



Analyzing TweetDeck Twaffric

By Laura Chappell

Founder, Chappell Seminar Series (www.chappellseminars.com)

Founder, Wireshark University (www.wiresharktraining.com)

The Analysis Process

After spending time playing with traffic from a single Tweet made directly from my page at www.twitter.com earlier this week, I decided to turn my attention over to TweetDeck (v0.25.1b).

Using Wireshark I created a display filter for my own traffic by systematically identifying and removing any background traffic from my trace (see Notes). A simple way to do this would be to create a filter such as `ip.addr==192.168.0.106`, but my system has loads of other garbage running in the background (some of it from Firefox plugins such as WOT) and my virus update programs. I added a number of exclusions to my filter to get a nice, clean TweetDeck set of trace files.

I tweeted “This is a test tweet to analyze TweetDeck traffic... let’s see what a link does... <http://tinyurl.com/rdnsgp>”

The resulting trace files are

- *tweetdeck-launch-login.pcap*
- *tweetdeck-idletime.pcap*
- *tweetdeck-refresh.pcap*
- *tweetdeck-tweetwithlink-tinyurl.pcap*.

These were the trace files I referenced for this analysis report.

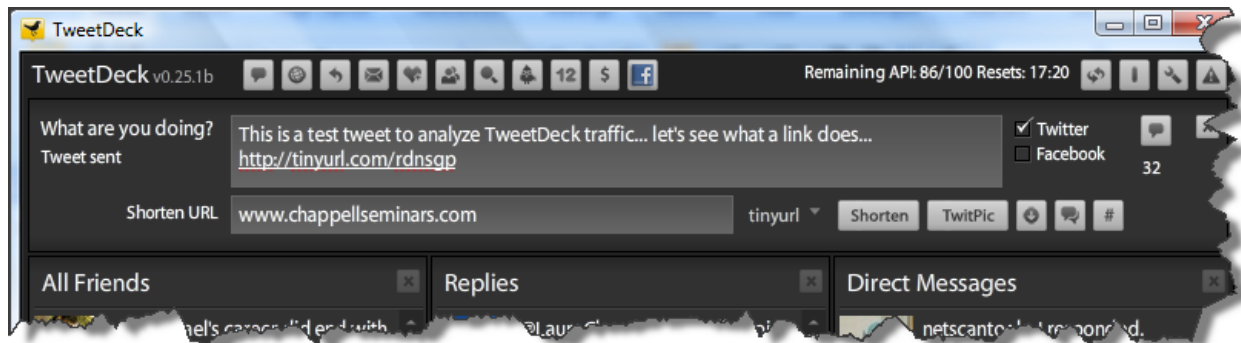


Figure 1: Preparing to send my sample Tweet with a shortened URL.

TweetDeck Launch Process – Getting to Twitter First

The initial program load process brought me to the TweetDeck login screen. This 29-packet process connected with twitter.com at 128.121.146.100. Note the User-Agent definitions in this HTTP GET request – there’s AppleWebKit and AdobeAir (I remembered the prompt to install AdobeAir when I installed TweetDeck).

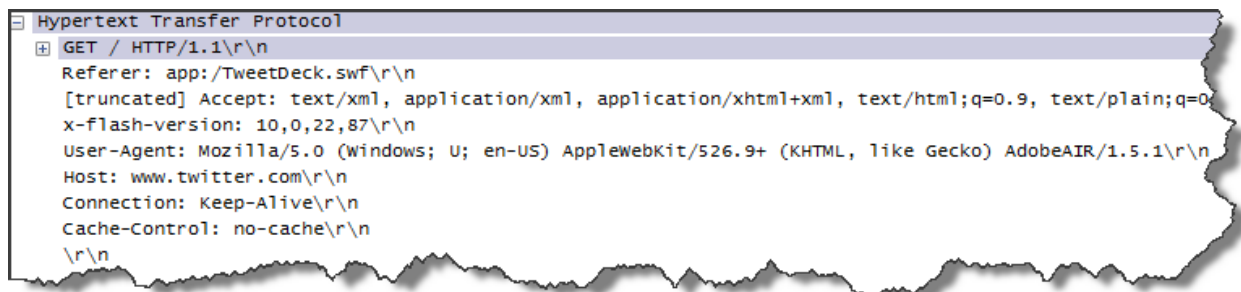


Figure 2: My first GET request listed the host as www.twitter.com.

