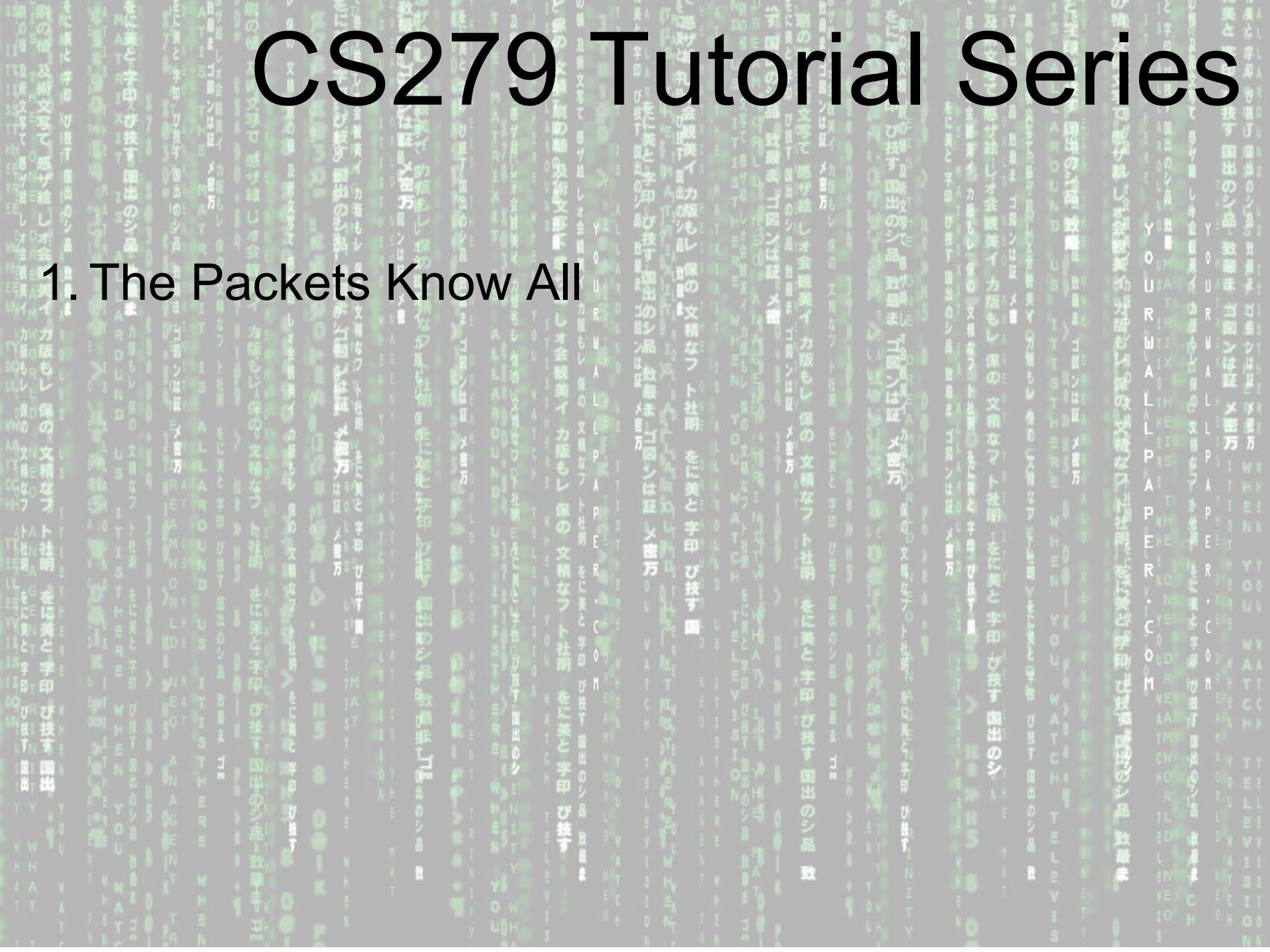


# CS279 Tutorial Series

## 1. The Packets Know All

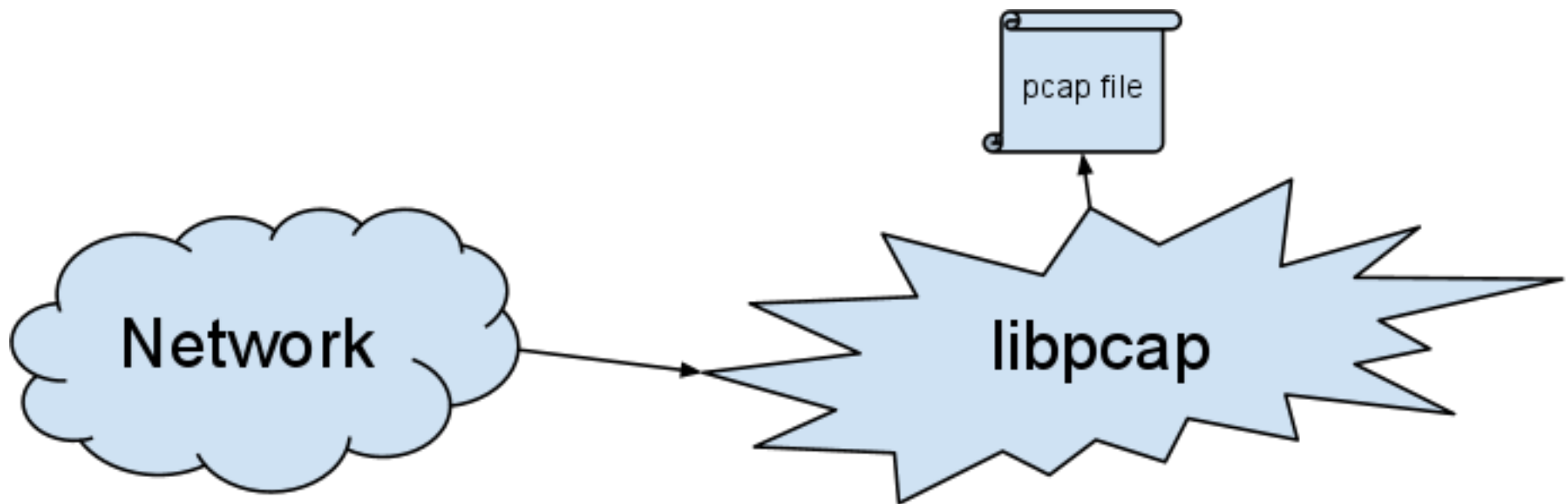


# ... tools of the trade

- tcpdump
  - passively capture network traffic
- tcpslice
  - slice apart and stitch together traffic dumps according to time
- tcpflow
  - separate network traffic into flows
- tcpreplay
  - replay traffic saved with tcpdump
- ettercap
  - does arp spoofing to sniff and do MITM attacks on switched networks
- countless others!
  - dsniff, driftnet, wireshark, 4g8, and lots more

