Global Information Assurance Certification Paper

A Beginners Guide to tcpdump

GSEC Practical v.1.4b
Andy Wagoner

Summary:

The primary goal for this paper is to explain packet sniffing using tcpdump. A short general explanation of how data is transmitted over a network will be covered as well as some important reasons a person may want to sniff network traffic. We will cover what packet sniffing is at an introductory level as well as where to find further resources for additional study. Two installation scenarios will be demonstrated in a systematic fashion. Different methods of sniffing traffic will be shown as well as how to analyze the data packets that have been captured. Practical examples with real data are used to demonstrate types of traffic that can be captured and analyzed on a network. This is will solidify a basic understanding of packet sniffing. A light introduction to other applications for sniffing traffic will be provided as well. Other applications will be covered at a higher level with enough detail to entice the reader to investigate and experiment.

Introduction:

Network traffic travels in data packets; each data packet contains the information that it needs to travel across the network. This information is contained in a TCP header. A TCP header will contain the destination and source address, state information, and protocol identifiers. The rest of the packet contains the data that is being sent. Devices that are responsible for routing read the information in these packets and send them to their correct destination. Sniffing is a process that passively monitors and captures these packets. Tcpdump is a packet-sniffing tool that is used by network administrators to sniff and analyze traffic on a network. A couple of reasons for sniffing traffic on a network would be to verify connectivity between hosts, or to analyze the traffic that is traversing the network. There are several tools available such as Ethereal, Snort, Etherape, tcpdump and more. You can find a more complete list of tools by visiting this link: http://www.cromwell-intl.com/security/monitoring.html

Packet Sniffing:

What is packet sniffing? “Packet sniffing is a form of wire-tap applied to computer networks instead of phone networks.” (http://www.iss.net/security_center/advice/Underground/Hacking/Methods/Technical/Packet_sniffing/default.htm)

In order to successfully sniff packets you need to be on a LAN that is connected to a hub or a switch that has port mirroring enabled. When using packet sniffing tools, keep in mind that there are distinct differences between a hub and switch, which can make your packet sniffing experience a difficult one if you don’t understand them. First, the hub does not know how to route the traffic. As a result, when one computer sends a packet to another computer connected to the hub, the packet is sent from the first computer through the hub and

broadcasted to all of the computers connected to it. Second, a switch knows to accept traffic on a specific port(s). The network traffic is not broadcast to all computers that are connected to it. If you want to sniff traffic from a switch, you must first enable port mirroring. This configures the switch to “duplicate” the traffic and send it to the specified port enabling a sniffer to gather information. For this paper, we will be connected to an 8-port hub.

Tcpdump HowTo:

Most Linux distributions install a version of tcpdump as part of a standard operating system install. Of course, this depends on the options you choose during the installation. If a custom install is chosen, then it is possible that this package will not be available until you install it manually. You can retrieve the package from the link listed below, or you can install it from the Linux CD’s. For this paper, we will be using RedHat Linux 9 workstation installation with tcpdump-3.7.2-1.9.1.

Installing tcpdump from the RPM:

To see if you have tcpdump installed on your system, type the following command from a Linux shell: rpm –q tcpdump
This should show you some output similar to the following (it may look slightly different depending on the version you have installed):

[root@iroland root]# rpm -q tcpdump
tcpdump-3.7.2-1.9.1

(Note: rpm represents RedHat Package Management, the –q option represents query. The –i option represents install, the –v is for verbose and the –h is to display status in the form of a hash mark. You can find more information regarding the use of rpm by reading the rpm manpage.)

If you do not have tcpdump installed you should see something like this:

[root@iroland root]# rpm -q tcpdump
package tcpdump is not installed

If the package is not installed, you can get the RPM from the RedHat CD. This is probably the easiest method of installation, however, installation from the source will be covered as well. First, verify that the libpcap rpm is installed. If it is not, then install libpcap-0.7.2-1.i386.rpm from disk 2 of the RedHat CD’s. Insert the CD, type cd /mnt/cdrom, next type cd RedHat/RPMS, then type:

rpm –ivh libpcap-0.7.2-1.i386.rpm

then…

rpm –ivh tcpdump-3.7.2-1.9.1.rpm

This will install the packages and you will then be ready to use tcpdump.