Now here’s where Twitter seemed to get a bit snarky.

Twitter.com responded with an HTTP 301 Moved Permanently message and pointed TweetDeck to... http://twitter.com! Yes – my client dutifully sent out a FIN and then reconnected to the same exact server. This time I was not rudely told to go away and come back later. What’s up there? Well... we needed to compare the two GET requests to see where they differed. Did we ask with a smile on our face the second time?

Double-clicking on each of the GET packets and selecting Show Packet in a New Window allowed me to put the packets side-by-side to compare them. (I am using Wireshark v1.2.)

Packet 5 (the first GET) is on the left and Packet 16 (the second GET) is on the right.

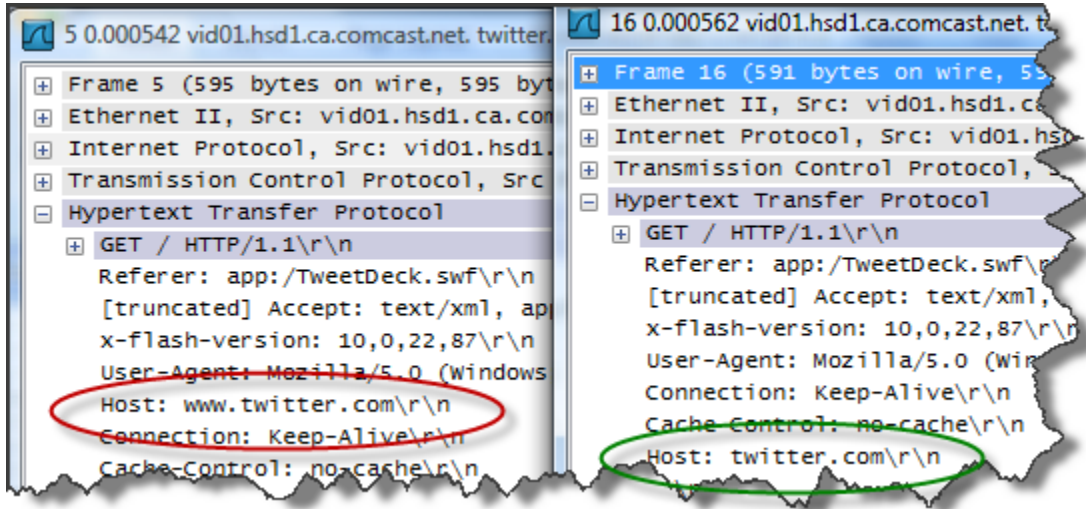


Figure 3: Comparing the two GET requests revealed a slight difference.

Oh... I see...

In the first packet we defined the host as `www.twitter.com` and in the second one we defined the host as `twitter.com`. I wish I'd not filtered out the DNS traffic now - I had to go back and re-launch it to see the DNS traffic. TweetDeck does an *A record* lookup for `www.twitter.com` and is provided the canonical name `twitter.com` with the IP address. My host didn't use the canonical name the first time around and Twitter spit at me – telling me to use `twitter.com`, not `www.twitter.com`.

How rude!

Ok, ok... I hope Twitter gets nicer to me.

