Interested in learning more about security?

SANS Institute InfoSec Reading Room

Analyzing Network Traffic With Basic Linux Tools

Tools to analyze network traffic can be expensive, complicated, and may require preparation before an investigation begins. By leveraging tools easily available in every Linux distribution (and often in UNIX/Mac OS X) combined with Tcpdump to analyze network traffic, you can determine the make-up of the network traffic in question find the most active hosts and protocols, search for oddities, and determine the most efficient next step of your investigation. Using this method, you are able to pare away the normal and mu...

Analyzing pcap with Linux

GIAC (GCIA) Gold Certification

Author: Travis Green, travis@travisgreen.net
Advisor: Rob VandenBrink, RVandenBrink@metafore.ca

Accepted: August 4th 2012

Abstract

Tools to analyze network traffic can be expensive, complicated, and may require preparation before an investigation begins. By leveraging tools easily available in every Linux distribution (and often in UNIX/Mac OS X) combined with Tcpdump to analyze network traffic, you can determine the make-up of the network traffic in question find the most active hosts and protocols, search for oddities, and determine the most efficient next step of your investigation. Using this method, you are able to pare away the normal and mundane to reveal and examine the unexpected.

1. Introduction

When examining network traffic, one may examine the packets individually with Tcpdump, or reconstruct it with sophisticated and sometimes expensive tools. It is extremely useful to quickly examine traffic in different ways, obtaining a feel for the traffic's makeup, and formulate an approach for in-depth analysis. No single tool can really accommodate this wish list, but the power and flexibility of the Bash shell and the GNU core utilities together with Tcpdump can provide this functionality. This combination of tools allows quick creation of statistics from the captured traffic as a

whole, or for particular hosts or protocols. You can search the payload of packets for strings, call out particular fields in a protocol that might be in plain text, such as "User Agent:" in HTTP or logging facility in syslog traffic. Also, you can exclude traffic that is expected from a capture file, and then examine the traffic that is left over, allowing you to sift through and find things that may be unexpected. Finally, basic methods of managing a large pcap archive will be examined.

2. Method

2.1. Getting Started

The elements that are going to be brought together here are pcap data, a tool to read that pcap data, Tcpdump, and the GNU coreutils. The Bash shell will bring them all together. The following sections cover the necessary ingredients in detail.

2.1.1. The Linux/UNIX terminal

Security focused distributions such as Knoppix Security Tools Distribution (STD) and Backtrack have required tools built in. You may need to add Tcpdump to other distributions. The easiest way to add Tcpdump is using your Linux distribution's package manager. Examples will us Ubuntu 12.04, which includes Tcpdump:

```
example@ubuntu:~$ sudo apt-get install tcpdump
Reading package lists... Done
Building dependency tree
Reading state information... Done
tcpdump is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 69 not upgraded.
example@ubuntu:~$ _
```

The feature that brings these different tools together is called I/O redirection. All Bash shells support I/O redirection, a way of taking output from one program and using it as input to another program, which can then give output to another program, and so on. For the analysis method being described here, Tcpdump will read a binary capture of network traffic, a pcap file, and generate a stream of plaintext from it. Then that plaintext output is redirected into a combination of other programs to obtain the desired output. The other programs in this method

manipulating the stream of text are the GNU coreutils. The Bash shell and GNU coreutils are packaged together as part of GNU/Linux, which you are likely using if you are using any popular Linux distribution. You may recognize many of these already because they make up the vast majority of everyday shell commands such as mkdir, chown, cut, wc, uniq, cat and ls. See the gnu.org manual page for coretuils for a complete listing (Free Software Foundation Inc., 2008).

2.1.2. Acquiring pcap data

At the core of this analysis method is pcap data. The pcap format was developed by the developers of Tcpdump at Lawrence Berkeley National Laboratory in 1987 (Ali, 2010), and because of this, many different network based programs can read and write in this format. You can use any of these programs to acquire pcap data, but the easiest way in this example is to use Tcpdump on our local computer to write directly to pcap files:

```
example@ubuntu:~$ sudo tcpdump -w capture.pcap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 65535 byte
s
^C745 packets captured
745 packets received by filter
0 packets dropped by kernel
example@ubuntu:~$ _
```

Another simple way is to use Wireshark, start a capture, then save the capture file. More advanced usage includes using the pcap log files from Snort alert logs, or performing a packet capture from a router or switch (Cisco, 2007).
For testing purposes, pcap files are also available for download from several online repositories.

2.2. Generating statistics

I/O redirection is accomplished on the command line using the pipe character ("|" is shift+\ on most US keyboards). A few basic examples of combining programs using pipes to generate relevant statistics are demonstrated, and then more advanced scenarios. Readers are encouraged to their own variations. Indulging one's curiosity and playing with the way these commands fit together is an excellent way to find

new and interesting ways to view traffic. You never know what useful things you will discover.

