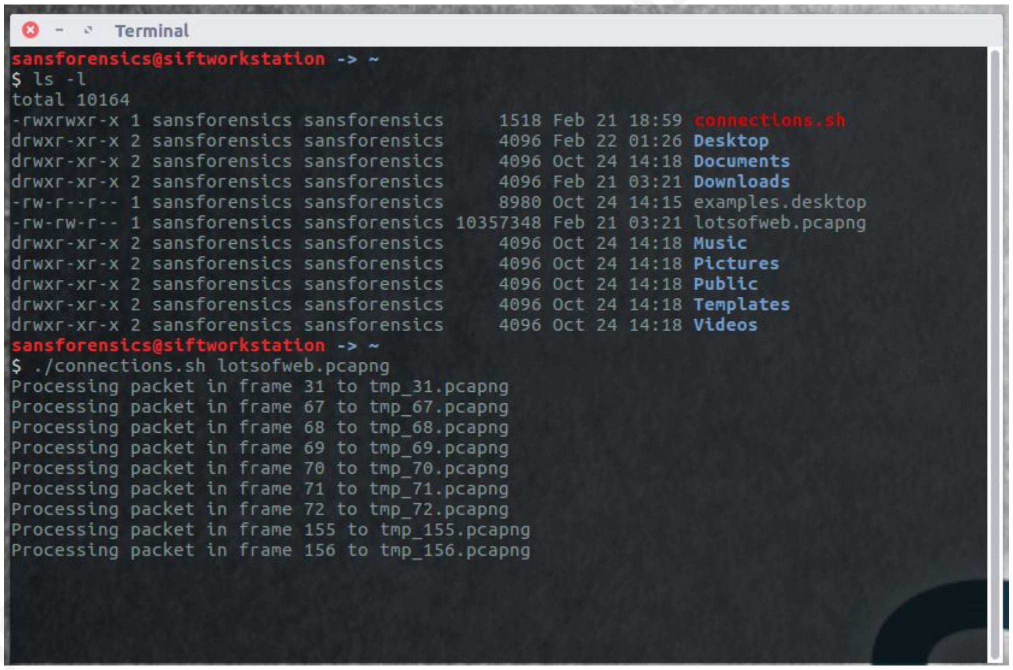
The FOR loop utilizes TShark to filter on our TCP SYN connections and extracts the frame number for each of those frames. The frame number is important because it uniquely identifies each frame. Within the loop, the frame number is appended to a file called “exclusion” for later use. Each frame that was identified in the filter is temporarily written to its own individual file. After it has been written to an individual file, EditCap is used to add the comment "New TCP SYN" to the frame. At the end of the FOR loop, each frame exists in its own temporary file which has been commented on. The reason I decided to break each frame out to individual files is that EditCap requires an input and output file and can only add comments one frame at a time. Using it in a FOR loop seemed the most efficient way to do it with those limitations.

Once the FOR loop has commented on the individual frames, the script has to merge the data back together. While writing the script, I discovered that simply using MergeCap to add the individual files to the original capture results in duplicate frames. This cannot be resolved by using the dedupe option in EditCap because the additional comment changes the md5 hash of the duplicated packet. For this reason, the script keeps track of which frames were edited in the "exclusion" file. This makes it simple to remove the edited frames from the original capture and rename it “excluded_$file.” The script then merges all the commented temporary files and the excluded_$file to produce the final product. The end result of the script is a new

© 2018 The SANS Institute Author retains full rights.

PCAPng file named "Commented_$file." The only difference in the new file is that all initial SYN packets are commented on. All temporary files are removed upon completion of the script. The only files that remain are the original capture and the newly commented capture files.

In order to run the script, the capture file is placed in the same directory as the script. Simply run the command ./connections.sh *inputfile.pcapng.* Figure 17 shows the process. The script will display feedback to the terminal as each frame is processed.