It DOES!!! Look at the HTTP reply when I called the host by the right name. I guess I do understand that... I don't like it when people call me Lauren. Look at the Server : line here 'hi' – well 'hi' to you too! Check out the "Expires" value of Tue, 31 Mar 1981 05:00:00 GMT... ? Hunh? That's interesting. I tried to generate other HTTP responses from other servers to generate an expiration date in the past and couldn't find one. I already saw the no-cache, no-store, must-revalidate so this seems like overkill to me. [If you have thoughts on this, please send them to me at laura@chappellseminars.com or tweet me @laurachappell.]

```
HTTP/1.1 200 OK
Date: Fri, 26 Jun 2009 20:15:07 GMT
Server: hi
Last-Modified: Fri, 26 Jun 2009 20:15:07 GMT
Status: 200 OK
ETag: "c52e796197a32edbac048aea35b94da9"
Pragma: no-cache
Cache-Control: no-cache, no-store, must-revalidate, pre-e
Content-Type: text/html; charset=utf-8
Content-Length: 8364
Expires: Tue, 31 Mar 1981 05:00:00 GMT
X-Revision: 9771e5e1e2e60ad100caaef57dbf2c1aef8443e9
X-Transaction: 1246047307-70874-25848
Set-Cookie: param_q=; path=/; expires=Thu, 01 Jan 1970 00:
Set-Cookie: param_page=; path=/; expires=Thu, 01 Jan 1970
Set-Cookie: param_status=; path=/; expires=Thu, 01 Jan 19
Set-Cookie: param_in_reply_to_status_id=; path=/; expires=
Set-Cookie: param_in_reply_to=; path=/; expires=Thu, 01 Ja
Set-Cookie: param_source=; path=/; expires=Thu, 01 Jan 197
Set-Cookie: dispatch_action=; path=/; expires=Thu, 01 Jan
Set-Cookie: _twitter_sess=BAh7CDoMY3NyZl9pZCI1ODg5ZDEwMTM
250AM2M6B2lkIiV1NzFhMzRkZDMzOGY2MWEwXNTI5ZWVjOGZiMDZkODUy
253D%253D--7d99cce8697a66e8c6f1b15a7c050d8f9d316517; dom
Vary: Accept-Encoding
```

Figure 4: The HTTP 200 response had an interesting expiration value.

When I followed the TCP stream I found something else interesting – it seems TweetDeck gets the contents of the native Twitter page for me, but doesn't show it. There was the "what," "how," and "why" section of the Twitter page – even the three testimonials on the main Twitter page were sent to me, but TweetDeck didn't show them.

When I entered my Twitter name and password and clicked the Sign In button, an SSL session started up. Upon completion I was logged into Twitter.

Rehydration Required

Now my host made a connection to rehydrate.com. Oh... and it repeated the entire connection to Twitter (complete with the Moved Permanently process) – I guess you can never be too sure you’re really connected to Twitter as it is down so often.

When I tried to connect to rehydrate.com in a browser window (as a side note), an HTTP 301 Moved Permanently response defined that the new location was tweetdeck.com. When I connected to tweetdeck.com, I received another HTTP 301 Moved Permanently response that pointed me to www.tweetdeck.com/beta.

When working inside TweetDeck, my host connected to rehydrate.com with a GET `/beta/adobe_update_blink182.xml`. Hunh again!? Blink 182? I’m feeling old here... ‘cuz I can’t even tell you a single song of theirs off the top of my head. Some research showed the TweetDeck relationship with the band Blink 182 (and who the heck they were).

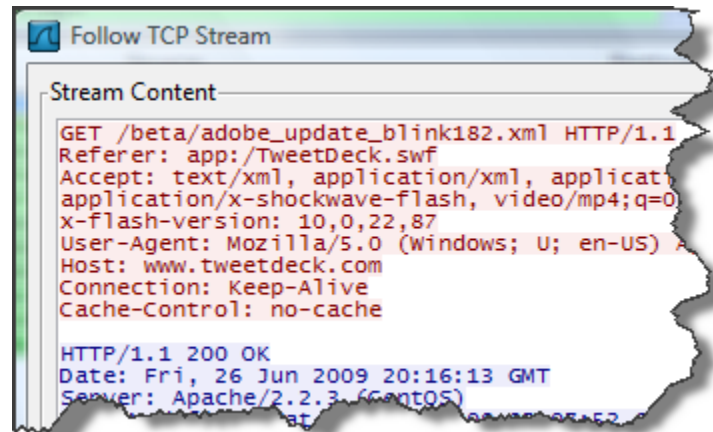


Figure 5: Blink 182? Hunh?