2.2.1. The basic formula

To start, count the number of packets in a particular pcap file with the following command:

```
example@ubuntu:~$ tcpdump -nn -r capture.pcap | wc -l
reading from file capture.pcap, link-type EN10MB (Ethernet)
12365688
example@ubuntu:~$ _
```

The program named wc takes the output of Tcpdump as input and counted the lines, and there are 12365688 of them. Another program, cut, can separate a particular part of output before piping it into another program. First, look at a sample of output from Tcpdump without modification:

```
example@ubuntu:~$ tcpdump -nn -r capture.pcap | head
reading from file capture.pcap, link-type EN10MB (Ethernet)
01:20:39.145954 IP 0.0.0.0.68 > 255.255.255.255.67: BOOTP/DHCP, Request from 00:
26:52:96:06:c0, length 576
01:20:40.466565 IP 10.1.1.17.59631 > 239.255.255.250.1900: UDP, length 97
01:20:40.511006 STP 802.1d, Config, Flags [none], bridge-id 8001.00:24:f7:82:d4:
00.8014, length 43
01:20:41.388025 IP 10.1.1.17.1900 > 239.255.255.250.1900: UDP, length 385
01:20:41.458781 IP 10.1.1.17.1900 > 239.255.255.250.1900: UDP, length 357
01:20:42.150260 IP 0.0.0.0.68 > 255.255.255.255.67: BOOTP/DHCP, Request from 00:
26:52:96:06:c0, length 576
01:20:42.517164 STP 802.1d, Config, Flags [none], bridge-id 8001.00:24:f7:82:d4:
00.8014, length 43
01:20:42.700055 IP 10.1.1.17.1900 > 239.255.255.250.1900: UDP, length 314
01:20:42.739956 IP 10.1.1.17.1900 > 239.255.255.250.1900: UDP, length 373
01:20:42.779669 IP 10.1.1.17.1900 > 239.255.255.250.1900: UDP, length 369
example@ubuntu:~$ _
```

The basic structure of Tcpdump output is:

```
[timestamp] [network protocol] [source IP].[source port] > [dest IP].[dest port]
```

From this output sample, you can find the field you want to focus on, and then count how many columns from the left that desired field is. It is important to note what sort of character separates the columns, in this example it is a space. Note that we tell the cut program what character by using the –d [delimiter] flag. Also of note is the use of the command head, which shows only the first 10 lines of output. Using head not only keeps the output from blasting your screen, it will also limit execution output. Commands that might take quite a while to complete will be limited to specified output parameters. This allows an investigator to try each step quickly

without processing the entire file. While in a real investigation, one would process the entire pcap file, this eliminates it from the command line. For an example, select only the source IP address with port, which is the third column, with the following command:

```
example@ubuntu:~$ tcpdump -nn -r capture.pcap | cut -f 3 -d " " | head
reading from file capture.pcap, link-type EN10MB (Ethernet)
0.0.0.0.68
10.1.1.17.59631
802.1d,
10.1.1.17.1900
10.1.1.17.1900
0.0.0.0.68
802.1d,
10.1.1.17.1900
10.1.1.17.1900
10.1.1.17.1900
example@ubuntu:~$ _
```

To filter to just TCP/IP traffic and exclude layer 2 traffic, add the Tcpdump filter of 'tcp or udp':

```
example@ubuntu:~$ tcpdump -nn -r capture.pcap 'tcp or udp' | cut -f 3 -d " " | h
ead
reading from file capture.pcap, link-type EN10MB (Ethernet)
0.0.0.0.68
10.1.1.17.59631
10.1.1.17.1900
10.1.1.17.1900
0.0.0.0.68
10.1.1.17.1900
10.1.1.17.1900
10.1.1.17.1900
10.1.1.17.1900
10.1.1.17.1900
example@ubuntu:~$ _
```

One more step is to use only the IP address and remove the source port by adding another cut that selects the first 4 columns separated by the "." character:

```
example@ubuntu:~$ tcpdump -nn -r capture.pcap 'tcp or udp' | cut -f 3 -d " " | c
ut -f 1-4 -d "." | head
reading from file capture.pcap, link-type EN10MB (Ethernet)
0.0.0.0
10.1.1.17
10.1.1.17
10.1.1.17
0.0.0.0
10.1.1.17
10.1.1.17
10.1.1.17
10.1.1.17
10.1.1.17
example@ubuntu:~$ _
```

2.2.2. Introducing uniq

Uniq, also part of the GNU coreutils family, is used to eliminate adjacent, duplicate lines of text. Note the requirement of "adjacent", which means you need to sort any input that goes into uniq, so that it will know if they are unique or just appear in different parts of the pcap file. Continuing from the last example of source IP addresses:

```
example@ubuntu:~$ tcpdump -nn -r capture.pcap 'tcp or udp' | cut -f 3 -d " " | c
ut -f 1-4 -d "." | sort | uniq | head
reading from file capture.pcap, link-type EN10MB (Ethernet)
0.0.0.0
100.43.116.142
10.10.10.3
101.103.15.175
101.103.162.193
101.103.27.231
101.108.245.139
101.108.86.46
101.109.141.106
10.1.1.15
example@ubuntu:~$ _
```