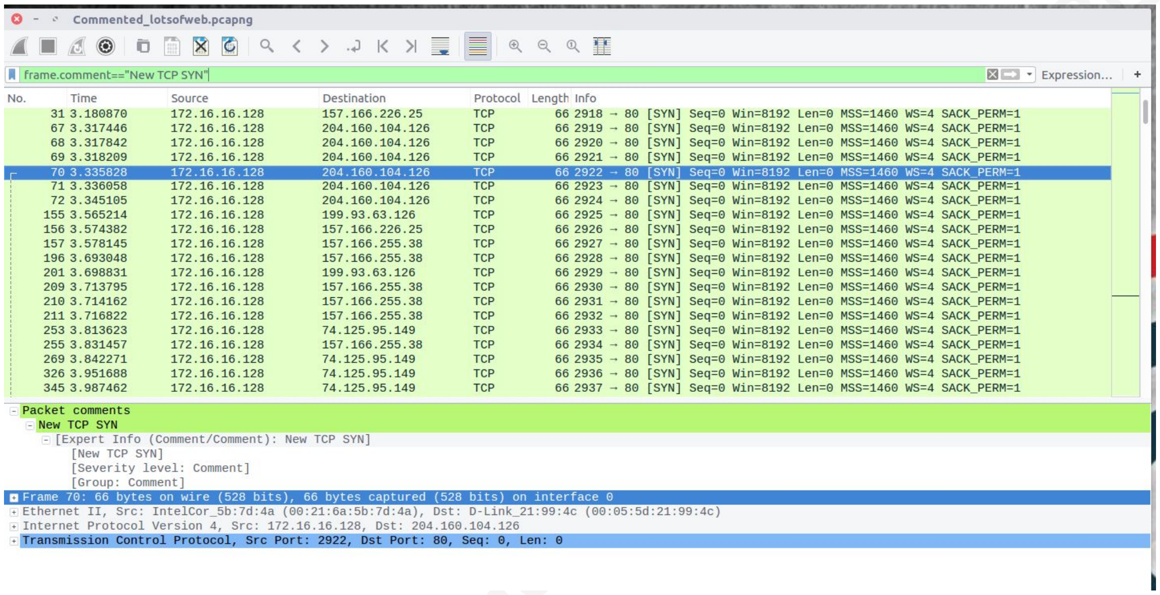


```
sansforensics@siftworkstation -> ~
$ ls -l
total 10164
-rwxrwxr-x 1 sansforensics sansforensics     1518 Feb 21 18:59 connections.sh
drwxr-xr-x 2 sansforensics sansforensics     4096 Feb 22 01:26 Desktop
drwxr-xr-x 2 sansforensics sansforensics     4096 Oct 24 14:18 Documents
drwxr-xr-x 2 sansforensics sansforensics     4096 Feb 21 03:21 Downloads
-rw-r--r-- 1 sansforensics sansforensics     8980 Oct 24 14:15 examples.desktop
-rw-rw-r-- 1 sansforensics sansforensics 10357348 Feb 21 03:21 lotsofweb.pcapng
drwxr-xr-x 2 sansforensics sansforensics     4096 Oct 24 14:18 Music
drwxr-xr-x 2 sansforensics sansforensics     4096 Oct 24 14:18 Pictures
drwxr-xr-x 2 sansforensics sansforensics     4096 Oct 24 14:18 Public
drwxr-xr-x 2 sansforensics sansforensics     4096 Oct 24 14:18 Templates
drwxr-xr-x 2 sansforensics sansforensics     4096 Oct 24 14:18 Videos
sansforensics@siftworkstation -> ~
$ ./connections.sh lotsofweb.pcapng
Processing packet in frame 31 to tmp_31.pcapng
Processing packet in frame 67 to tmp_67.pcapng
Processing packet in frame 68 to tmp_68.pcapng
Processing packet in frame 69 to tmp_69.pcapng
Processing packet in frame 70 to tmp_70.pcapng
Processing packet in frame 71 to tmp_71.pcapng
Processing packet in frame 72 to tmp_72.pcapng
Processing packet in frame 155 to tmp_155.pcapng
Processing packet in frame 156 to tmp_156.pcapng
```

Figure 17 – Running the Script

When the script is complete, a new file called Commented_$file.pcapng will be placed in the current directory. In this case, the file is called "Commented_lotsofweb.pcapng." Using Wireshark, opening the new file will show that all TCP SYN frames are now commented with "New TCP SYN". We can filter on these using the Wireshark filter frame.comment=="New TCP SYN". We can now filter on the comment and view the initial connection to all TCP streams. This is just one way to take advantage of the PCAPng commenting feature. The filter is shown in Figure 18.

