INTRO TO ETHICAL HACKING

MIS 5211.701

Week 7

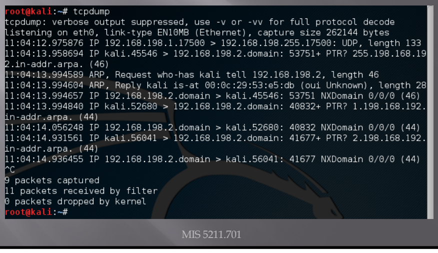
http://community.mis.temple.edu/mis5211sec701fall2018/

Tonight's Plan

- Tcpdump
- Windump
- Just a little bit of Wireshark
- Network Taps

tcpdump

- Tcpdump is a network analysis tool
- Requires root or sudo privileges
- Displays network traffic in a raw state



```
root@kali:~# tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
11:04:12.975876 IP 192.168.198.1.17500 > 192.168.198.255.17500: UDP, length 133
11:04:13.958694 IP kali.45546 > 192.168.198.2.domain: 53751+ PTR? 255.198.168.19
2.in-addr.arpa. (46)
11:04:13.994589 ARP, Request who-has kali tell 192.168.198.2, length 46
11:04:13.994604 ARP, Reply kali is-at 00:0c:29:53:e5:db (oui Unknown), length 28
11:04:13.994657 IP 192.168.198.2.domain > kali.45546: 53751 NXDomain 0/0/0 (46)
11:04:13.994840 IP kali.52680 > 192.168.198.2.domain: 40832+ PTR? 1.198.168.192.
in-addr.arpa. (44)
11:04:14.056248 IP 192.168.198.2.domain > kali.52680: 40832 NXDomain 0/0/0 (44)
11:04:14.931561 IP kali.56041 > 192.168.198.2.domain: 41677+ PTR? 2.198.168.192.
in-addr.arpa. (44)
11:04:14.936455 IP 192.168.198.2.domain > kali.56041: 41677 NXDomain 0/0/0 (44)
^C
9 packets captured
11 packets received by filter
0 packets dropped by kernel
root@kali:~#
```

Windows and Mac

- On Windows there is an equivalent called windump
  - Available at: https://www.winpcap.org/windump/
  - WinDump captures using the WinPcap library and drivers, which are freely downloadable from the WinPcap.org website.

Note: Installing windows version of Wireshark will add the WinPcap files needed by WinDump

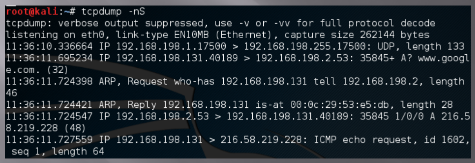
- For Mac tcpdump is built in
  - Apple provides some direction on use at: https://support.apple.com/en-us/HT202013

Basic Use

- Some basic flags (See man page for more)
  - -c Count function, how many packets to you want. If you don't say it will just keep running until you hit CTRL-C
  - -n Don't resolve addresses to names
  - -nn Don't resolve address or port names
  - -s Snap Length, how much of the packet do you want
  - -S Absolute sequence number
  - -v, -vv, and –vvv Varying degrees of verbose. How much do you want tcpdump to tell you
  - -X Data from each packet

First example

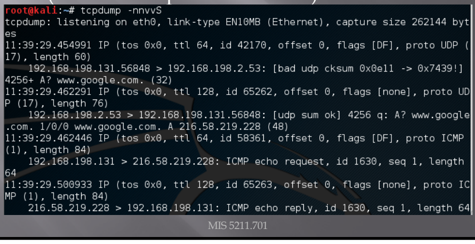
- Tried tcpdump –nS
  - -n (Don't convert addresses)
  - -S (Absolute sequence numbers)



```
root@kali:~# tcpdump -nS
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
11:36:10.336664 IP 192.168.198.1.17500 > 192.168.198.255.17500: UDP, length 133
11:36:11.695234 IP 192.168.198.131.40189 > 192.168.198.2.53: 35845+ A? www.googl
e.com. (32)
11:36:11.724398 ARP, Request who-has 192.168.198.131 tell 192.168.198.2, length
46
11:36:11.724421 ARP, Reply 192.168.198.131 is-at 00:0c:29:53:e5:db, length 28
11:36:11.724547 IP 192.168.198.2.53 > 192.168.198.131.40189: 35845 1/0/0 A 216.5
8.219.228 (48)
11:36:11.727559 IP 192.168.198.131 > 216.58.219.228: ICMP echo request, id 1602,
 seq 1, length 64
```

