[I didn't have presentation slides. This is a from-memory transcription of my talk.]

Hello, my name is Steve, I'm the quartermaster of the Massachusetts Pirate Party, and this is the Cryptoparty segment of Software Freedom day. Cryptoparties are a way to introduce people to encryption and privacy-enhancing technologies. Kendra will demonstrate Jitsi and Noe will demonstrate Mailvelope and PGP. I'm here to give an introduction, and to set some context for why these things are important.

In the next couple of minutes, I'm going to demonstrate of packet sniffing. Packet sniffing sounds like something you did at the back of the bus in high school, but it's really a form of traffic capture and network surveillance. I'm going to use tshark -- a text-only version of wireshark -- to capture traffic from my web browser, and dump it to a file. Then, we'll poke around to see what we can find.

I like wireshark because it understands protocols. Protocols are basically rules of engagement, and they've been around much longer than computers. For example, green means go and red means stop. That's a protocol. Your cat walks over to her food dish and starts yowling, and you feed the cat. That's a protocol. For us, protocols are the rules for how computers talk to each other over the network.

Here's an example of the SMTP protocol, which is used to deliver email.

```
220 buffy.mayfirst.org ESMTP Postfix (Debian/GNU)
HELO buffy.mayfirst.org
250 buffy.mayfirst.org
MAIL FROM: <steve@....>
250 2.1.0 Ok
RCPT TO: <steve@....>
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
From: info@...
To: steve@...
Subject: im in yr mailz

kthxbye
.
250 2.0.0 Ok: queued as 90913FB13
QUIT
221 2.0.0 Bye
Connection closed by foreign host.
```

Here, I've just sent myself an email message. That's exactly how your mail program works.

Another common protocol, and the one we're going to focus on, is HTTP. Here's how your web browser asks for a web page:

```
$ telnet fsf.org 80
Trying 208.118.235.131...
Connected to fsf.org.
Escape character is '^]'.
GET / HTTP/1.0
Host: fsf.org

HTTP/1.0 301 Moved Permanently
Server: nginx/1.1.19
```

```
  Date: Sun, 21 Sep 2014 15:01:16 GMT
  Content-Type: text/html
  Content-Length: 185
  Location: http://www.fsf.org/
  X-Cache: MISS from www.fsforg
  X-Cache-Lookup: MISS from www.fsforg:80
  Via: 1.0 www.fsforg (squid/3.1.19)
  Connection: close

  <html>
  <head><title>301 Moved Permanently</title></head>
  <body bgcolor="white">
  <center><h1>301 Moved Permanently</h1></center>
  <hr><center>nginx/1.1.19</center>
  </body>
  </html>
```

In this case, we got a redirect from fsf.org, to www.fsf.org.

This is what network traffic looks like. In these examples, I'm directly generating the network traffic. Now, I'll use tshark to capture network traffic generated by another program.

Let's start tshark, and capture some packets

```
  sudo tshark -O http -i wlan0 -R "http.request || http.response" | tee pcap.txt
```

This is going to capture all of the HTTP request and response packets going through my laptop's wireless card. This won't capture page content (because it's big and messy to wade through) -- it's just metadata. But of course, I could capture the content if I wanted to.

Before doing this, I need to make a few changes to my web browser. Normally, I try to keep my browser locked down; I'll undo that for this demonstration, so that we have more interesting things to look at.

The first thing I'll do is disable a couple of extensions. Specifically, I'm going to disable NoScript (which selectively blocks javascript); I'm going to disable RefControl (which blocks http referer headers), and I'm going to disable HTTPS Everywhere. HTTPS Everywhere tries to force traffic over HTTPS, but our packet capture won't be very interesting if the traffic is encrypted. Finally, I'm going to go into my browser's cookie preferences and change "Accept Third-Party cookies" from "Never" to "Always".

Now, let's ask for the front page of the washington post.

```
  http://www.washingtonpost.com/
```

Now that Washington Post's front page has finished loading, let's look at our packet capture:

Here's the first request

```
  Frame 45: 379 bytes on wire (3032 bits), 379 bytes captured (3032 bits) on interface 0
  Ethernet II, Src: HonHaiPr_ec:9f:9e (00:26:5e:ec:9f:9e), Dst: 34:4d:f7:38:84:c3 (34:4d:
f7:38:84:c3)
  Internet Protocol Version 4, Src: 192.168.43.52 (192.168.43.52), Dst: 192.33.31.56 (192
.33.31.56)
  Transmission Control Protocol, Src Port: 42382 (42382), Dst Port: http (80), Seq: 1, Ac
k: 1, Len: 313
  Hypertext Transfer Protocol
```

```
      GET / HTTP/1.1\r\n
          [Expert Info (Chat/Sequence): GET / HTTP/1.1\r\n]
              [Message: GET / HTTP/1.1\r\n]
              [Severity level: Chat]
              [Group: Sequence]
          Request Method: GET
          Request URI: /
          Request Version: HTTP/1.1
      Host: www.washingtonpost.com\r\n
      User-Agent: Mozilla/5.0 (X11; Linux i686; rv:28.0) Gecko/20100101 Firefox/28.0 Icew
easel/28.0\r\n
      Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
      Accept-Language: en-US,en;q=0.5\r\n
      Accept-Encoding: gzip, deflate\r\n
      DNT: 1\r\n
      Connection: keep-alive\r\n
      \r\n
      [Full request URI: http://www.washingtonpost.com/]
```