© 2018 The SANS Institute Author retains full rights.



```
Commented_lotsofweb.pcapng
frame.comment=="New TCP SYN"

No.  Time      Source         Destination     Protocol Length Info
 31 3.180870   172.16.16.128  157.166.226.25  TCP      66 2918 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
 67 3.317446   172.16.16.128  204.160.104.126 TCP      66 2919 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
 68 3.317842   172.16.16.128  204.160.104.126 TCP      66 2920 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
 69 3.318209   172.16.16.128  204.160.104.126 TCP      66 2921 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
 70 3.335828   172.16.16.128  204.160.104.126 TCP      66 2922 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
 71 3.336058   172.16.16.128  204.160.104.126 TCP      66 2923 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
 72 3.345105   172.16.16.128  204.160.104.126 TCP      66 2924 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
155 3.565214   172.16.16.128  199.93.63.126   TCP      66 2925 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
156 3.574382   172.16.16.128  157.166.226.25  TCP      66 2926 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
157 3.578145   172.16.16.128  157.166.255.38  TCP      66 2927 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
196 3.693048   172.16.16.128  157.166.255.38  TCP      66 2928 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
201 3.698831   172.16.16.128  199.93.63.126   TCP      66 2929 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
209 3.713795   172.16.16.128  157.166.255.38  TCP      66 2930 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
210 3.714162   172.16.16.128  157.166.255.38  TCP      66 2931 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
211 3.716822   172.16.16.128  157.166.255.38  TCP      66 2932 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
253 3.813623   172.16.16.128  74.125.95.149   TCP      66 2933 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
255 3.831457   172.16.16.128  157.166.255.38  TCP      66 2934 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
269 3.842271   172.16.16.128  74.125.95.149   TCP      66 2935 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
326 3.951688   172.16.16.128  74.125.95.149   TCP      66 2936 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
345 3.987462   172.16.16.128  74.125.95.149   TCP      66 2937 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1

Packet comments
  New TCP SYN
    [Expert Info (Comment/Comment): New TCP SYN]
      [New TCP SYN]
      [Severity level: Comment]
      [Group: Comment]
Frame 70: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
Ethernet II, Src: IntelCor_5b:7d:4a (00:21:6a:5b:7d:4a), Dst: D-Link_21:99:4c (00:05:5d:21:99:4c)
Internet Protocol Version 4, Src: 172.16.16.128, Dst: 204.160.104.126
Transmission Control Protocol, Src Port: 2922, Dst Port: 80, Seq: 0, Len: 0
```

Figure 18 – Commented Frames

The script can automate comments for various types of filters that need to be applied. The only changes necessary would be to change the filter used on the original TShark command in the FOR loop – then change the comment in the EditCap line to whatever the analyst desires. The script can be a great way to take advantage of commenting features that the PCAPng file format is capable of. Until other tools incorporate some of these features, scripting is a great way to take advantage of PCAPng. I encourage others to edit the script for their own use. The full script is included in Appendix A.

4. Conclusion

There is no doubt that PCAPng is an improvement over the old libpcap file format. Its additional capabilities such as multiple interface capture, per-packet comments, and improved time resolution make the transition a worthy one. So far, packet capture applications have failed to fully implement the capabilities of the new format. Even Wireshark, which is responsible for much of PCAPng's advancement to date, shows limitations in its ability to take advantage of the new fields. The necessity to pass a PCAPng file from tool to tool is unfortunate, and analysts could increase efficiency if tools were more supportive of the format.

© 2018 The SANS Institute Author retains full rights.