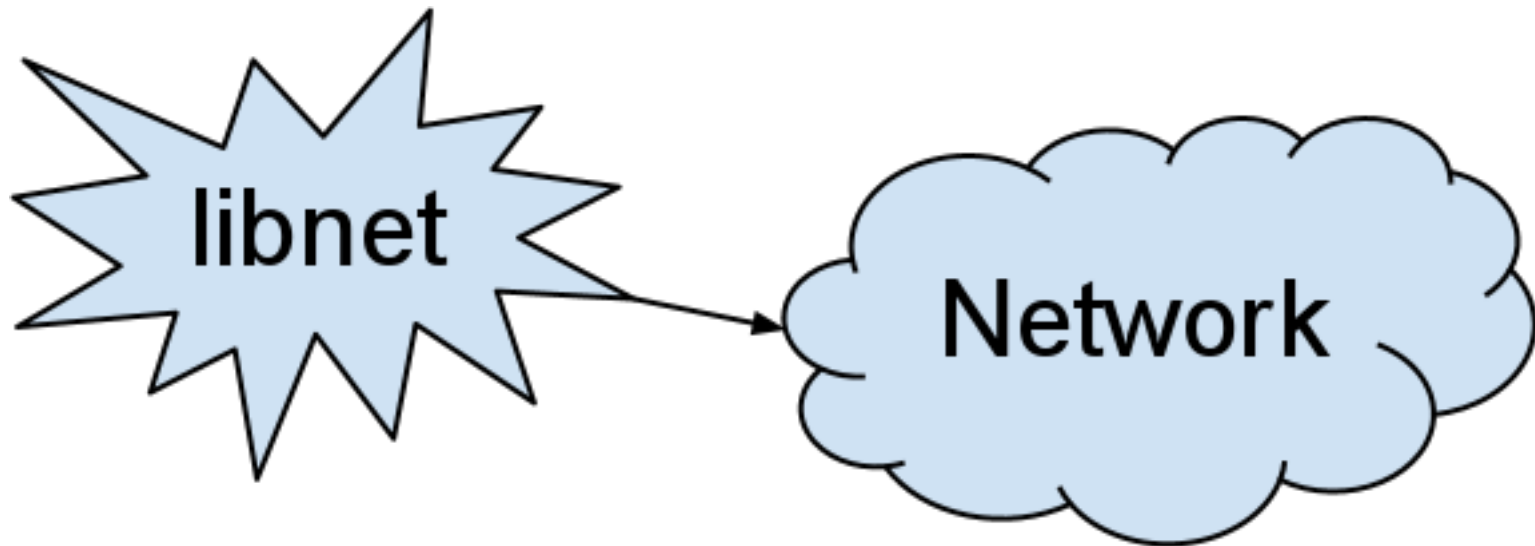
# ... one library to rule them all

- libpcap provides a cross-platform, very powerful framework for implementing network sniffing software
- most programs based on libpcap have the capability to save captured network traffic in a pcap file
- likewise, most of them can use a pcap file for input instead of live network traffic
- this lets us record traffic with tcpdump and analyze it offline later.



# ... and the other one!

- libnet provides a cross-platform, very powerful, and insanely undocumented framework for generating low-level network traffic
- this is what, for example, ettercap and tcpreplay use to send traffic



# introducing: tcpdump

- tcpdump is a general-purpose packet capturing tool
- can save packets for later analysis
- powerful filtering capability
- Very configurable. Useful options:
  - choose interface: -i interface
    - can use "any" in Linux to capture on all interfaces
  - don't use DNS to resolve addresses (much quicker): -n
  - dump packet contents as well as packet headers: -X
  - capture only number packets: -c number
  - write packets to file: -w file
  - print absolute sequence numbers: -S
  - print the whole packet (along with -X): -s 0

# tcpdump filters

- Example tcpdump filters:
  - Get only HTTP requests to google:
    - `dst host google.com and port 80`
  - Get all TCP traffic except for ssh:
    - `tcp and not port 22`
  - Get call link-local ipv6 traffic:
    - `ip6 and dst net fe80::/64`
  - What the heck?
    - `tcp port 80 and (((ip[2:2] - ((ip[0]&0xf)<<2)) - ((tcp[12] &0xf0)>>2)) != 0)`
    - `icmp[icmptype] != icmp-echo and icmp[icmptype] != icmp-echoreply`

# demo: tcpdump

- examine tcpdump output
- sniff a telnet session
- sniff an HTTP GET request
- save a telnet session to pcap files for later



# introducing: tcpslice

- can split and merge pcap files by timestamp
- useful when you want to make sure you have all the captured data in one pcap
- scenario:
  - you have compromised several dozen core backbone routers and are sniffing traffic
  - however, packets of a given connection might take different routes, so you might see them on different routers
  - in order to analyze a flow, you need to splice the different packets together
- there might also be some valid reason to use tcpslice to slice files up as well, but I've never really needed to do that...