Next Example

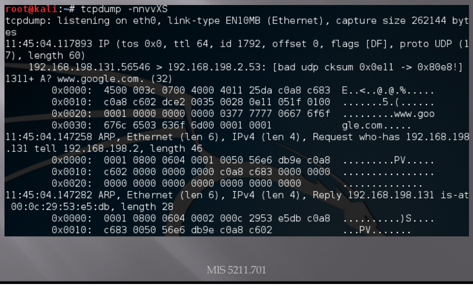
- Tried tscpdump –nnvvS
  - -nn (Don't resolve address or port to names)
  - -vv (Tell me more)
  - -S (Absolute sequence numbers)



```
root@kali:~# tcpdump -nnvvS
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 byt
es
11:39:29.454991 IP (tos 0x0, ttl 64, id 42170, offset 0, flags [DF], proto UDP (
17), length 60)
    192.168.198.131.56848 > 192.168.198.2.53: [bad udp cksum 0x0e11 -> 0x7439!]
4256+ A? www.google.com. (32)
11:39:29.462291 IP (tos 0x0, ttl 128, id 65262, offset 0, flags [none], proto UD
P (17), length 76)
    192.168.198.2.53 > 192.168.198.131.56848: [udp sum ok] 4256 q: A? www.google
.com. 1/0/0 www.google.com. A 216.58.219.228 (48)
11:39:29.462446 IP (tos 0x0, ttl 64, id 58361, offset 0, flags [DF], proto ICMP
(1), length 84)
    192.168.198.131 > 216.58.219.228: ICMP echo request, id 1630, seq 1, length
64
11:39:29.500933 IP (tos 0x0, ttl 128, id 65263, offset 0, flags [none], proto IC
MP (1), length 84)
    216.58.219.228 > 192.168.198.131: ICMP echo reply, id 1630, seq 1, length 64
```

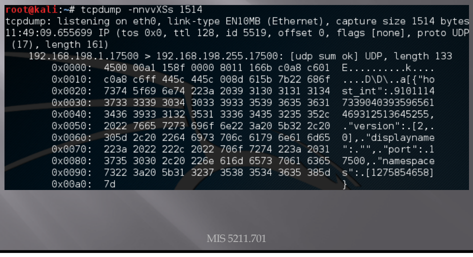
Next Example

- Try tcpdump –nnvvXS
  - Add -X (Captures data)



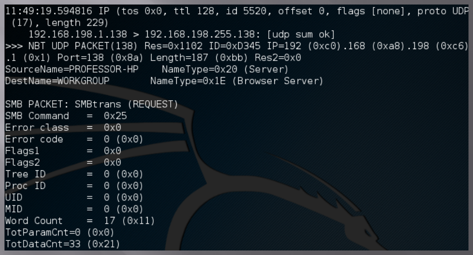
```
root@kali:~# tcpdump -nnvvXS
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 byt
es
11:45:04.117893 IP (tos 0x0, ttl 64, id 1792, offset 0, flags [DF], proto UDP (1
7), length 60)
    192.168.198.131.56546 > 192.168.198.2.53: [bad udp cksum 0x0e11 -> 0x80e8!]
1311+ A? www.google.com. (32)
        0x0000:  4500 003c 0700 4000 4011 25da c0a8 c683  E..<..@.@.%.....
        0x0010:  c0a8 c602 dce2 0035 0028 0e11 051f 0100  .......5.(......
        0x0020:  0001 0000 0000 0000 0377 7777 0667 6f6f  .........www.goo
        0x0030:  676c 6503 636f 6d00 0001 0001            gle.com.....
11:45:04.147258 ARP, Ethernet (len 6), IPv4 (len 4), Request who-has 192.168.198
.131 tell 192.168.198.2, length 46
        0x0000:  0001 0800 0604 0001 0050 56e6 db9e c0a8  .........PV.....
        0x0010:  c602 0000 0000 0000 c0a8 c683 0000 0000  ................
        0x0020:  0000 0000 0000 0000 0000 0000 0000       ..............
11:45:04.147282 ARP, Ethernet (len 6), IPv4 (len 4), Reply 192.168.198.131 is-at
 00:0c:29:53:e5:db, length 28
        0x0000:  0001 0800 0604 0002 000c 2953 e5db c0a8  ..........)S....
        0x0010:  c683 0050 56e6 db9e c0a8 c602            ...PV.......
```