What's do we see here? First, we see the source and destination MAC addresses. These are the Layer II hardware network addresses.

```
  Ethernet II, Src: HonHaiPr_ec:9f:9e (00:26:5e:ec:9f:9e), Dst: 34:4d:f7:38:84:c3 (34:4d:
f7:38:84:c3)
```

We also see the source and destination IP addresses

```
  Internet Protocol Version 4, Src: 192.168.43.52 (192.168.43.52), Dst: 192.33.31.56 (192
.33.31.56)
```

and the source and destination port numbers

```
  Transmission Control Protocol, Src Port: 42382 (42382), Dst Port: http (80), Seq: 1, Ac
k: 1, Len: 313
```

We saw a lot of stuff go whizzing by on the screen. Let's look at the different Host: headers, to get a sense of what our browser was doing. Here are the first couple of matching lines

```
   Host: www.washingtonpost.com\r\n
   Host: css.washingtonpost.com\r\n
   Host: wp-eng-static.washingtonpost.com\r\n
   Host: www.washingtonpost.com\r\n
   Host: media.washingtonpost.com\r\n
   Host: img.washingtonpost.com\r\n
   Host: img.washingtonpost.com\r\n
   Host: img.washingtonpost.com\r\n
   Host: img.washingtonpost.com\r\n
   Host: img.washingtonpost.com\r\n
   Host: img.washingtonpost.com\r\n
   Host: www.washingtonpost.com\r\n
   Host: www.washingtonpost.com\r\n
   Host: www.washingtonpost.com\r\n
   Host: www.washingtonpost.com\r\n
   Host: js.washingtonpost.com\r\n
   Host: js.washingtonpost.com\r\n
   Host: js.washingtonpost.com\r\n
```