This example command line found all source IP addresses in the pcap file and listed them individually. You can change the cut statements to find out other information. For example, if you wanted to see the destination IP addresses rather than sources, you'd change the first cut statement to cut –f 5, so that it is selecting the 5th column instead.

```
example@ubuntu:~$ tcpdump -nn -r capture.pcap 'tcp or udp' | cut -f 5 -d " " | c
ut -f 1-4 -d "." | sort | uniq | head
reading from file capture.pcap, link-type EN10MB (Ethernet)
100.43.116.142
10.10.10.3
101.0.49.126
101.103.15.175
101.103.162.193
101.103.27.231
101.108.157.244
101.108.245.139
101.108.86.46
101.109.111.170
example@ubuntu:~$ _
```

A very useful feature of uniq is that it will also count the number of instances it found, so you can use it to see the top number of X. For example, the top 10 destination IP

addresses:

```
example@ubuntu:~$ tcpdump -nn -r capture.pcap 'tcp or udp' | cut -f 5 -d " " | c
ut -f 1-4 -d "." | sort | uniq -c | sort -nr | head
reading from file capture.pcap, link-type EN10MB (Ethernet)
5341255 10.1.1.35
3732995 10.1.1.35:
 514540 24.35.37.164
 346805 82.77.103.21
 198281 68.58.223.188
 194046 14.2.29.61
 182528 94.174.4.76
 165131 124.170.15.235
 119965 69.16.169.100
 118525 98.160.217.21
example@ubuntu:~$
```

Note that you sort the list a second time with the "-nr" flag to show the output sorted in descending order by numerical value.

To examine destination ports, start by selecting only destination IPs and ports for new TCP sessions using a Tcpdump filter of 'tcp[13]=2' which selects only packets with the SYN flag set. That way you don't accidentally give undue weight to commonly used ports like 443 and 80, where there may be a large number of packets over very few sessions as in the case of a HTTP or HTTPS download:

```
example@ubuntu:~$ tcpdump -nn -r capture.pcap 'tcp[13]=2' | cut -f 5 -d " " | so
rt | uniq -c | sort -nr | head
reading from file capture.pcap, link-type EN10MB (Ethernet)
   1762 184.107.[redacted].8118:
    788 180.94.73.2.59928:
    599 122.108.123.101.20696:
    535 124.195.202.220.28026:
    531 173.135.133.36.63955:
    351 59.182.69.224.46138:
    315 180.94.82.42.32149:
    311 87.0.67.215.40959:
    287 41.119.220.166.63942:
    286 110.33.196.233.35829:
example@ubuntu:~$ _
```

Now you've got the top destinations, you can use cut to select only the port:

```
example@ubuntu:~$ tcpdump -nn -r capture.pcap 'tcp[13]=2' | cut -f 5 -d " " | cu
t -f 5 -d "." | sort | uniq -c | sort -nr | head
reading from file capture.pcap, link-type EN10MB (Ethernet)
   1762 8118:
    805 59928:
    629 28026:
    610 63955:
    599 20696:
    469 6881:
    384 63942:
    365 40959:
    351 46138:
    348 80:
example@ubuntu:~$ _
```

Travis Green, travis@travisgreen.net

This example shows port 8118 as a top destination, and from that it can be inferred that we have a user utilizing a Privoxy server. The appearance of port 6881 shows that the Bittorrent protocol is in use. The SANS Internet Storm Center Port Details (SANS ISC, 2012) site is a great resource for determining common port numbers and corresponding protocols and programs. In this site, one may find well known port information, CVEs by port, and number of attacks for each port by source and destination.

We can do the same with source IP addresses and see who the top talkers are:

```
example@ubuntu:~$ tcpdump -nn -r capture.pcap 'tcp[13]=2' | cut -f 3 -d " " | cu
t -f 1-4 -d "." | sort | uniq -c | sort -nr | head
reading from file capture.pcap, link-type EN10MB (Ethernet)
  25730 10.1.1.35
     14 10.1.1.17
      9 111.68.38.182
      8 76.19.200.54
      8 75.73.51.44
      8 178.61.230.221
      6 93.73.56.104
      6 218.82.76.138
      5 72.27.120.168
      4 82.204.100.32
example@ubuntu:~$ _
```

You can do the same procedure to see our top source ports, but often they are random or sequential, and of marginal utility.