- Try tcpdump –nnvvXSs 1514
  - Final –s extends defaults snap length to capture full packet



```
root@kali:~# tcpdump -nnvvXSs 1514
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 1514 bytes
11:49:09.655699 IP (tos 0x0, ttl 128, id 5519, offset 0, flags [none], proto UDP
 (17), length 161)
    192.168.198.1.17500 > 192.168.198.255.17500: [udp sum ok] UDP, length 133
        0x0000:  4500 00a1 158f 0000 8011 166b c0a8 c601  E..........k....
        0x0010:  c0a8 c6ff 445c 445c 008d 615b 7b22 686f  ....D\D\..a[{"ho
        0x0020:  7374 5f69 6e74 223a 2039 3130 3131 3134  st_int":.9101114
        0x0030:  3733 3339 3034 3033 3933 3539 3635 3631  7339040393596561
        0x0040:  3436 3933 3132 3531 3336 3435 3235 352c  469312513645255,
        0x0050:  2022 7665 7273 696f 6e22 3a20 5b32 2c20  ."version":.[2,.
        0x0060:  305d 2c20 2264 6973 706c 6179 6e61 6d65  0],."displayname
        0x0070:  223a 2022 222c 2022 706f 7274 223a 2031  ":."",."port":.1
        0x0080:  3735 3030 2c20 226e 616d 6573 7061 6365  7500,."namespace
        0x0090:  7322 3a20 5b31 3237 3538 3534 3635 385d  s":.[1275854658]
        0x00a0:  7d                                       }
```

More Capture

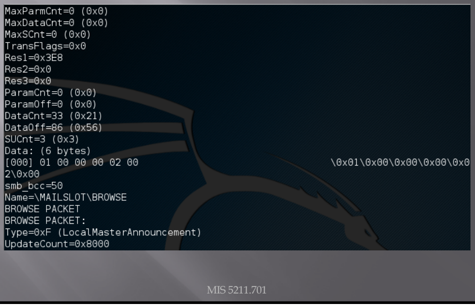


```
11:49:19.594816 IP (tos 0x0, ttl 128, id 5520, offset 0, flags [none], proto UDP
 (17), length 229)
    192.168.198.1.138 > 192.168.198.255.138: [udp sum ok]
>>> NBT UDP PACKET(138) Res=0x1102 ID=0xD345 IP=192 (0xc0).168 (0xa8).198 (0xc6)
.1 (0x1) Port=138 (0x8a) Length=187 (0xbb) Res2=0x0
SourceName=PROFESSOR-HP    NameType=0x20 (Server)
DestName=WORKGROUP       NameType=0x1E (Browser Server)

SMB PACKET: SMBtrans (REQUEST)
SMB Command   =  0x25
Error class   =  0x0
Error code    =  0 (0x0)
Flags1        =  0x0
Flags2        =  0x0
Tree ID       =  0 (0x0)
Proc ID       =  0 (0x0)
UID           =  0 (0x0)
MID           =  0 (0x0)
Word Count    =  17 (0x11)
TotParamCnt=0 (0x0)
TotDataCnt=33 (0x21)
```

Yet More Capture

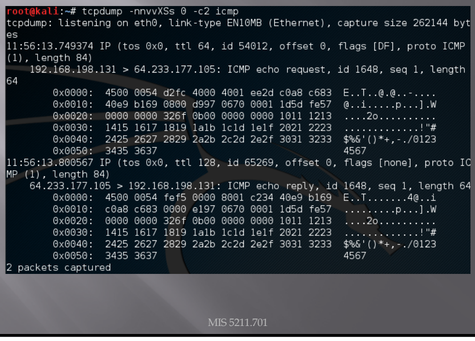
- There is more after this, but I'll stop here.



```
MaxParmCnt=0 (0x0)
MaxDataCnt=0 (0x0)
MaxSCnt=0 (0x0)
TransFlags=0x0
Res1=0x3E8
Res2=0x0
Res3=0x0
ParamCnt=0 (0x0)
ParamOff=0 (0x0)
DataCnt=33 (0x21)
DataOff=86 (0x56)
SUCnt=3 (0x3)
Data: (6 bytes)
[000] 01 00 00 00 02 00                                       \0x01\0x00\0x00\0x00\0x0
2\0x00
smb_bcc=50
Name=\MAILSLOT\BROWSE
BROWSE PACKET
BROWSE PACKET:
Type=0xF (LocalMasterAnnouncement)
UpdateCount=0x8000
```

Adding More

- Try tcpdump –nnxxXSs 0 -c2 icmp
  - 0 Actually results in no data capture even with X and s set
  - -c2 Restricts capture to two packets
  - Using icmp filters so only icmp packet headers are captured
  - See Next slide for example