Okay, there's www.washingtonpost.com, css.washingtpost.com, and

```
js.washingtonpost.com; those seem related to our initial request.
What other Host: headers can we find?

    $ grep Host: pcap.txt | sort -u
    Host: a.tile.openstreetmap.org\r\n
    Host: aax.amazon-adsystem.com\r\n
    Host: ad.360yield.com\r\n
    Host: ad.doubleclick.net\r\n
    Host: adadvisor.net\r\n
    Host: admaym.com\r\n
    Host: ads.adsonar.com\r\n
    Host: ads.yahoo.com\r\n
    Host: ak1.abmr.net\r\n
    Host: api.bizographics.com\r\n
    Host: apiservices.krxd.net\r\n
    Host: b.scorecardresearch.com\r\n
    Host: b.tile.openstreetmap.org\r\n
    Host: beacon-3.newrelic.com\r\n
    Host: beacon.krxd.net\r\n
    Host: bh.contextweb.com\r\n
    Host: bs.serving-sys.com\r\n
    Host: c.amazon-adsystem.com\r\n
    Host: c.tile.openstreetmap.org\r\n
    Host: cdn.doubleverify.com\r\n
    Host: cdn.krxd.net\r\n
    Host: cdn.tacoda.at.atwola.com\r\n
    Host: cdn.tt.omtrdc.net\r\n
    Host: ce.lijit.com\r\n
    Host: cm.g.doubleclick.net\r\n
    Host: cotads.adscale.de\r\n
    Host: cs.specificclick.net\r\n
    Host: css.washingtonpost.com\r\n
    Host: d.agkn.com\r\n
    Host: d1ros97qkrwjf5.cloudfront.net\r\n
    Host: delivery.swid.switchads.com\r\n
    Host: dis.criteo.com\r\n
    Host: dizqy8916g7hx.cloudfront.net\r\n
    Host: ds.serving-sys.com\r\n
    Host: dt.adsafeprotected.com\r\n
    Host: e.nexac.com\r\n
    Host: event.trove.com\r\n
    Host: ib.adnxs.com\r\n
    Host: ib.mookie1.com\r\n
    Host: id.washingtonpost.com\r\n
    Host: idsync.rlcdn.com\r\n
    Host: idvisitor.socialreader.com\r\n
    Host: ih.adscale.de\r\n
    Host: image2.pubmatic.com\r\n
    Host: img.washingtonpost.com\r\n
    Host: js.adsonar.com\r\n
    Host: js.moatads.com\r\n
    Host: js.revsci.net\r\n
    Host: js.washingtonpost.com\r\n
    Host: load.s3.amazonaws.com\r\n
    Host: loadm.exelator.com\r\n
    Host: media.washingtonpost.com\r\n
    Host: meraxes-cdn.polarmobile.com\r\n
    Host: metrics.washingtonpost.com\r\n
    Host: moneta.trove.com\r\n
    Host: o.aolcdn.com\r\n
    Host: openstreetmaps.org\r\n
    Host: p.acxiom-online.com\r\n
    Host: partner.googleadservices.com\r\n
    Host: ping.chartbeat.net\r\n
```

```
    Host: piwik.openstreetmap.org\r\n
    Host: pix.btrll.com\r\n
    Host: pixel.adsafeprotected.com\r\n
    Host: pixel.mathtag.com\r\n
    Host: pixel.rubiconproject.com\r\n
    Host: plugin.mediavoice.com\r\n
    Host: pubads.g.doubleclick.net\r\n
    Host: pwapi.washingtonpost.com\r\n
    Host: r.casalemedia.com\r\n
    Host: r.nexac.com\r\n
    Host: rtax.criteo.com\r\n
    Host: rtb-csync.smartadserver.com\r\n
    Host: rumds.wpdigital.net\r\n
    Host: s.ixiaa.com\r\n
    Host: s.kau.li\r\n
    Host: s.troveread.com\r\n
    Host: s1.2mdn.net\r\n
    Host: search.spotxchange.com\r\n
    Host: static.adsafeprotected.com\r\n
    Host: static.chartbeat.com\r\n
    Host: stats.opbandit.com\r\n
    Host: su.addthis.com\r\n
    Host: sync.adap.tv\r\n
    Host: sync.go.sonobi.com\r\n
    Host: sync.mathtag.com\r\n
    Host: sync.search.spotxchange.com\r\n
    Host: sync.tidaltv.com\r\n
    Host: sync.zenoviaexchange.com\r\n
    Host: sync2.dist.us-east.zenoviaexchange.com\r\n
    Host: t.mookie1.com\r\n
    Host: tags.bluekai.com\r\n
    Host: tpc.googlesyndication.com\r\n
    Host: tps10225.doubleverify.com\r\n
    Host: tps30.doubleverify.com\r\n
    Host: ums.adtechus.com\r\n
    Host: us-u.openx.net\r\n
    Host: valyria-cdn.polarmobile.com\r\n
    Host: washpost.bloomberg.com\r\n
    Host: widgets.outbrain.com\r\n
    Host: wp-eng-static.washingtonpost.com\r\n
    Host: wpni.tt.omtrdc.net\r\n
    Host: www.adadvisor.net\r\n
    Host: www.burstnet.com\r\n
    Host: www.openstreetmap.org\r\n
    Host: www.washingtonpost.com\r\n
    Host: y.one.impact-ad.jp\r\n
```

Wow, that’s a lot. Many of them are advertisers, or what I refer to as the "corporate surveillance industry". Some of these websites are going to plant cookies with unique identifiers, allowing them to track you from site to site. Of course, if someone with a big data center in Utah captures these identifiers, they can also use them to track you from site to site.

Cookies can be interesting to look at. Let’s examine a few

```
    Host: pixel.mathtag.com\r\n
    Referer: http://www.washingtonpost.com/\r\n
    DNT: 1\r\n
    [truncated] Set-Cookie: mt_mop=10025:1410015465|41:1410015465|10042:1410015465|37:141
0015465|35:1410015465|10008:1410015465|39:1410015465|10031:1410015465|10018:1410015465|3:
1410015465|5:1410015465|4:1410015465|10013:1410015465|9:141001546
```

This cookie is from a mathtag tracking pixel. I'm sure these numbers mean something to someone. So much for my Do Not Track headers.

I think the most interesting cookie is one that washingtonpost.com sets directly:

```
    Cookie: de=; client_region=0; X-WP-Split=A; devicetype=0; rpld1=0:myvzw.com|20:usa|21
:ma|22:boston|23:42.358002|24:-71.056999|\r\n
```

This is a geocoding cookie. I'm betting that myvzw.com comes from Verizon wireless (thanks to the nice fellow who's letting me use his phone as a wireless hotspot). We also see USA, MA, and Boston. The last two numbers look like latitude and longitude. Let's plug them into openstreetmap and see where they are.

```
  http://www.openstreetmap.org/search?query=42.358002%2C-71.056999#map=17/42.35800/-71.05
700
```

The location in downtown Boston, just off Congress street. Presumably, this is where Verizon Wireless's backhaul terminates.

Last night, I tried this exercise at home. The geolocation was about six blocks from my house.

What should you take away from this? Even without looking at content, there's a lot of information in our packet capture. If you can collect network traffic wholesale, then it's not hard to store, catalog, and query. If you're traffic is encrypted, it can still be captured and stored. But it will take a significant amount of work to get any information out.