“Stay up to date with your favourite band, with the exclusive Blink 182 TweetDeck, and be in with a chance to win a shout-out from the band themselves.”

My host also communicated directly with www.tweetdeck.com. These packets provided me with a bit of interesting update information. Something about an integer overrun.

```
<update
xmlns="http://ns.adobe.com/air/framework/update/description/1.0">
  <version>0.25.1</version>
  <url>http://tweetdeck.com/mint/pepper/tillkruess/downloads/tracker.php
?url=http://www.tweetdeck.com/beta/TweetDeck_0_25_upgrade_Blink182.air
</url>
  <description><![CDATA[
  TweetDeck v0.25.1 includes the following updates:
  .Fix for int(eger) overrun, causing cleared tweets to reload
  ]]></description>
</update>
```

Did a quick Google on “tillkruess” and found the reference to Till Kruess and mention of Mint and Pepper. Till lives in Cologne, Germany. Never been there, but I’d like to go. It’s a small world, isn’t it?



Figure 6: There seems to be a link between this update process and a guy named Till Krues.

Not being a programmer and recognizing Mint or Pepper, I did a bit more Googling to find http://haveamint.com/peppermill/mint/all_versions/. Ok... so I get the relationship here.

There were only 11 packets in the communication to rehydrate.com – if you browse to rehydrate.com you’ll be redirected to <http://tweetdeck.com/beta/>.

More Columns = More Traffic – and it's Interesting Stuff!

In TweetDeck, I had the following columns setup:

- All Friends
- Replies
- Direct messages
- Search: #chappellseminars
- Search: wireshark
- Search: troubleshooting
- Search: sharkfest
- Search: laura chappell
- Search: netscantools
- Twitscoop

Basically, TweetDeck went to search.twitter.com to do the searches first. It also goes out to twitscoop.com to get the twitscoop info. All that makes sense... Hmm... is this analysis going to get boring now? Uh... I don't think so. Search and Twitscoop results are unencrypted.

I applied `http.request.method == "GET"` as my display filter to just see what my host asked for from the various servers.

*The file name of every Tweeter's picture showed up in clear text – “Hey, Keith – I bet you were in Nice, France with Jill when that picture was taken!!!”
Keith__Jill_in_Nice is the name of his Twitter pic right now. You can learn a lot about people by the names they give their pictures.*

It turned out that the most active GET requests were for the images of Tweepers that were displayed in TweetDeck. I could see that the images are stored on an Amazon Web Server (AWS).

Now the Fun Began...

To just get the Twitter image names, I used the display filter `http.request.uri contains "/twitter_production/profile_images/"`. It looks like every Tweeter has a unique ID assigned to them and a directory is created after the `profile_images` directory based on that user ID number.

In addition, it appeared that the original file name used in the picture upload is maintained on the server and visible in the trace file.