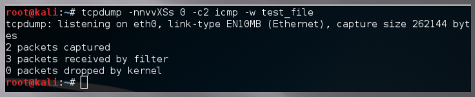
```
root@kali:~# tcpdump -nnvvXSs 0 -c2 icmp
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 byt
es
11:56:13.749374 IP (tos 0x0, ttl 64, id 54012, offset 0, flags [DF], proto ICMP
(1), length 84)
    192.168.198.131 > 64.233.177.105: ICMP echo request, id 1648, seq 1, length
64
        0x0000:  4500 0054 d2fc 4000 4001 ee2d c0a8 c683  E..T..@.@..-....
        0x0010:  40e9 b169 0800 d997 0670 0001 1d5d fe57  @..i.....p...].W
        0x0020:  0000 0000 326f 0b00 0000 0000 1011 1213  ....2o..........
        0x0030:  1415 1617 1819 1a1b 1c1d 1e1f 2021 2223  .............!"#
        0x0040:  2425 2627 2829 2a2b 2c2d 2e2f 3031 3233  $%&'()*+,-./0123
        0x0050:  3435 3637                                4567
11:56:13.800567 IP (tos 0x0, ttl 128, id 65269, offset 0, flags [none], proto IC
MP (1), length 84)
    64.233.177.105 > 192.168.198.131: ICMP echo reply, id 1648, seq 1, length 64
        0x0000:  4500 0054 fef5 0000 8001 c234 40e9 b169  E..T.......4@..i
        0x0010:  c0a8 c683 0000 e197 0670 0001 1d5d fe57  .........p...].W
        0x0020:  0000 0000 326f 0b00 0000 0000 1011 1213  ....2o..........
        0x0030:  1415 1617 1819 1a1b 1c1d 1e1f 2021 2223  .............!"#
        0x0040:  2425 2627 2829 2a2b 2c2d 2e2f 3031 3233  $%&'()*+,-./0123
        0x0050:  3435 3637                                4567
2 packets captured
```

Other Options

- Try adding host to look for traffic based on IP address (also works with hostname if you're not using -n)
- Try adding SRC or DST to find traffic from only a source or destination (eliminates one side of a host conversation)
- Try adding port to see only traffic to or from a certain port
- Lots more: portrange, less/more, or >/<

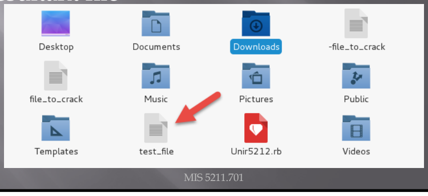
Writing to a File

- Try –w to write to a file



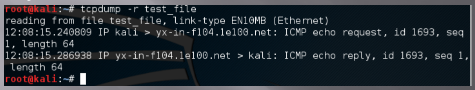
```
root@kali:~# tcpdump -nnvvXSs 0 -c2 icmp -w test_file
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 byt
es
2 packets captured
3 packets received by filter
0 packets dropped by kernel
root@kali:~# 
```

- Resultant file



Reading Files

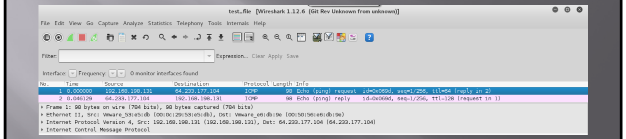
- Try –r to read a file in



```
root@kali:~# tcpdump -r test_file
reading from file test_file, link-type EN10MB (Ethernet)
12:08:15.240809 IP kali > yx-in-f104.1e100.net: ICMP echo request, id 1693, seq
1, length 64
12:08:15.286938 IP yx-in-f104.1e100.net > kali: ICMP echo reply, id 1693, seq 1,
 length 64
root@kali:~# 
```

File Contents

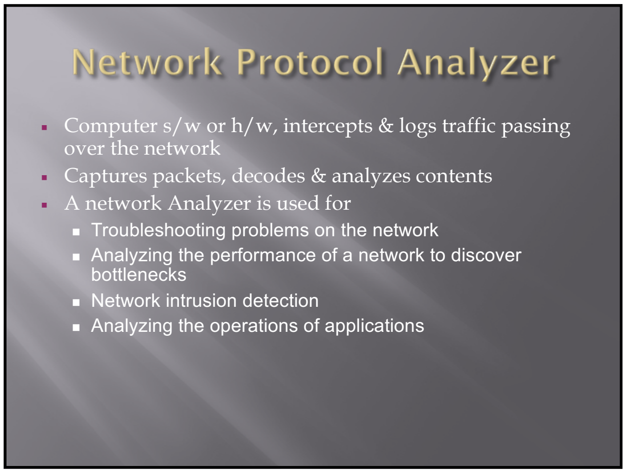
- Opens by default in Wireshark


test_file [Wireshark 1.12.6 (Git Rev Unknown from unknown)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save
Interface:
Frequency:
0 monitor interfaces found

```
No.  Time      Source           Destination      Protocol Length Info
   1 0.000000  192.168.198.131  64.233.177.104   ICMP         98 Echo (ping) request  id=0x069d, seq=1/256, ttl=64 (reply in 2)
   2 0.046129  64.233.177.104   192.168.198.131  ICMP         98 Echo (ping) reply    id=0x069d, seq=1/256, ttl=128 (request in 1)
▸ Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits)
▸ Ethernet II, Src: Vmware_53:e5:db (00:0c:29:53:e5:db), Dst: Vmware_e6:db:9e (00:50:56:e6:db:9e)
▸ Internet Protocol Version 4, Src: 192.168.198.131 (192.168.198.131), Dst: 64.233.177.104 (64.233.177.104)
▸ Internet Control Message Protocol
```