Further research on this topic would seek to integrate the PCAPng file format into more tools so its new fields can be harnessed for advanced research and increased cyber forensic capability. As networks grow faster and more complex, cyber defenders' capability to analyze threats must be accurate and provide as much metadata as possible. This research intends to show application developers the benefits of the new file format. Additionally, it is my hope that it gives them the encouragement to integrate it into their code so these benefits are supported across a multitude of tools. Packet analysis will continue to play an important role in defending networks and analyzing malware, especially with the increased use of file-less malware. It is important that projects such as PCAPng are supported so that they can continue to provide adaptable solutions to problems that defenders will face in the future.

© 2018 The SANS Institute, Author Retains Full Rights

© 2018 The SANS Institute Author retains full rights.

References

Development/LibpcapFileFormat. (n.d.). Retrieved November 14, 2017, from https://wiki.wireshark.org/Development/LibpcapFileFormat#Libraries

Koch, M. (2016). *Implementing Full Packet Capture*. Retrieved October 5th, 2017, from https://www.sans.org/reading-room/whitepapers/forensics/implementing-full-packet-capture-37392.

NIST. (2016). *NIST Special Publication 800-53 Rev4*. Retrieved from NIST.gov: http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf

P. (n.d.). Pcapng/pcapng. Retrieved November 13, 2017, from https://github.com/pcapng/pcapng/wiki/Implementations

Packet Foo. (2014). The trouble with multiple capture interfaces. Retrieved from https://blog.packet-foo.com/2014/08/the-trouble-with-multiple-capture-interfaces/

Sanders, C. (2017, June 19). Chrissanders/packets. Retrieved February 18, 2018, from https://github.com/chrissanders/packets

Sanders, C. (2017). *Practical packet analysis: using Wireshark to solve real-world network problems.* San Francisco, CA: No Starch Press

Sanders, C., & Smith, J. (2014). Applied Network Security Monitoring.

TCPDUMP & LIBPCAP. (n.d.). Retrieved November 14, 2017, from http://www.tcpdump.org/

Tuexen, E., Risso, F., Bongertz, J., Combs, G., Harris, G., (2017). *PCAP Next Generation (pcapng) Capture File Format*. Retrieved October 10, 2017, from http://xml2rfc.tools.ietf.org/cgibin/xml2rfc.cgi?url=https://raw.githubusercontent.com/pcapng/pcapng/master/draft-tuexenopsawg-pcapng.xml&modeAsFormat=html/ascii&type=ascii#rfc.section.9

Walls, J. (2012, October 02). Five Reasons to Move to the Pcapng Capture Format (by Jason Walls). Retrieved December 12, 2017, from http://www.lovemytool.com/blog/2012/10/five-reasons-to-move-to-the-pcapng-capture-format-by-jason-walls.html

Author

© 2018 The SANS Institute Author retains full rights.

© 2018 The SANS Institute, Author Retains Full Rights

Appendix A

Commenting Script

```
# Author: Scott Fether
# February 21, 2018
# This script will identify new TCP Connections and add per-packet comments to
# the initial SYN frame. The script filters specifically on new TCP connections, but
# it can be modified to filter on anything TShark accepts. Comments can be changed
# to describe the interesting traffic.

#!/bin/bash
file=$1
for framenumber in `tshark -r $file -Y "tcp.flags.syn==1 && !(tcp.flags.ack==1)" -T
fields -e frame.number`
do
	frame=$framenumber
	echo $frame >> exclusion
	tempfile="tmp_`echo $frame`.pcapng"
	echo "Processing packet in frame $frame to $tempfile”
	tshark -r $file -w $tempfile -Y "frame.number==$frame"
	editcap -a 1:"New TCP SYN" $tempfile Commented_$tempfile
done

exclude=`cat exclusion`
editcap $file excluded_$file $exclude
mergecap -w Commented_$file Commented_tmp_*.pcapng excluded_$file
rm tmp_*.pcapng
rm Commented_tmp_*.pcapng
rm excluded_*.pcapng
rm exclusion
```

© 2018 The SANS Institute Author retains full rights.

© 2018 The SANS Institute, Author Retains Full Rights