2.3. Searching in plain text payloads

Many network protocols store their data as plain text in the payload portion of a packet (SMTP, Syslog, POP3, FTP ASCII mode, HTTP, DNS, etc), and Tcpdump can display this text by using the -A switch:

```
example@ubuntu:~$ tcpdump -Ann -r capture.pcap 'dst port 25 or dst port 514 or p
ort 110 or dst port 21 or dst port 53 or dst port 80' | head -15
reading from file capture.pcap, link-type EN10MB (Ethernet)
01:21:07.471626 IP 10.1.1.35.7361 > 199.47.219.151.80: Flags [P.], seq 419759956
2:4197599778, ack 108083525, win 64860, length 216
E...1.@....O
..#./.....P.2IJ.q9EP..\Y...GET /subscribe?host_int=346408111&ns_map=46731833_364
037179773497,95228564_8685163156&ts=1348906874 HTTP/1.1
Host: notify20.dropbox.com
Accept-Encoding: identity
Connection: keep-alive
X-Dropbox-Locale: en_US


01:21:53.047676 IP 10.1.1.35.49599 > 8.8.8.8.53: 32044+ A? fpdownload.macromedia
.com. (43)
E..G1.......
..#.......5.3.@},..........
fpdownload
macromedia.com.....
01:22:03.474715 IP 10.1.1.35.7361 > 199.47.219.151.80: Flags [P.], seq 216:432,
ack 180, win 64681, length 216
example@ubuntu:~$ _
```

Pipe the ASCII output of Tcpdump into grep and then use grep to look for a particular text string. If you want to make a rudimentary IDS, use a regular expression with grep. If you have found a host making a HTTP request for a malicious file you can use “grep -i” with the file name to see if other hosts have requested the same file. This will be covered more in-depth in the “Examining HTTP” section.

3. Examining abnormal traffic

Another useful technique is analyzing the abnormal traffic. Normal traffic is defined as being created through use of widely available software being used for its intended purpose. For every network protocol there is a Request For Comment (RFC) produced by the IETF that defines how network applications should work (ITEF, 2012), and most commercial software honors these definitions. Abnormal traffic would then be considered network traffic that does not comply with RFC or complies in a way designed to gain some other outcome.

Removing normal and expected network traffic, then examining the remaining traffic is a powerful technique to find things that cannot be anticipated.

Travis Green, travis@travisgreen.net

3.1. Top Level Domains

Country code top level domain names (ccTLD) are more commonly abused inside the USA (Kadam, 2012) and one's knowledge of this abuse can be used to the investigator's advantage. Search the captured traffic for DNS and then cut, sort, and uniq the results to find the most frequently resolved:

```
example@ubuntu:~$ tcpdump -nn -r capture.pcap 'port 53' | head -5
reading from file capture.pcap, link-type EN10MB (Ethernet)
01:21:53.047676 IP 10.1.1.35.49599 > 8.8.8.8.53: 32044+ A? fpdownload.macromedia
.com. (43)
01:21:53.215131 IP 8.8.8.8.53 > 10.1.1.35.49599: 32044 4/0/0 CNAME fpdownload.wi
p4.adobe.com., CNAME fpdownload.macromedia.com.edgekey.net., CNAME e526.d.akamai
edge.net., A 2.20.210.70 (178)
01:26:17.871158 IP 10.1.1.35.49681 > 8.8.8.8.53: 6366+ A? softwareupdate.vmware.
com. (43)
01:26:18.040663 IP 8.8.8.8.53 > 10.1.1.35.49681: 6366 3/0/0 CNAME softwareupdate
.vmware.com.edgekey.net., CNAME e751.d.akamaiedge.net., A 2.20.211.51 (142)
01:32:32.389622 IP 10.1.1.35.56988 > 8.8.8.8.53: 6116+ A? safebrowsing.clients.g
oogle.com. (49)
example@ubuntu:~$ _
```

Then add a grep statement to exclude your more commonplace TLDs:

```
example@ubuntu:~$ tcpdump -nn -r capture.pcap 'port 53' | grep -Ev '(com|net|org
|gov|mil|arpa)' | cut -f 9 -d " " | head
reading from file capture.pcap, link-type EN10MB (Ethernet)
69.16.169.99
69.16.169.99
69.16.169.95,
69.16.169.98
69.16.169.113,
74.209.160.12
74.209.160.12
74.125.139.105,
74.125.139.105,
184.107.18.203
example@ubuntu:~$
example@ubuntu:~$ _
```

Finally, filter for names only, not IP addresses:

```
example@ubuntu:~$ tcpdump -nn -r capture.pcap 'port 53' | grep -Ev '(com|net|org
|gov|mil|arpa)' | cut -f 9 -d " " | grep -E '[a-z]'
reading from file capture.pcap, link-type EN10MB (Ethernet)
anycast.filepicker.io.,
anycast.filepicker.io.,
anycast.filepicker.io.,
anycast.filepicker.io.,
[14953a]
example@ubuntu:~$ _
```

After you have found some suspect names you'd like to investigate, simply go back through the capture file using Tcpdump piped to grep, which you'll use to search for your suspect domain name as outlined previously.