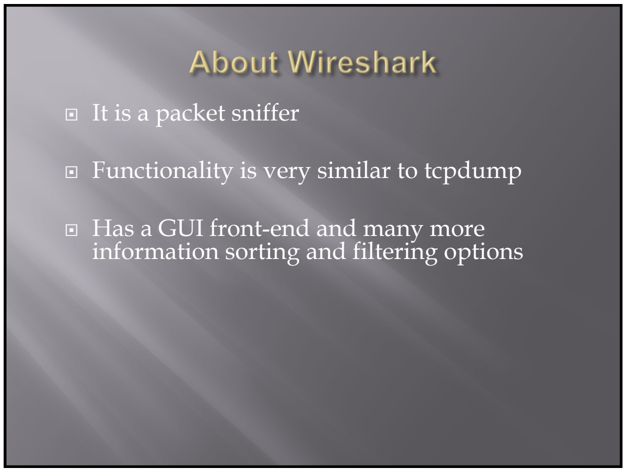
Reference for tcpdump

- Lots more at:
- http://www.tcpdump.org/



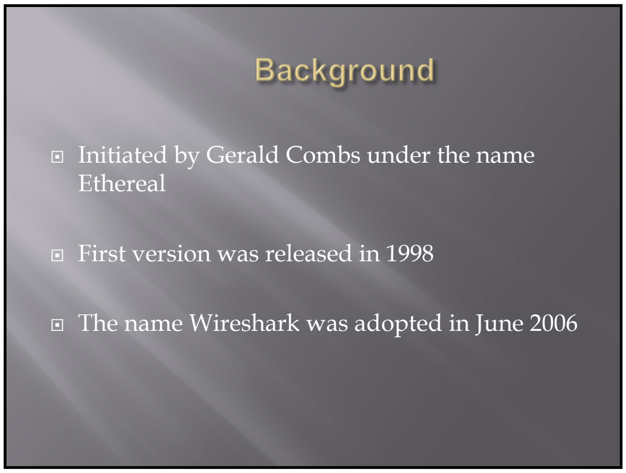
Network Protocol Analyzer

- Computer s/w or h/w, intercepts & logs traffic passing over the network
- Captures packets, decodes & analyzes contents
- A network Analyzer is used for
  - Troubleshooting problems on the network
  - Analyzing the performance of a network to discover bottlenecks
  - Network intrusion detection
  - Analyzing the operations of applications



About Wireshark

- It is a packet sniffer
- Functionality is very similar to tcpdump
- Has a GUI front-end and many more information sorting and filtering options



Background

- Initiated by Gerald Combs under the name Ethereal
- First version was released in 1998
- The name Wireshark was adopted in June 2006

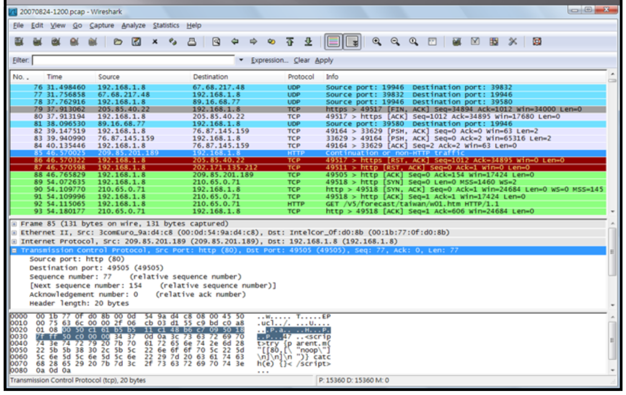
Features

- “Understands" the structure of different network protocols.
- Displays encapsulation and single fields and interprets their meaning.
- It can only capture on networks supported by pcap.
- It is cross-platform running on various OS (Linux, Mac OS X, Microsoft windows)

WinP Cap

- Industry standard tool for link layer network access in windows environment
- Allows application to capture and transmit network packets by passing the protocol stack
- Consists of a driver-extends OS to provide low level network access
- Consists of library for easy access to low level network layers
- Also contains windows version of libPCap Unix API