What fun I had! (Who cares about IO rates now, eh?)

```
GET /search.atom?rpp=100&q=%23chappellseminars HTTP/1.1
GET /beta/tdMessage.xml HTTP/1.1
GET /tweetdeck-api-23RfEz2fE.xml HTTP/1.1
GET /search.atom?rpp=100&q=wireshark HTTP/1.1
GET /search.atom?rpp=100&q=sharkfest HTTP/1.1
GET /images/default_profile_normal.png HTTP/1.1
GET /search.atom?rpp=100&q=laura%20chappell HTTP/1.1
GET /search.atom?rpp=100&q=netscantools HTTP/1.1
GET /search.atom?rpp=100&q=troubleshooting HTTP/1.1
GET /twitter_production/profile_images/279588180/lcyoung_normal.jpg HTTP/1.1
GET /twitter_production/profile_images/279588180/lcyoung_normal.jpg HTTP/1.1
GET /twitter_production/profile_images/279588180/lcyoung_normal.jpg HTTP/1.1
GET /twitter_production/profile_images/279588180/lcyoung_normal.jpg HTTP/1.1
GET /twitter_production/profile_images/279588180/lcyoung_normal.jpg HTTP/1.1
GET /twitter_production/profile_images/81747659/joe_normal.jpg HTTP/1.1
GET /twitter_production/profile_images/76325915/DrCussHead_normal.jpg HTTP/1.1
GET /twitter_production/profile_images/270091492/OStaticLogo_normal.png HTTP/1.1
GET /twitter_production/profile_images/261427518/rs_normal.png HTTP/1.1
GET /twitter_production/profile_images/217330425/p-7eefb2e4a9fd11dd82a4003048678d04-large-
GET /twitter_production/profile_images/268428955/stormtrooper_armor_normal.jpg HTTP/1.1
GET /twitter_production/profile_images/148901777/pic3_normal.JPG HTTP/1.1
GET /twitter_production/profile_images/279588180/lcyoung_normal.jpg HTTP/1.1
GET /twitter_production/profile_images/55297518/dennymiu_normal.jpg HTTP/1.1
GET /twitter_production/profile_images/227817931/bat_normal.jpg HTTP/1.1
GET /twitter_production/profile_images/228525499/NoBull_wo_normal.jpg HTTP/1.1
GET /twitter_production/profile_images/55297518/dennymiu_normal.jpg HTTP/1.1
GET /twitter_production/profile_images/73847069/XChange08_026-1_normal.jpg HTTP/1.1
GET /twitter_production/profile_images/227594828/rnutter_normal.jpg HTTP/1.1
GET /twitter_production/profile_images/269855366/green_8730_Foto_24_normal.jpg HTTP/1.1
GET /twitter_production/profile_images/121277118/trace-small-logo_normal.gif HTTP/1.1
GET /twitter_production/profile_images/276422939/Keith___Jill_in_Nice_normal.jpg HTTP/1.1
GET /twitter_production/profile_images/254283533/yoda_twit_normal.jpg HTTP/1.1
GET /twitter_production/profile_images/276422939/Keith___Jill_in_Nice_normal.jpg HTTP/1.1
GET /twitter_production/profile_images/190427585/nstproicon48x48_normal.png HTTP/1.1
GET /twitter_production/profile_images/190427585/nstproicon48x48_normal.png HTTP/1.1
GET /twitter_production/profile_images/190427585/nstproicon48x48_normal.png HTTP/1.1
GET /twitter_production/profile_images/190427585/nstproicon48x48_normal.png HTTP/1.1
GET /twitter_production/profile_images/218632256/facelessMan_normal.jpg HTTP/1.1
GET /twitter_production/profile_images/244668448/yee-haw-79M_normal.jpg HTTP/1.1
GET /twitter_production/profile_images/244668448/yee-haw-79M_normal.jpg HTTP/1.1
GET /twitter_production/profile_images/51590831/blurryicon_normal.jpg HTTP/1.1
GET /twitter_production/profile_images/80285375/marketnet_normal.jpg HTTP/1.1
GET /twitter_production/profile_images/277524189/IMG_0701-cropped_normal.jpg HTTP/1.1
GET /twitter_production/profile_images/68358282/andrewav_normal.jpg HTTP/1.1
GET /twitter_production/profile_images/275262904/straight_jacket2_normal.jpg HTTP/1.1
GET /twitter_production/profile_images/232594263/100_4241_edited-1_normal.jpg HTTP/1.1
GET /twitter_production/profile_images/106106300/Kevins_head_normal.jpg HTTP/1.1
```

Figure 7: Now the fun began!

Tip #1: Rename Images before Uploading to Your Twitter Profile!