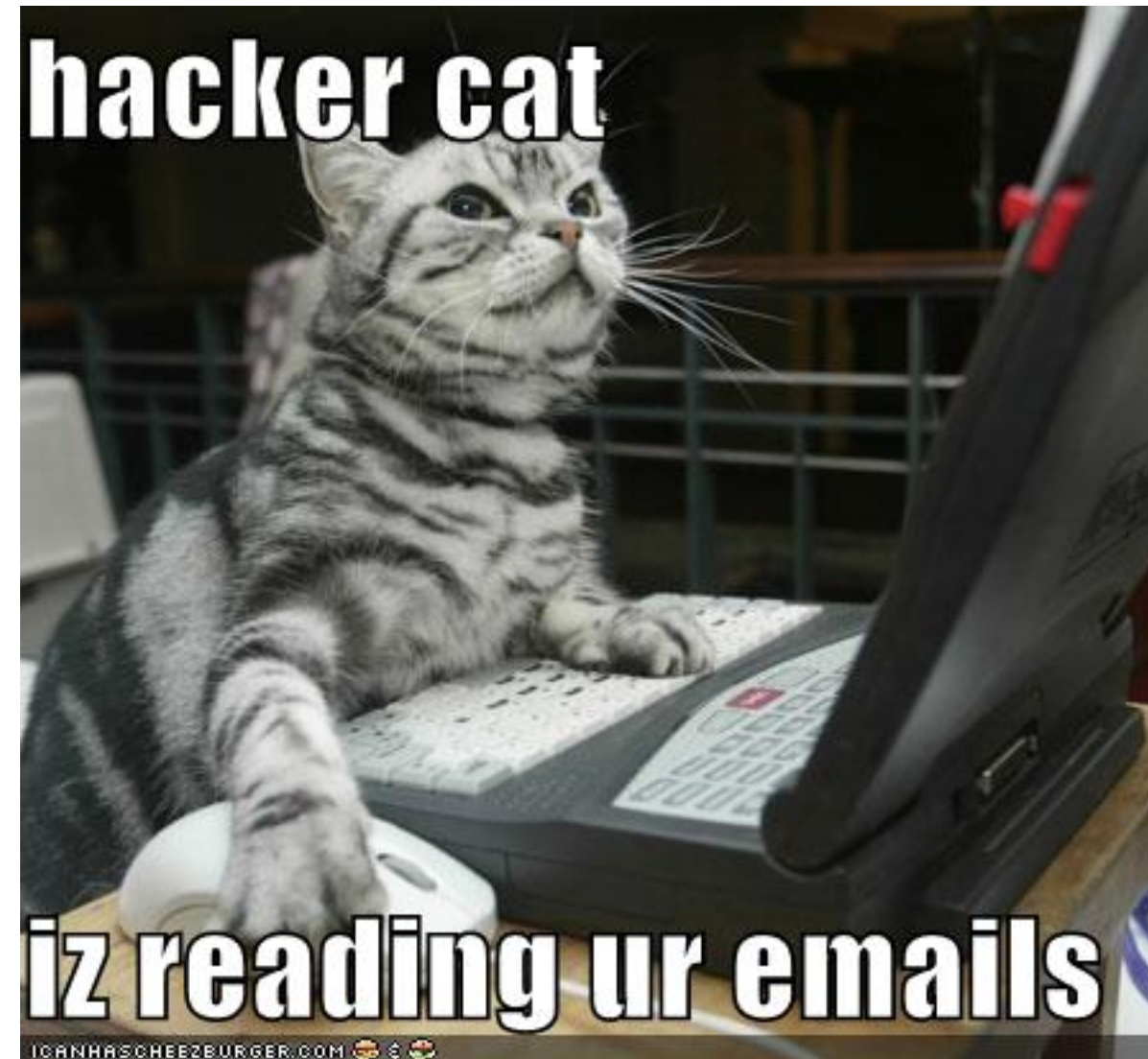
# demo: tcpslice

- merge the pcap files from the tcpdump dump
- special guest: dsniff!

# introducing: tcpflow

- oftentimes, the lower-level network information is not important, and it is best to focus on application-level data
  - protocol analysis
  - forensics
- tcpflow separates out the different flows (connections) in a network capture and outputs each to its own file
- provides libpcap-style filters to further narrow down network content

# demo: tcpflow



- examine flows in the pcap file generated by tcpslice

# introducing: tcpreplay

- provides a capability to modify and replay captured network traffic
- complicated approach:
  - tcpprep
    - pre-processes the pcap file, identifying clients and servers
  - tcprewrite
    - mangles pcap file to modify IPs, ports, etc
  - tcpreplay
    - replays mangled pcap file
- **HOWEVER:** tcpreplay is unable to update TCP sequence numbers, so it's not all that interesting for us as it can't bounce captured traffic against a server.

# introducing: ettercap

- □ everything so far works great if you're the router or you're on a hub, but if you're on a switched network, sniffing isn't quite so easy
- switches keep track of which MAC addresses belong on which ports, so the attacker would not see traffic belonging to the victim
- solution: ARP poison the victim(s) and the gateway so that they send all their packets to us! Noisy, but effective.
- and, while we have all that traffic going to us anyways, why not do some sneaky MITM stuff?
- if the goal is just to capture/redirect traffic from one host, 4g8 is a lighter tool
- note that once the ARP poisoning is in place, any of the standard tcp\* suite of tools can be used again!

# demo: ettercap

- sneaky ARP-driven evil!
- command injection



# introducing: wireshark

- wireshark is a powerful graphical network traffic analyzer
  - tcpdump and tcpflow and some extra goodness rolled into one
- also works on top of libpcap, so it can read and write pcap files like everyone else
- exposes a protocol dissector API for known protocols, which can be quite useful
- provides another, quite powerful filtering capability on top of tcpdump's

# demo: wireshark

- open our spliced pcap file
- look at some live-capture



# Questions?