Installing tcpdump from the source files:

If you do not have access to the operating system CD's; an alternative way to install tcpdump is to point a web browser to http://www.tcpdump.org and find the most current version. Again, for this paper, the most recent version is tcpdump-3.7.2.tar.gz. It is important to note that libpcap be installed prior to the installation of tcpdump. This is a library file, "which provides a packet filtering mechanism based on the BSD packet filter (BPF)." (http://freshmeat.net/projects/libpcap/) tcpdump will not function without it. libpcap can also be found on http://www.tcpdump.org.

At the time of this paper, the most current stable version available is libpcap 0.7.2. Download the appropriate files and save them to a temporary directory.
Change to the temporary directory and type:

tar –zxvf libpcap-0.7.2.tar.gz

after this extracts completely type

tar –zxvf tcpdump-3.7.2.tar.gz

(Note: tar is an archiving program designed to store and extract files from an archive file known as a tarfile. This information was taken from the tar manpage, RedHat Linux 9. See the manpage for a further description of this program and its various options.)

This will unzip the package and unpack it in one smooth operation. After you have completed this step, you will see the tcpdump-3.7.2 and the libpcap-0.7.2 directories. First, change to the libpcap-0.7.2 directory. As each process finishes, type:

sh ./configure
sh ./make
sh ./make install (Note: You must be root or have root privileges to run ./make install)

Repeat this process from within the tcpdump-3.7.2 directory. This will install libpcap and tcpdump. You should be ready to use the program at this point.

Using tcpdump:

To start tcpdump type: **tcpdump** from the command prompt. By not specifying a device, port or type of traffic, tcpdump will pick the lowest numbered device, all ports and all types of traffic. In most cases, the lowest numbered device is going to be "eth0". Please check the documentation of your system for the proper description of the Ethernet interface.

Following will be a discussion concerning some of the different ways to use tcpdump. Please view the available options as listed from the manpage on Redhat Linux 9:

tcpdump [-adeflnNOpqRStuvxX] [-c count]
[-C file_size] [-F file]
[-i interface] [-m module] [-r file]
[-s snaplen] [-T type] [-U user] [-w file]
[-E algo:secret] [expression]

As the goal of this paper is to give a basic understanding of tcpdump, covering all of the available filtering capabilities is outside the scope of this discussion. Therefore, it is left to the readers curiosity to explore some of the other options.

The first command and output we will cover is: **tcpdump –i eth0**. This option tells tcpdump to look at the interface (-i) eth0 for all traffic that is traveling through the specified interface. This is an extremely verbose method of getting all types of traffic and can be difficult to process the number of packets that are being transmitted. Since all traffic seen on this interface is displayed, it can be tough to isolate what you are looking for exactly. Here is an example of the output:

```
17:35:50.098668 192.168.0.102.50022 > 192.168.0.101.1095: P 73168:73328(160) ack 801 win 20976 (DF) [tos
0x10]
17:35:50.099008 192.168.0.102.50022 > 192.168.0.101.1095: P 73328:73488(160) ack 801 win 20976 (DF) [tos
0x10]
17:35:50.099112 192.168.0.101.1095 > 192.168.0.102.50022: . ack 73328 win 17520 (DF)
17:35:50.099385 192.168.0.102.50022 > 192.168.0.101.1095: P 73488:73616(128) ack 801 win 20976 (DF) [tos
0x10]
17:35:50.099724 192.168.0.102.50022 > 192.168.0.101.1095: P 73616:73776(160) ack 801 win 20976 (DF) [tos
0x10]
17:35:50.099791 192.168.0.101.1095 > 192.168.0.102.50022: . ack 73616 win 17232 (DF)
17:35:50.100112 192.168.0.102.50022 > 192.168.0.101.1095: P 73776:73936(160) ack 801 win 20976 (DF) [tos
0x10]
17:35:50.100431 192.168.0.102.50022 > 192.168.0.101.1095: P 73936:74064(128) ack 801 win 20976 (DF) [tos
0x10]
17:35:50.100712 192.168.0.101.1095 > 192.168.0.102.50022: . ack 73936 win 16912 (DF)
17:35:50.100825 192.168.0.102.50022 > 192.168.0.101.1095: P 74064:74224(160) ack 801 win 20976 (DF) [tos
0x10]
17:35:50.101164 192.168.0.102.50022 > 192.168.0.101.1095: P 74224:74384(160) ack 801 win 20976 (DF) [tos
0x10]

593 packets received by filter
5 packets dropped by kernel
```

After typing **tcpdump –i eth0** and sniffing traffic for approximately 2 seconds, there were 593 packets sniffed as indicated in the highlighted portion of the frame. This is just to give you an idea of how much information might be displayed by not specifying a specific port or a specific type of data to sniff.