I ran Wireshark with my profile_images filter while I went to dinner. When I came back I had enough fodder to last me the weekend! Some of the Tweeter's names were:

- FreeYourMindAmsterdam09 (and your body, right?)
- WhatSheWants (or so you think)
- MikeSangel (loads of first/last names – changed this to protect Mike)
- JoeClassy (I beg to differ)
- MeNoWife (uh... have you told **her** that yet? You tweet about her kindly.)
- Green_me_bitch (I'm glad you said that, not me)
- Me_looking_at_Tre_Jonas_Wedding (you know the Jonas Brothers?!))
- Madi_and_me (you told me you never dated her)
- Reno_052108 (hmmm... you told me you were visiting your mother!)
- Th_blowupdebbie (uh... she was cut out of the picture!)
- Spoon_too_big (I'm not even going to look at this picture)
- Jessica_twitter (ok... this is only fun if you read the next paragraph)

I had some fun with one Tweeter... “Hey – you look like a girl who dated my boyfriend – is your name Jessica?” He he... first I made sure the Tweeter hadn't ever said her name in her profile or previous tweets. Talk about a mind-blowing experience. I'm not sure she believes me when I said I got the name of her picture she used on Twitter. She seemed quite certain I had hacked into her computer and had access to her entire picture directory – imagine the horror! She hasn't tweeted since... hmmm... sorry.

*I have to add this... one wise Tweeter said
“Craigslislist is the worst place to find a boyfriend, my fellow Tweepers.”
Really? Worse than the flea market? You're too picky! <g>*

Tip #2: Don't Search for Anything You Don't Want Seen!

Your tweets are encrypted with SSL. Each tweet creates a new SSL connection with the twitter.com server (of course then who knows what happens with them, eh?).

Your search tweets are not encrypted. This is true when you are tweeting from the www.twitter.com site as well. Neither are your original URLs before you shorten them.

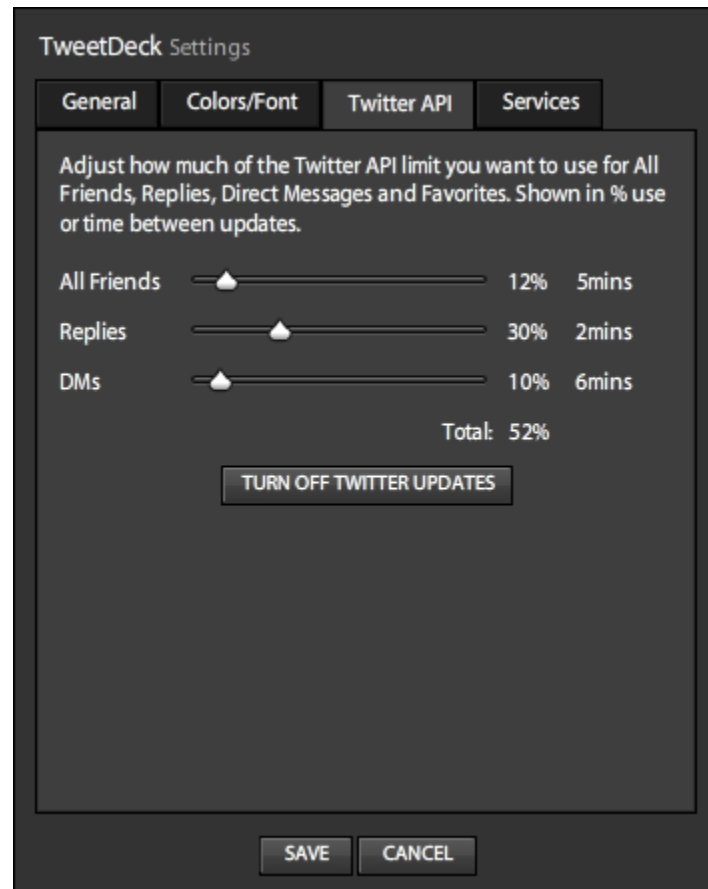
So before you search for “leather chaps”, think a moment. Could anyone be listening? Do you have some encryption scheme wrapping around your traffic? Or are you at that open AP that they provide for you at the coffee shop?