Example



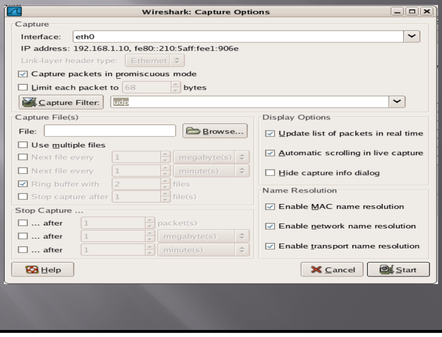
```
No. .  Time       Source           Destination      Protocol  Info
    76 31.498460  192.168.1.8      67.68.217.48     UDP       Source port: 19946  Destination port: 39832
    77 31.756858  67.68.217.48     192.168.1.8      UDP       Source port: 39832  Destination port: 19946
    78 37.762916  192.168.1.8      89.16.68.77      UDP       Source port: 19946  Destination port: 39580
    79 37.913062  205.85.40.22     192.168.1.8      TCP       https > 49517 [FIN, ACK] Seq=34894 Ack=1012 Win=34000 Len=0
    80 37.913194  192.168.1.8      205.85.40.22     TCP       49517 > https [ACK] Seq=1012 Ack=34895 Win=17680 Len=0
    81 38.096530  89.16.68.77      192.168.1.8      UDP       Source port: 39580  Destination port: 19946
    82 39.147519  192.168.1.8      76.87.145.159    TCP       49164 > 33629 [PSH, ACK] Seq=0 Ack=0 Win=63 Len=2
    83 39.940990  76.87.145.159    192.168.1.8      TCP       33629 > 49164 [PSH, ACK] Seq=0 Ack=2 Win=65316 Len=2
    84 40.135446  192.168.1.8      76.87.145.159    TCP       49164 > 33629 [ACK] Seq=2 Ack=2 Win=63 Len=0
    85 46.570025  209.85.201.189   192.168.1.8      HTTP      Continuation or non-HTTP traffic
    86 46.570322  192.168.1.8      205.85.40.22     TCP       49517 > https [RST, ACK] Seq=1012 Ack=34895 Win=0 Len=0
    87 46.570598  192.168.1.8      202.171.135.212  TCP       49511 > http [RST, ACK] Seq=0 Ack=1 Win=0 Len=0
    88 46.765829  192.168.1.8      209.85.201.189   TCP       49505 > http [ACK] Seq=0 Ack=154 Win=17424 Len=0
    89 54.072635  192.168.1.8      210.65.0.71      TCP       49518 > http [SYN] Seq=0 Len=0 MSS=1460 WS=2
    90 54.109770  210.65.0.71      192.168.1.8      TCP       http > 49518 [SYN, ACK] Seq=0 Ack=1 Win=24684 Len=0 WS=0 MSS=145
    91 54.109996  192.168.1.8      210.65.0.71      TCP       49518 > http [ACK] Seq=1 Ack=1 Win=17424 Len=0
    92 54.115065  192.168.1.8      210.65.0.71      HTTP      GET /V5/forecast/taiwan/w01.htm HTTP/1.1
    93 54.180177  210.65.0.71      192.168.1.8      TCP       http > 49518 [ACK] Seq=1 Ack=606 Win=24684 Len=0
```

```
⊞ Frame 85 (131 bytes on wire, 131 bytes captured)
⊞ Ethernet II, Src: 3comEuro_9a:d4:c8 (00:0d:54:9a:d4:c8), Dst: IntelCor_0f:d0:8b (00:1b:77:0f:d0:8b)
⊞ Internet Protocol, Src: 209.85.201.189 (209.85.201.189), Dst: 192.168.1.8 (192.168.1.8)
⊟ Transmission Control Protocol, Src Port: http (80), Dst Port: 49505 (49505), Seq: 77, Ack: 0, Len: 77
    Source port: http (80)
    Destination port: 49505 (49505)
    Sequence number: 77    (relative sequence number)
    [Next sequence number: 154    (relative sequence number)]
    Acknowledgement number: 0    (relative ack number)
    Header length: 20 bytes
```

```
0000  00 1b 77 0f d0 8b 00 0d  54 9a d4 c8 08 00 45 50   ..w..... T.....EP
0010  00 75 63 6c 00 00 2f 06  cb 03 d1 55 c9 bd c0 a8   .ucl../. ...U....
0020  01 08 00 50 c1 61 b5 b5  11 c1 48 b6 c7 09 50 18   ...P.a.. ..H...P.
0030  7f ff 50 c0 00 00 34 37  0d 0a 3c 73 63 72 69 70   ..P...47 ..<scrip
0040  74 3e 74 72 79 20 7b 70  61 72 65 6e 74 2e 6d 28   t>try {p arent.m(
0050  22 5b 5b 38 30 2c 5b 5c  22 6e 6f 6f 70 5c 22 5d   "[[80,[\ "noop\"]
0060  5c 6e 5d 5c 6e 5d 5c 6e  22 29 7d 20 63 61 74 63   \n]\n]\n ")} catc
0070  68 28 65 29 20 7b 7d 3c  2f 73 63 72 69 70 74 3e   h(e) {}< /script>
0080  0a 0d 0a                                           ...
```

```
Transmission Control Protocol (tcp), 20 bytes        P: 15360 D: 15360 M: 0
```

Capture Options



Capture

Interface: eth0

IP address: 192.168.1.10, fe80::210:5aff:fee1:906e

Link-layer header type: Ethernet

☑ Capture packets in promiscuous mode

☐ Limit each packet to 68 bytes

Capture Filter: udp

Capture File(s)

File:

☐ Use multiple files

☐ Next file every 1 megabyte(s)

☐ Next file every 1 minute(s)

☑ Ring buffer with 2 files

☐ Stop capture after 1 file(s)

Stop Capture ...