Lets break down some of what is shown. This first field shows the time that the packets were traveling, in this format hh:mm:ss:fraction

17:35:50.098668

The second field shows the source host address and port, followed by the destination host address and port:

192.168.0.102.50022 > 192.168.0.101.1095

The third field shows the TCP flag P to indicate that it is pushing data. Immediately after you see the sequence numbers (73168:73328) and the amount of data to be sent indicated in bytes.

: P 73168:73328(160)

Next, you see the ack TCP flag, followed by the window size (win 20976) that is being advertised.

ack 801 win 20976

Finally, you see (DF) which means don't fragment the packet and the (tos) or the TCP type of service.

(DF) [tos 0x10]

Keep in mind that this is just a sample packet taken at random; it does not show the three-way handshake, which is: SYN, ACK, SYN ACK. These are known as TCP flags. A few examples are S = SYN synchronizes the sequence numbers, F = FIN termination of connection by sender, P = PUSH push data, and R = RST reset the connection. When there are no TCP flags a "." is displayed, more or less as a placeholder.

The Three-way Handshake:

The three-way handshake is simply the source host and the destination host requesting a connection, and then confirming to each other that a connection has been made. As mentioned above, to open a session a client determines a local source port and an Initial Sequence Number (ISN). The ISN is a randomly determined integer between 0 and 4,294,967,295. Communicating hosts exchange ISNs during connection initialization. Each host sets two counters: sequence and acknowledgement. In the context of a single TCP packet, the sequence number is set by the sending host, and the acknowledgement number is set by the receiving host.

(Taken from http://ruadh.hypermart.net/Classes/NetEss/ThreeWay.htm) The general format in which tcpdump displays TCP is: src > dst: flags data-sequence number ack window urgent options (Taken from the tcpdump man page) It is important to understand sequence numbers when working with tcpdump. The best explanation regarding sequence numbers comes from the GSEC Security Essentials information. Chapter 3, Section 1 – Networking

Concepts. It reads as follows:

> ”TCP numbers every byte of data it sends with a unique *sequence number*. This allows either side of the connection to refer to specific bytes by number (i.e., “the 103rd byte you sent me”). A connection’s *Initial Sequence Number (ISN)* is the first sequence number used in that connection. TCP initializes the ISN to a random or semi-random value (the more random, the better, for security reasons). Sequence numbers for the rest of the bytes in the connection are then derived from the ISN by incrementing it by 1 for each byte sent. The sequence number of the first byte sent always equals the ISN + 1. Therefore, if the ISN was 3003873, then the 103rd byte would be sequence number (3003873 + 103) or 3003976.”

A brief description of TCP flags was taken from the GSEC Security Essentials information. Chapter 3, Section 1 – Networking Concepts:

> “TCP stacks sometimes need to communicate about the data they’re exchanging. The catch is they can’t insert their own information into the payload because that would corrupt the data stream and might confuse the applications. Instead, the TCP protocol provides six one-bit flags that can be specified in the packet headers. Some of these are more common than others, but the unusual use of TCP flags is a good indicator of suspicious traffic, so you should become familiar with all of them.”

A snap shot of the three-way handshake follows. For this example, we are connecting to a Samba server. For this we are going to use the command:

tcpdump –i eth0 port 139

Here is the output:

```
[root@iroland root]# tcpdump -i eth0 port 139
tcpdump: listening on eth0
17:38:56.273633 192.168.0.100.2904 > 192.168.0.102.netbios-ssn: S 2363130086:2363130086(0) win
64240 <mss 1460,nop,nop,sackOK> (DF)

13 packets received by filter
0 packets dropped by kernel
```

Right away, we see a packet similar to the one we decoded earlier, highlighted in yellow.

This packet has the standard time format: hh:mm:ss:frac, and the source host and port. If you notice the information behind the destination host address, you see the protocol netbios-ssn. We are looking for traffic on port 139. The output shows what type of traffic is seen on that port. Further into the packet you see the “S” signaling that this is the first part in the three way hand shake SYN, then the sequence bytes. Next the output shows the window size (64240) with more information about the maximum segment size or <mss 1460,nop,nop,sackOK> and finally (DF) don’t fragment.