Travis Green, travis@travisgreen.net

3.2. Examining HTTP

3.2.1. HTTP Methods

This formula also applies to the HTTP protocol. First, remove the most common and "safe" methods (Fielding, 1999) as defined by the W3C. Searching for "normal" HTTP methods:

```
example@ubuntu:~$ tcpdump -Ann -r capture.pcap 'dst port 80' | head -15
reading from file capture.pcap, link-type EN10MB (Ethernet)
01:21:07.471626 IP 10.1.1.35.7361 > 199.47.219.151.80: Flags [P.], seq 419759956
2:4197599778, ack 108083525, win 64860, length 216
E...1.@....0
..#./.....P.2IJ.q9EP..\Y...GET /subscribe?host_int=346408111&ns_map=46731833_364
037179773497,95228564_8685163156&ts=1348906874 HTTP/1.1
Host: notify20.dropbox.com
Accept-Encoding: identity
Connection: keep-alive
X-Dropbox-Locale: en_US


01:22:03.474715 IP 10.1.1.35.7361 > 199.47.219.151.80: Flags [P.], seq 216:432,
ack 180, win 64681, length 216
E...1.@.....
..#./.....P.2J".q9.P...\...GET /subscribe?host_int=346408111&ns_map=46731833_364
037179773497,95228564_8685163156&ts=1348906930 HTTP/1.1
Host: notify20.dropbox.com
Accept-Encoding: identity
Connection: keep-alive
example@ubuntu:~$ _
```

Removing GET and HEAD methods:

```
example@ubuntu:~$ tcpdump -Ann -r capture.pcap 'dst port 80' | grep 'HTTP' | gre
p -Ev '(GET|HEAD)' | head
reading from file capture.pcap, link-type EN10MB (Ethernet)
..#J}-....P.ec.....P.@.I...POST /safebrowsing/downloads?client=navclient-auto-ff
ox&appver=15.0.1&pver=2.2&wrkey=AKEgNisrJpx6XuyXeVpInBdyGMiadIAco2WVfkEojbK72J53
2_bETLmPbYwmOl6wJFqqPOOJxjeuFH35axf99RHnQPOrA-x-Ig== HTTP/1.1
..#J}-e...P........P.@.[...POST /safebrowsing/downloads?client=navclient-auto-ff
ox&appver=15.0.1&pver=2.2&wrkey=AKEgNisrJpx6XuyXeVpInBdyGMiadIAco2WVfkEojbK72J53
2_bETLmPbYwmOl6wJFqqPOOJxjeuFH35axf99RHnQPOrA-x-Ig== HTTP/1.1
..#J}.f. .PV$.....RP.@..e..POST /safebrowsing/downloads?client=navclient-auto-ff
ox&appver=15.0.1&pver=2.2&wrkey=AKEgNisrJpx6XuyXeVpInBdyGMiadIAco2WVfkEojbK72J53
2_bETLmPbYwmOl6wJFqqPOOJxjeuFH35axf99RHnQPOrA-x-Ig== HTTP/1.1
..#..3H...P"..z..dbP....._..POST / HTTP/1.1
..#..3H...P..U)..{.P.......POST / HTTP/1.1
...P.Gu...b.P..ia...POST /speedtest/upload.php?x=0.7980574979446828 HTTP/1.1
.BWP..i....POST /speedtest/upload.php?x=0.6772549194283783 HTTP/1.1
.BWP..i....POST /speedtest/upload.php?x=0.6772549194283783 HTTP/1.1
...P.N....cGP...a...POST /speedtest/upload.php?x=0.7178841331042349 HTTP/1.1
.C.P...]...POST /speedtest/upload.php?x=0.22267218213528395 HTTP/1.1
example@ubuntu:~$ _
```

Another field of HTTP that is of interest is the "referer" field. Please note that referrer is misspelled in the HTTP protocol (Fielding, 1999):

```
example@ubuntu:~$ tcpdump -Ann -r capture.pcap 'port 80' | grep -i 'referer' | h
ead
reading from file capture.pcap, link-type EN10MB (Ethernet)
Referer: http://members.easynews.com/global5/search.html
Referer: http://members.easynews.com/global5/search.html?gps=gtd&sbj=&from=&ns=&
fil=&fex=&vc=&ac=&s1=dtime&s1d=-&s2=nrfile&s2d=%2B&s3=dsize&s3d=%2B&pby=100&pno=
1&sS=0&u=1&svL=&d1=&d1t=&d2=&d2t=&b1=&b1t=&b2=&b2t=&px1=&px1t=&px2=&px2t=&fps1=&
fps1t=&fps2=&fps2t=&bps1=&bps1t=&bps2=&bps2t=&hz1=&hz1t=&hz2=&hz2t=&rn1=&rn1t=&r
n2=&rn2t=&fly=2
Referer: http://members.easynews.com/global5/search.html?gps=gtd&sbj=&from=&ns=&
fil=&fex=&vc=&ac=&fty%5B%5D=VIDEO&s1=dtime&s1d=-&s2=nrfile&s2d=%2B&s3=dsize&s3d=
%2B&pby=100&pno=1&sS=0&u=1&svL=&d1=&d1t=&d2=&d2t=&b1=&b1t=&b2=&b2t=&px1=&px1t=&p
x2=&px2t=&fps1=&fps1t=&fps2=&fps2t=&bps1=&bps1t=&bps2=&bps2t=&hz1=&hz1t=&hz2=&hz
2t=&rn1=&rn1t=&rn2=&rn2t=&fly=2
Referer: http://www.speedtest.net/
Referer: http://www.speedtest.net/
Referer: http://www.speedtest.net/
Referer: http://www.speedtest.net/
Referer: http://www.speedtest.net/
Referer: http://www.speedtest.net/
Referer: http://www.speedtest.net/
example@ubuntu:~$ _
```

If you found a malicious URL, you could examine the referrer field to see what other request had caused the malicious request.