☐ ... after 1 packet(s)

☐ ... after 1 megabyte(s)

☐ ... after 1 minute(s)

Display Options

☑ Update list of packets in real time

☑ Automatic scrolling in live capture

☐ Hide capture info dialog

Name Resolution

☑ Enable MAC name resolution

☑ Enable network name resolution

☑ Enable transport name resolution

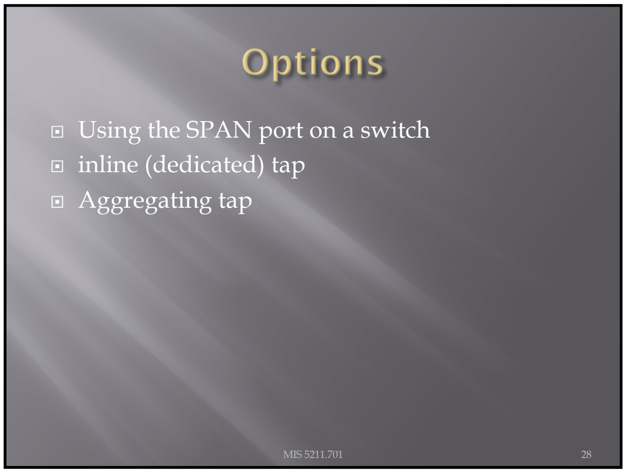
Wireshark

- Reference
  - http://www.wireshark.org/docs/wsug_html_chunked/
- Books
  - https://www.amazon.com/dp/1593271492/?tag=stackoverfl08-20
  - https://www.amazon.com/dp/1597490733/?tag=stackoverfl08-20
- Tutorials
  - https://cs.gmu.edu/~astavrou/courses/ISA_564_F15/Wireshark-Tutorial.pdf
- Blog
  - https://blog.wireshark.org/

Packet Sniffing or Taps

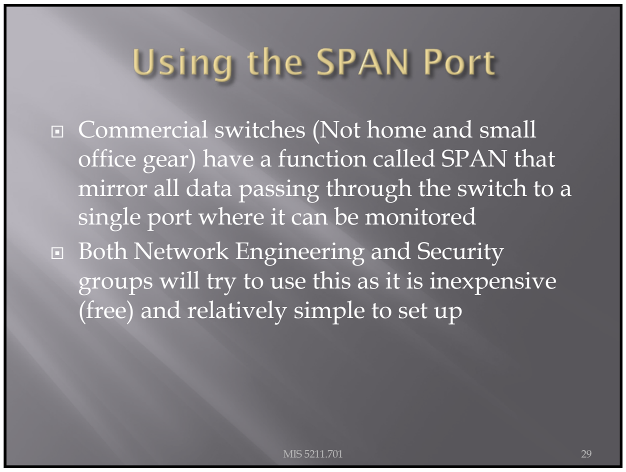
Packet Sniffer Definition:

A **packet sniffer** is a wire-tap device that plugs into computer networks and eavesdrops on the network traffic.