The reply of the destination host looks something like this:

```
17:38:56.273711 192.168.0.102.netbios-ssn > 192.168.0.100.2904: S 4279673549:4279673549(0)
ack 2363130087 win 5840 <mss 1460,nop,nop,sackOK> (DF)
```

Most of the information looks the same, with the exception of the sequence numbers and the “ack” TCP flag. The destination host address is sending its own set of sequence bytes. As explained in the previous section, this is the part where the two systems synchronize their sequence bytes. Hence, the “S” TCP flag on both packets.
The final transmission of the three-way handshake:

```
17:38:56.273885 192.168.0.100.2904 > 192.168.0.102.netbios-ssn: . ack 1 win 64240 (DF)
```

This packet looks a little different from the rest of the ones we have been looking at. It has the timestamp, the source host and port, the destination host and type of session, and the “ack” TCP flag is set. This signifies that the connection between the two hosts is complete therefore ending the three-way handshake.

An example of an FTP connection:

In the next example, we will look at the process of connecting to an FTP server, and viewing the activity of the connection in hex and ASCII text. Type the following command:

tcpdump -n -nn -X port 21

The output follows:

```
Tcpdump: listening on eth0
21:33:38.681840 192.168.0.102.21 > 192.168.0.105.32851: P 267:309(42) ack 1 win 5792 <nop,nop,timestamp
99907248 99898767> (DF)
0x0000   4500 005e a8d3 4000 4006 0fa7 c0a8 0066        E..^..@.@......f
0x0010   c0a8 0069 0015 8053 ce62 df87 a227 2efa        ...i...S.b...'..
0x0020   8018 16a0 0233 0000 0101 080a 05f4 76b0        .....3........v.
0x0030   05f4 558f 3232 3020 5072 6f46 5450 4420        ..U.220.ProFTPD.
0x0040   312e 322e 3820 5365 7276 6572 2028 4654        1.2.8.Server.(FT
0x0050   5029                                           P)
```

The example above shows the initial connection, notice that you can see the version and name of the FTP server application driving the FTP session, highlighted in yellow. The “-n” option means, do not convert the host addresses to names. This is a way of cleaning up the output for enhanced readability. The “-nn” option means, do not convert the protocol and port numbers to names either. Chances are if you are doing these types of scans, you are already going to know this information and do not need to see it displayed in the output. The “-X” tells tcpdump to print both hex and ASCII for the data that is being pushed across the wire. Now look below at the areas highlighted in yellow.