3.2.2. User-Agent field

Malware authors may be using unique, unusual, or malformed "User-Agent" string when making HTTP GET requests. (Manners, 2011). To see all the user-agent fields in our capture file use the following command:

```
example@ubuntu:~$ tcpdump -Ann -r capture.pcap 'port 80' | grep -Ei 'user-agent'
 | sort | uniq -c | sort -nr | head -15
reading from file capture.pcap, link-type EN10MB (Ethernet)
    202 User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:15.0) Gecko/20100101
Firefox/15.0.1
     22 User-Agent: FDM 3.x
     17 User-Agent: BTWebClient/3200(27886)
     15 User-Agent: Windows-Update-Agent
     11 User-Agent: uTorrent/3200(27886)
      6 User-Agent: Microsoft-CryptoAPI/6.1
      4 User-Agent: Windows-Media-Player/12.0.7601.17514
      3 User-Agent: MSDW
      2 User-Agent: Skype. 5.10
      2 User-Agent: NSPlayer/7.10.0.3059
      2 User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Tr
ident/5.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Me
dia Center PC 6.0)
      2 User-Agent: GBXHTTP Client/1.0
      1 ..#X..  . .P9.$..z..P...9N..User-Agent: VLC/2.0.1 LibVLC/2.0.1
      1 ..#X..  ...P:...1CkvP.......User-Agent: VLC/2.0.1 LibVLC/2.0.1
      1 User-Agent: Valve/Steam HTTP Client 1.0
example@ubuntu:~$ _
```

There are many ways to analyze "User-agent:" and spending one's time doing so is useful to the investigator (). In the previous example, we can see that there is a

variety of User-Agents at work on our network, torrent software, skype, Windows Media Player and others. Several websites catalog user agent strings and are can be found via web search. On our output, one User-Agent stands out as somewhat irregular; “GBXHTTP”. At the time of writing, this was not listed in any User-Agent string database, but further analysis reveals it is the GearBox Software downloader:

```
example@ubuntu:~$ tcpdump -Ann -r capture.pcap 'port 80' | sed -n '/GBXHTTP/,$p'
 | head
reading from file capture.pcap, link-type EN10MB (Ethernet)
User-Agent: GBXHTTP Client/1.0
X-Last-Request-Retries: 0
X-Last-Request-Response-Time-Millis: 0
X-Recent-Failed-Requests: 0
X-Build-Identifier: WILLOW2-PCSHIP-35-CL660067
Host: cdn.services.gearboxsoftware.com


08:40:22.208921 IP 10.1.1.35.18849 > 69.16.169.100.80: Flags [.], ack 14512201,
win 64870, options [nop,nop,sack 3 {14525601:14539001}{14516221:14522921}{145135
41:14514881}], length 0
E..D>O@.....
example@ubuntu:~$ _
```

The “Host:” field shows us we probably talking with a software company’s content distribution network. Using the “--context=5” flag for grep, we can look at a bit more of the HTTP GET:

```
example@ubuntu:~$ tcpdump -Ann -r capture.pcap 'port 80' | grep -Ei 'GBXHTTP' --
context=5 | head
reading from file capture.pcap, link-type EN10MB (Ethernet)

08:40:21.968229 IP 10.1.1.35.19136 > 54.240.162.104.80: Flags [P.], seq 1:332, a
ck 1, win 64860, length 331
E..s=.@.....
..#6..hJ..P....<..jP..\....GET /sparktms/willow2/pc/steam/WillowTMS.cfg HTTP/1.1
Connection: Keep-Alive
Accept: application/octet-stream
User-Agent: GBXHTTP Client/1.0
X-Last-Request-Retries: 0
X-Last-Request-Response-Time-Millis: 0
X-Recent-Failed-Requests: 0
X-Build-Identifier: WILLOW2-PCSHIP-35-CL660067
example@ubuntu:~$
example@ubuntu:~$ _
```

Now we know the IP address, and a reverse lookup of the IP confirms the theory it is a content distribution network, as Cloudfront is the name of Amazon.com’s CDN service:

```
example@ubuntu:~$ host 54.240.162.104
104.162.240.54.in-addr.arpa domain name pointer server-54-240-162-104.fra6.r.clo
udfront.net.
example@ubuntu:~$ _
```