In fact... it's time to end this analysis report so I can hop down to the local coffee shop. It's filled with Twit-heads and I'd like to see what's on their minds... one last thing – traffic level overview.

TweetDeck Traffic Level

The traffic level increases for each column you have and the number of entries you want to retain in each column.

TweetDeck refreshes based on your TweetDeck settings for the Twitter API. Searches seem to be stuck at a 60-second interval. Fewer searches = less traffic. Even with my set of searches, my IO rate rarely popped over 250,000 bps though. My average packet rate was 141 packets per minute.



Overview of Findings

Twitter traffic is a relatively minor annoyance on the network – like ants crossing the playground. Tweeters who use twitter.com typically spend time on their own page OR another Tweeter’s page OR searching. TweetDeck, however, encourages opening many simultaneous views or columns at a time. Sure – Tweeters using twitter.com could open more than one browser window to do the same thing, but it’s clunky and requires moving from window to window to see the various pages.

TweetDeck has some funky behavior issues in its connection process with Twitter and I’d be a tad concerned about the open nature of all those searches – sure, you can do the same searches from Twitter.com, but TweetDeck lends itself to running multiple searches automatically in the background all day long.

Blocking Twitter Traffic

To block Twitter traffic following this standard communication process, an IT administrator could block the DNS queries for www.twitter.com or twitter.com or block access to the Twitter address block (128.121.144.0-128.121.159.255). There are numerous signatures inside the packets as well (the “twitter_production” for example).

Final Notes

I started this analysis project with some basic assumptions on how TweetDeck probably worked – yes, I knew it had to link into twitter.com. There were many fun surprises along the way though – the reference to Blink 192, Till Krueger and Mint, the Tweet picture names and the unencrypted Search results. I’d half expected the actual Tweets (direct and general replies)

Filter Notes

I created a filter for all traffic to/from my IP address (ip.addr==192.168.0.106) and then filtered out any of my unrelated traffic. I was working backwards and separating out my Firefox traffic and any other noise that affects my host. I created a number ‘exclusions’ to my display filter as I identified my background traffic to my printer, my router’s management port, DHCP noise, ARP noise, traffic from my iPhone (which was being bridged onto the wired network), Google Analytics and Google Malware updates from Firefox, World News and BBC background feeds from Firefox and anything else not related to my TweetDeck communications. When my convoluted display filter was completed, I could see no background traffic from superfluous processes.

```
ip.addr==192.168.0.106 && !srvloc && !dns && !ip.addr==74.6.114.56 && !ip.addr==239.255.255.250 &&  
!ip.addr==96.17.0.0/16 && !ip.addr==192.168.0.102 && !smb && !nbns && !ip.addr== 192.168.0.103 &&  
!ip.addr==64.74.80.187 && ! ip.addr==83.150.67.33 && !ip.addr==67.217.0.0/16 && !ip.addr==66.102.7.101 &&  
!ip.addr==216.115.0.0/16 && !ip.addr==216.219.0.0/16 && !ip.addr==69.90.30.72
```

Who the Heck are We?

Good question. Why do you want to know?

Well... I'm a network analyst – been perusing packets for almost 20 years now. I still find it fascinating. In 2007, I founded Wireshark University after talking with Gerald Combs, the creator of Wireshark, about the need for training to support Wireshark. This year I launched Chappell University to broaden the training topics past Wireshark to the other tools I use every day. I also released the Chappell Seminar Series, enabling me to teach online every week on a variety of topics dealing with networking technology, troubleshooting, optimization and security.

I'm also a Mom of two great kids who inherited my twisted sense of humor (and hopefully not my tremendously disappointing cooking skills). They make me laugh every day. I am lucky.

Join me for one of the free Wireshark Jumpstart online courses – the schedule is at chappellseminars.com. You can also access over 200 video-based course modules online at chappellU.com and the Wireshark University course/certification information at wiresharkU.com.

Comments? Questions? More cool information about Twitter/TweetDeck twaffic? Send an email to laura@chappellseminars.com.