The first section shows the server accepting the username “pals”, just after that you see that it is requiring a password for the user. Further, below that you see the actual password for this session “m4xr19ty” displayed clearly. Finally, you see that the user pals has successfully logged in.

```
21:34:29.243935 192.168.0.105.32851 > 192.168.0.102.21: P 1:12(11) ack 309 win 6432
<nop,nop,timestamp 99903824 99907248> (DF) [tos 0x10]
0x0000   4510 003f 7b7a 4000 4006 3d0f c0a8 0069        E..?{z@.@.=....i
0x0010   c0a8 0066 8053 0015 a227 2efa ce62 dfb1        ...f.S...'...b..
0x0020   8018 1920 5653 0000 0101 080a 05f4 6950        ....VS.........iP
0x0030   05f4 76b0 5553 4552 2070 616c 730d 0a          ..v.USER.pals..
21:34:29.244194 192.168.0.102.21 > 192.168.0.105.32851: . ack 12 win 5792 <nop,nop,timestamp
99912305 99903824> (DF)
0x0000   4500 0034 a8d4 4000 4006 0fd0 c0a8 0066        E..4..@.@......f
0x0010   c0a8 0069 0015 8053 ce62 dfb1 a227 2f05        ...i...S.b...'/.
0x0020   8010 16a0 dea9 0000 0101 080a 05f4 8a71        ...............q
0x0030   05f4 6950                                      ..iP
21:34:29.246535 192.168.0.102.21 > 192.168.0.105.32851: P 309:342(33) ack 12 win 5792
<nop,nop,timestamp 99912305 99903824> (DF)
0x0000   4500 0055 a8d5 4000 4006 0fae c0a8 0066        E..U..@.@......f
0x0010   c0a8 0069 0015 8053 ce62 dfb1 a227 2f05        ...i...S.b...'/.
0x0020   8018 16a0 0d40 0000 0101 080a 05f4 8a71        .....@.........q
0x0030   05f4 6950 3333 3120 5061 7373 776f 7264        ..iP331.Password
0x0040   2072 6571 7569 7265 6420 666f 7220 7061        .required.for.pa
0x0050   6c73                                          ls
21:34:29.247049 192.168.0.105.32851 > 192.168.0.102.21: . ack 342 win 6432 <nop,nop,timestamp
99903825 99912305> (DF) [tos 0x10]
0x0000   4510 0034 7b7b 4000 4006 3d19 c0a8 0069        E..4{{@.@.=....i
0x0010   c0a8 0066 8053 0015 a227 2f05 ce62 dfd2        ...f.S...'/..b..
0x0020   8010 1920 dc07 0000 0101 080a 05f4 6951        .............iQ
0x0030   05f4 8a71                                      ...q
21:35:40.044915 192.168.0.105.32851 > 192.168.0.102.21: P 12:27(15) ack 342 win 6432
<nop,nop,timestamp 99910905 99912305> (DF) [tos 0x10]
0x0000   4510 0043 7b7c 4000 4006 3d09 c0a8 0069        E..C{|@.@.=....i
0x0010   c0a8 0066 8053 0015 a227 2f05 ce62 dfd2        ...f.S...'/..b..
0x0020   8018 1920 991b 0000 0101 080a 05f4 84f9        ................
0x0030   05f4 8a71 5041 5353 206d 3478 7231 3974        ...qPASS.m4xr19ty
0x0040   790d 0a                                        y..
21:35:40.056722 192.168.0.102.21 > 192.168.0.105.32851: P 342:368(26) ack 27 win 5792
<nop,nop,timestamp 99919388 99910905> (DF)
0x0000   4500 004e a8d6 4000 4006 0fb4 c0a8 0066        E..N..@.@......f
0x0010   c0a8 0069 0015 8053 ce62 dfd2 a227 2f14        ...i...S.b...'/.
0x0020   8018 16a0 bff0 0000 0101 080a 05f4 a61c        ................
0x0030   05f4 84f9 3233 3020 5573 6572 2070 616c        ....230.User.pal
0x0040   7320 6c6f 6767 6564 2069 6e2e 0d0a            s.logged.in...
```

By viewing this information, you can clearly determine who is doing what on your network. If we were to continue monitoring this session we would see the different directories that the user goes to, the files downloaded etc…

Using the –w option allows you to write the session to a file. You can run it through other applications that can read pcap dump files such as Ethereal, or Snort, for more detailed analysis of the data. This option will store the packets in raw format to the file you specify.

tcpdump –i eth0 –w tdump1

Using this option is going to be faster since you will not be printing the output to the screen. Instead, the data is saved to the file in binary format.

The example shown will capture all traffic on eth0. As we saw in the first example of this paper, this will produce a lot of data and disk space becomes an issue at some point.
(Note: This option is shown for the purpose of example. There are times when you may want to do this, possibly to analyze the data further with other options provided by tcpdump or other applications like Ethereal or Snort.)

It is possible to collect a specific amount of data by using the –c option. This tells tcpdump to count the number of packets and stop when it reaches the number that you specify.

tcpdump –i eth0 –c 1000 –w file1

This will capture 1000 packets on interface eth0 and put it into a file called file1. Since the "-w" option only writes to the file in binary format, it is not very readable for humans. To convert the data to human readable format, you will need to use the following filter:

tcpdump –i eth0 –r file1>file2

This reads the data from file1, converts it to human readable format and saves it into file2. After using this filter, you are now able to see the packet headers. If you want to take this a step further and convert the files into hex and ASCII for more detailed analysis do this:

tcpdump –r file1 –X -nn

This command reads from file1 and converts the data to a hex and ASCII display. With the –nn option the protocols and port numbers are not converted to names. The output here is going to look like the example above where we saw the username and password being passed as traffic across the wire.

```
0x0000   4510 003f 7b7a 4000 4006 3d0f c0a8 0069      E..?{z@.@.=....i
0x0010   c0a8 0066 8053 0015 a227 2efa ce62 dfb1      ...f.S...'...b..
0x0020   8018 1920 5653 0000 0101 080a 05f4 6950      ....VS.........iP
0x0030   05f4 76b0 5553 4552 2070 616c 730d 0a        ..v.USER.pals
```

Other Tools:

Several other tools are available that will complete the same tasks that tcpdump is capable of and more. Some provide a GUI interface allowing for a more appealing, formatted view of the data you are sniffing.

Etherape:

"Etherape is a graphical network monitor for Unix modeled after etherman. Featuring link layer, ip and TCP modes, it displays network activity graphically. Hosts and links change in size with traffic. Color-coded protocols display. It supports Ethernet, FDDI, Token Ring, ISDN, PPP and SLIP devices. It can filter traffic to be shown, and can read traffic from a file as well as live from the network." (http://www.cromwell-intl.com/security/monitoring.html) This is a free download and functions very much like Ethereal. You can find a higher level of detail about the uses and additional features of this application by visiting the previous link.

Ethereal:

Ethereal is also a free download and can be used on Windows, or Unix platforms. Be sure to check the prerequisites required to use this program with your operating system. As described: "Ethereal allows you to examine data from a live network or from a capture file on disk. You can interactively browse the capture data, viewing summary and detail information for each packet. Ethereal has several powerful features, including a rich display filter language and the ability to view the reconstructed stream of a TCP session." (http://www.ethereal.com/) Some of the features include the ability to read capture files from tcpdump, as well as other network sniffing programs. The graphical user interface in Ethereal makes it easy to find the specific type of data you are looking for by breaking down the packet into a tree structure based on headers. This would be the IP header, the TCP header, and the HTTP Data. You will be able to find more information on Ethereal, from the following link: http://www.ethereal.com/introduction.html - features

Snort:

"Snort is an open source network intrusion detection system, capable of performing real-time traffic analysis and packet logging on IP networks. It can perform protocol analysis, content searching/matching and can be used to detect a variety of attacks and probes, such as buffer overflows, stealth port scans, CGI attacks, SMB probes, OS fingerprinting attempts, and much more.

Snort uses a flexible rules language to describe traffic that it should collect or pass, as well as a detection engine that utilizes a modular plugin architecture. Snort has a real-time alerting capability as well, incorporating alerting

mechanisms for syslog, a user specified file, a UNIX socket, or WinPopup messages to Windows clients using Samba's smbclient.

Snort has three primary uses. It can be used as a straight packet sniffer like tcpdump, a packet logger (useful for network traffic debugging, etc), or as a full blown network intrusion detection system." (http://www.snort.org/about.html)

Snort will run on many Linux distributions as well as Win9x/NT/2000. You can find the most up-to-date information at the following link: http://www.snort.org/

Conclusion:

As you can see by the few examples that have been illustrated in this paper, tcpdump is a valuable tool. There are many filtering variations and command options available allowing you to analyze the data that is moving across the network. As a whole, you may not want to limit yourself to tcpdump. The other tools mentioned can be equally valuable. As in all things, it is important to select the right tool for the right job. Overall, tcpdump is a very flexible, powerful application available to assist you in analyzing network traffic. For further reading, the manpages are a very good resource for more detailed information. There are some very informative Internet resources in the reference section as well, to assist you in solidifying your understanding of network analyzing tools, and their various uses.

Sources:
"Security Essentials Introduction" SANS online training program.

"Monitoring with tcpdump." Internet End-to-end Performance Monitoring.
URL: http://www-iepm.slac.stanford.edu/monitoring/passive/tcpdump.html
Revised 17 Aug. 1999.

Matulis, Peter. "Understanding tcpdump."
URL: http://www.aei.ca/~pmatulis/pub/pack_cap/pack_cap1.html Revised 19 July 2002.

Jacobson, Van, Leres, Craig, and McCanne, Steven. "tcpdump – dump traffic on a network."
URL: http://www.tcpdump.org/tcpdump_man.html (30 Sept. 2003).

"Dsniff 'n the Mirror."
URL: http://www.linuxsecurity.com/feature_stories/dsniff/ripped.html (28 Sept. 2003).

Brown, Joel, Crossfield, Jeanie, Katri, Cindy and Nash, Tom. "Using tcpdump to Analyze an FTP Transfer."
URL: http://www.humboldt.edu/~itl/equipint/Software/TCP-DUMP/group6proj.htm (28 Sept. 2003).

Owl River Company.
URL: http://www.owlriver.com/tips/tcpdump-tech/ Revised 22 Sept. 2002.