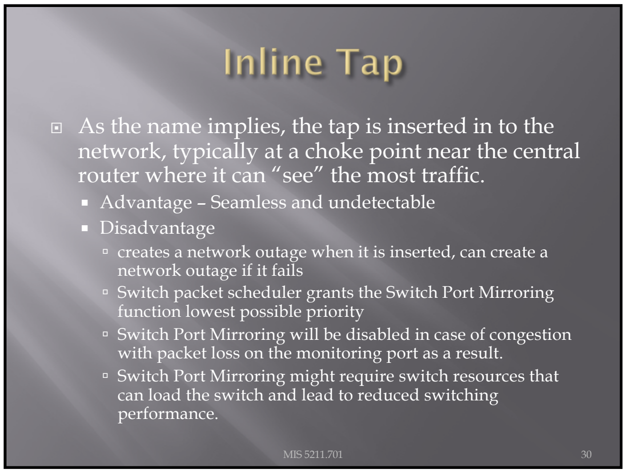
Options

- Using the SPAN port on a switch
- inline (dedicated) tap
- Aggregating tap



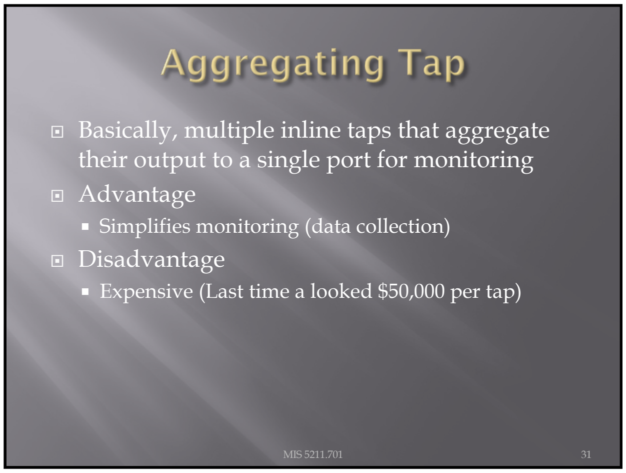
Using the SPAN Port

- Commercial switches (Not home and small office gear) have a function called SPAN that mirror all data passing through the switch to a single port where it can be monitored
- Both Network Engineering and Security groups will try to use this as it is inexpensive (free) and relatively simple to set up



Inline Tap

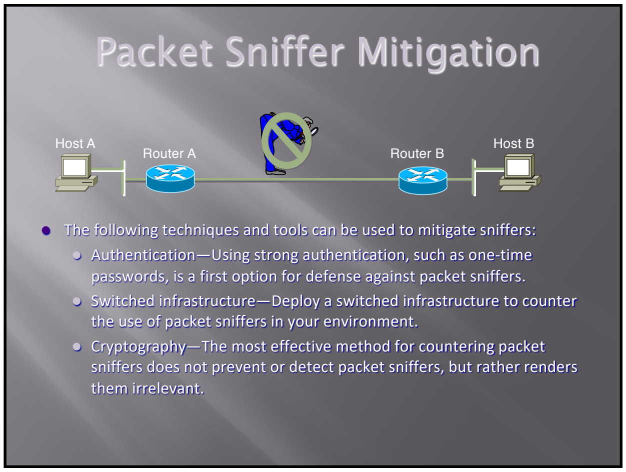
- As the name implies, the tap is inserted in to the network, typically at a choke point near the central router where it can “see” the most traffic.
  - Advantage – Seamless and undetectable
  - Disadvantage
    - creates a network outage when it is inserted, can create a network outage if it fails
    - Switch packet scheduler grants the Switch Port Mirroring function lowest possible priority
    - Switch Port Mirroring will be disabled in case of congestion with packet loss on the monitoring port as a result.
    - Switch Port Mirroring might require switch resources that can load the switch and lead to reduced switching performance.



Aggregating Tap

- Basically, multiple inline taps that aggregate their output to a single port for monitoring
- Advantage
  - Simplifies monitoring (data collection)
- Disadvantage
  - Expensive (Last time a looked $50,000 per tap)

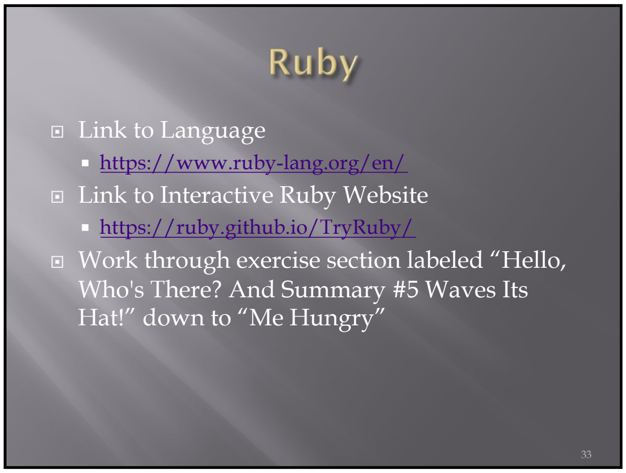
MIS 5211.701 31



Packet Sniffer Mitigation

Host A
Router A
Router B
Host B

- The following techniques and tools can be used to mitigate sniffers:
  - Authentication—Using strong authentication, such as one-time passwords, is a first option for defense against packet sniffers.
  - Switched infrastructure—Deploy a switched infrastructure to counter the use of packet sniffers in your environment.
  - Cryptography—The most effective method for countering packet sniffers does not prevent or detect packet sniffers, but rather renders them irrelevant.



Ruby

- Link to Language
  - https://www.ruby-lang.org/en/
- Link to Interactive Ruby Website
  - https://ruby.github.io/TryRuby/
- Work through exercise section labeled “Hello, Who's There? And Summary #5 Waves Its Hat!” down to “Me Hungry”

33

Questions

?

MIS 5211.701 34