Travis Green, travis@travisgreen.net

4. Scaling up

4.1. Creating a pcap repository

When you want to capture network traffic over a long period of time, this author highly recommends creating a pcap repository or archive. Before Tcpdump 3.7.1 there was no automatic way to change and manage the logs. However, after 3.7.1, the authors of Tcpdump have made things very easy on us by providing the –C [MBytes] and –W [number of files] switches.

```
example@ubuntu:~$ sudo -b tcpdump -nn -i eth0 -w capture.pcap -C 1 -W 5
example@ubuntu:~$ tcpdump: listening on eth0, link-type EN10MB (Ethernet), captu
re size 65535 bytes
^C
example@ubuntu:~$ ls -l
total 1308
-rw-r--r-- 1 root root 1000579 Oct  2 22:31 capture.pcap0
-rw-r--r-- 1 root root  334062 Oct  2 22:32 capture.pcap1
example@ubuntu:~$ _
```

The –C switch tells Tcpdump to start a new file after n million bytes, and –W tells Tcpdump to only keep n files on hand. After 3.9.1, Tcpdump will do the same behavior but with time using the –G [seconds] flag.

```
example@ubuntu:~$ sudo -b tcpdump -nn -i eth0 -w capture-%s.pcap -G 600 -W 5
example@ubuntu:~$ tcpdump: listening on eth0, link-type EN10MB (Ethernet), captu
re size 65535 bytes

example@ubuntu:~$
example@ubuntu:~$ ls -l
total 6432
-rw-r--r-- 1 root root 1171256 Oct  2 22:43 capture-1349242433.pcap
-rw-r--r-- 1 root root  980419 Oct  2 22:53 capture-1349243033.pcap
-rw-r--r-- 1 root root  397312 Oct  2 22:58 capture-1349243633.pcap
-rw-r--r-- 1 root root 1000579 Oct  2 22:31 capture.pcap0
-rw-r--r-- 1 root root 1000073 Oct  2 22:37 capture.pcap1
-rw-r--r-- 1 root root 1000407 Oct  2 22:48 capture.pcap2
-rw-r--r-- 1 root root 1000411 Oct  2 22:58 capture.pcap3
-rw-r--r-- 1 root root   10599 Oct  2 22:58 capture.pcap4
example@ubuntu:~$ _
```

4.2. Dealing with groups of pcap files

If you're using a pcap repository, you'll find it useful to be able to run the same Tcpdump command against a set of files. This is easy to do with the "for" loop built

in to Bash.

```
example@ubuntu:~$ for something in a b c; do echo $something; done;
a
b
c
example@ubuntu:~$ for output in $(echo a b c); do echo $output; done;
a
b
c
example@ubuntu:~$ _
```

In order to run Tcpdump against all the files that end in .pcap do something like this:

```
example@ubuntu:~$ sudo chown -R example:example *
example@ubuntu:~$ for capfile in $(ls *.pcap); do tcpdump -nnr $capfile 'dst por
t 53' | head -1; done;
reading from file capture-1349242433.pcap, link-type EN10MB (Ethernet)
22:34:02.703114 IP 10.1.1.35.53894 > 8.8.8.8.53: 22769+ A? 250sx.net. (27)
reading from file capture-1349243033.pcap, link-type EN10MB (Ethernet)
22:43:54.506788 IP 10.1.1.35.57079 > 8.8.8.8.53: 29243+ A? 250sx.net. (27)
reading from file capture-1349243633.pcap, link-type EN10MB (Ethernet)
22:53:53.160589 IP 10.1.1.35.57272 > 8.8.4.4.53: 8158+ A? secure.logmein.com. (3
6)
reading from file capture-1349244233.pcap, link-type EN10MB (Ethernet)
23:03:53.468485 IP 10.1.1.35.64273 > 8.8.4.4.53: 55202+ A? 250sx.net. (27)
tcpdump: pcap_loop: truncated dump file; tried to read 345 captured bytes, only
got 235
example@ubuntu:~$ _
```

The trick to successfully employing this “for loop”, is to start as simple as possible and get it to run without error. Then add pieces bit by bit checking it at each step.

5. Conclusion

The method of combining of Tcpdump and GNU coreutils provide multiple interesting ways to see top sources and destinations, as well as ways of searching text based network traffic. Also discussed was examination of edge case traffic scenarios that are outside what may be considered “normal” in DNS traffic, HTTP methods and User-Agent strings. Basics of generating and using a pcap repository as outlined will help you build a base of network traffic to use for investigation, or to establish a baseline for later comparison. These methods rely on free software and require no preparation outside of capturing the traffic and having a Linux/UNIX Bash and the GNU coreutils at the investigator’s disposal. An added benefit is that these are completely free and readily available utilities available to nearly every Linux distribution. This author recommends that readers and adopters of this

method experiment with these recipes by substituting your favorite programs and by using different combinations and selecting different traffic. As stated previously, you never know what interesting things you may find when you use different tools to look.