Microsoft Corporation. "Explanation of the Three-Way Handshake via TCP/IP."
URL: http://support.microsoft.com/default.aspx?scid=KB;EN-US;Q172983&LN=EN-US Reviewed 12 May 2003 (2.0)

Firetower Information Security. "Tutorial: tcpdump."
URL: http://www.firetower.com/forum/tcpdump.html (6 Oct. 2003)

Internet Security Systems "Packet Sniffing"
URL:http://www.iss.net/security_center/advice/Underground/Hacking/Methods/Technical/Packet_sniffing/default.htm (13 December 2003)

TCP's Three-way handshake
http://ruadh.hypermart.net/Classes/NetEss/ThreeWay.htm (13 December 2003)

Other internet resources:
http://www.cromwell-intl.com/security/monitoring.html
http://freshmeat.net/projects/libpcap/
http://www.tcpdump.org.
http://www.ethereal.com/
http://www.snort.org/

Last Updated: July 20th, 2017

Upcoming Training

SANSFIRE 2017	Washington, DC	Jul 22, 2017 - Jul 29, 2017	Live Event
SANS San Antonio 2017	San Antonio, TX	Aug 06, 2017 - Aug 11, 2017	Live Event
SANS Prague 2017	Prague, Czech Republic	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Boston 2017	Boston, MA	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Salt Lake City 2017	Salt Lake City, UT	Aug 14, 2017 - Aug 19, 2017	Live Event
SANS New York City 2017	New York City, NY	Aug 14, 2017 - Aug 19, 2017	Live Event
Community SANS Omaha SEC401*	Omaha, NE	Aug 14, 2017 - Aug 19, 2017	Community SANS
SANS Virginia Beach 2017	Virginia Beach, VA	Aug 21, 2017 - Sep 01, 2017	Live Event
SANS Adelaide 2017	Adelaide, Australia	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS Chicago 2017	Chicago, IL	Aug 21, 2017 - Aug 26, 2017	Live Event
Community SANS Trenton SEC401	Trenton, NJ	Aug 21, 2017 - Aug 26, 2017	Community SANS
Virginia Beach 2017 - SEC401: Security Essentials Bootcamp Style	Virginia Beach, VA	Aug 21, 2017 - Aug 26, 2017	vLive
Community SANS Pasadena SEC401 @ NASA	Pasadena, CA	Aug 23, 2017 - Aug 30, 2017	Community SANS
Mentor Session - SEC401	Minneapolis, MN	Aug 29, 2017 - Oct 10, 2017	Mentor
SANS San Francisco Fall 2017	San Francisco, CA	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS Tampa - Clearwater 2017	Clearwater, FL	Sep 05, 2017 - Sep 10, 2017	Live Event
Mentor Session - SEC401	Edmonton, AB	Sep 06, 2017 - Oct 18, 2017	Mentor
SANS Network Security 2017	Las Vegas, NV	Sep 10, 2017 - Sep 17, 2017	Live Event
Mentor Session - SEC401	Ventura, CA	Sep 11, 2017 - Oct 12, 2017	Mentor
Community SANS Albany SEC401	Albany, NY	Sep 11, 2017 - Sep 16, 2017	Community SANS
Community SANS Dallas SEC401	Dallas, TX	Sep 18, 2017 - Sep 23, 2017	Community SANS
Community SANS Columbia SEC401	Columbia, MD	Sep 18, 2017 - Sep 23, 2017	Community SANS
SANS Copenhagen 2017	Copenhagen, Denmark	Sep 25, 2017 - Sep 30, 2017	Live Event
Community SANS Boise SEC401	Boise, ID	Sep 25, 2017 - Sep 30, 2017	Community SANS
Baltimore Fall 2017 - SEC401: Security Essentials Bootcamp Style	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	vLive
Community SANS New York SEC401	New York, NY	Sep 25, 2017 - Sep 30, 2017	Community SANS
Rocky Mountain Fall 2017	Denver, CO	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS London September 2017	London, United Kingdom	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Baltimore Fall 2017	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	Live Event
Community SANS Sacramento SEC401	Sacramento, CA	Oct 02, 2017 - Oct 07, 2017	Community SANS
SANS DFIR Prague 2017	Prague, Czech Republic	Oct 02, 2017 - Oct 08, 2017	Live Event