6. References

Free Software Foundation Inc. (2008, November 03). *Gnu coreutils*. Retrieved from http://www.gnu.org/software/coreutils/manual/html_node/index.html

Ali, M. (2010, MARCH). *Tcpdump and wireshark*. Retrieved from http://www.cs.princeton.edu/courses/archive/spr10/cos461/assignments/sockets/Tcpdump.pdf

Cisco. (2007, November). *Router ip traffic export packet capture enhancements.*. Retrieved from http://www.cisco.com/en/US/docs/ios/12_4t/12_4t11/ht_rawip.html

SANS ISC. (2012, October 23). *Port details*. Retrieved from https://isc.sans.edu/port.html

ITEF. (2012, October 23). *Request for comments (rfc) pages*. Retrieved from http://www.ietf.org/rfc.html

Kadam, P. (2012, November). *Most abused tlds*. Retrieved from http://prathameshkadam.blogspot.com/2012/06/most-abuse-tlds.html

Fielding, R. (1999, June 1). *Rfc 2616 hypertext transfer protocol*. Retrieved from http://tools.ietf.org/html/rfc2616

Travis Green, travis@travisgreen.net

Law, E. (2009, October 7). *The user-agent string: Use and abuse*. Retrieved from http://blogs.msdn.com/b/ieinternals/archive/2009/10/08/extending-the-user-agent-string-problems-and-alternatives.aspx

Manners, D. (2011, October 11). *The user agent field: Analyzing and detecting the abnormal or malicious in your organization*. Retrieved from http://www.sans.org/reading_room/whitepapers/hackers/user-agent-field-analyzing-detecting-abnormal-malicious-organization_33874.pdf

Travis Green, travis@travisgreen.net

Last Updated: August 19th, 2017

Upcoming SANS Training

Click Here for a full list of all Upcoming SANS Events by Location

SANS Chicago 2017	Chicago, ILUS	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS Virginia Beach 2017	Virginia Beach, VAUS	Aug 21, 2017 - Sep 01, 2017	Live Event
SANS San Francisco Fall 2017	San Francisco, CAUS	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS Tampa - Clearwater 2017	Clearwater, FLUS	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS Network Security 2017	Las Vegas, NVUS	Sep 10, 2017 - Sep 17, 2017	Live Event
SANS Dublin 2017	Dublin, IE	Sep 11, 2017 - Sep 16, 2017	Live Event
SANS Baltimore Fall 2017	Baltimore, MDUS	Sep 25, 2017 - Sep 30, 2017	Live Event
Data Breach Summit & Training	Chicago, ILUS	Sep 25, 2017 - Oct 02, 2017	Live Event
SANS Copenhagen 2017	Copenhagen, DK	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS London September 2017	London, GB	Sep 25, 2017 - Sep 30, 2017	Live Event
Rocky Mountain Fall 2017	Denver, COUS	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS SEC504 at Cyber Security Week 2017	The Hague, NL	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS DFIR Prague 2017	Prague, CZ	Oct 02, 2017 - Oct 08, 2017	Live Event
SANS Oslo Autumn 2017	Oslo, NO	Oct 02, 2017 - Oct 07, 2017	Live Event
SANS October Singapore 2017	Singapore, SG	Oct 09, 2017 - Oct 28, 2017	Live Event
SANS AUD507 (GSNA) @ Canberra 2017	Canberra, AU	Oct 09, 2017 - Oct 14, 2017	Live Event
SANS Phoenix-Mesa 2017	Mesa, AZUS	Oct 09, 2017 - Oct 14, 2017	Live Event
Secure DevOps Summit & Training	Denver, COUS	Oct 10, 2017 - Oct 17, 2017	Live Event
SANS Tysons Corner Fall 2017	McLean, VAUS	Oct 14, 2017 - Oct 21, 2017	Live Event
SANS Brussels Autumn 2017	Brussels, BE	Oct 16, 2017 - Oct 21, 2017	Live Event
SANS Tokyo Autumn 2017	Tokyo, JP	Oct 16, 2017 - Oct 28, 2017	Live Event
SANS Berlin 2017	Berlin, DE	Oct 23, 2017 - Oct 28, 2017	Live Event
SANS Seattle 2017	Seattle, WAUS	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS San Diego 2017	San Diego, CAUS	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Gulf Region 2017	Dubai, AE	Nov 04, 2017 - Nov 16, 2017	Live Event
SANS Miami 2017	Miami, FLUS	Nov 06, 2017 - Nov 11, 2017	Live Event
SANS Amsterdam 2017	Amsterdam, NL	Nov 06, 2017 - Nov 11, 2017	Live Event
SANS Milan November 2017	Milan, IT	Nov 06, 2017 - Nov 11, 2017	Live Event
SANS Sydney 2017	Sydney, AU	Nov 13, 2017 - Nov 25, 2017	Live Event
Pen Test Hackfest Summit & Training 2017	Bethesda, MDUS	Nov 13, 2017 - Nov 20, 2017	Live Event
SANS Paris November 2017	Paris, FR	Nov 13, 2017 - Nov 18, 2017	Live Event
SANS Adelaide 2017	OnlineAU	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced