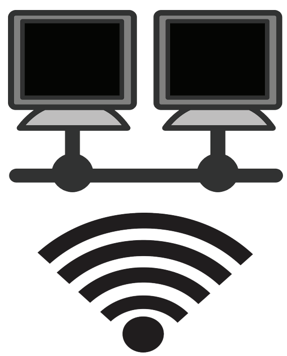
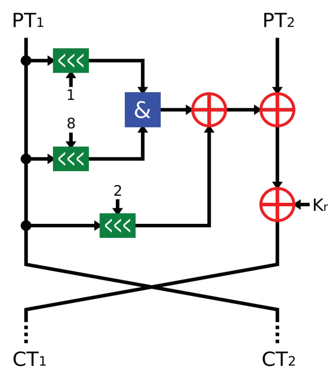
CSE 40567 / 60567: Computer Security





Network Security 1

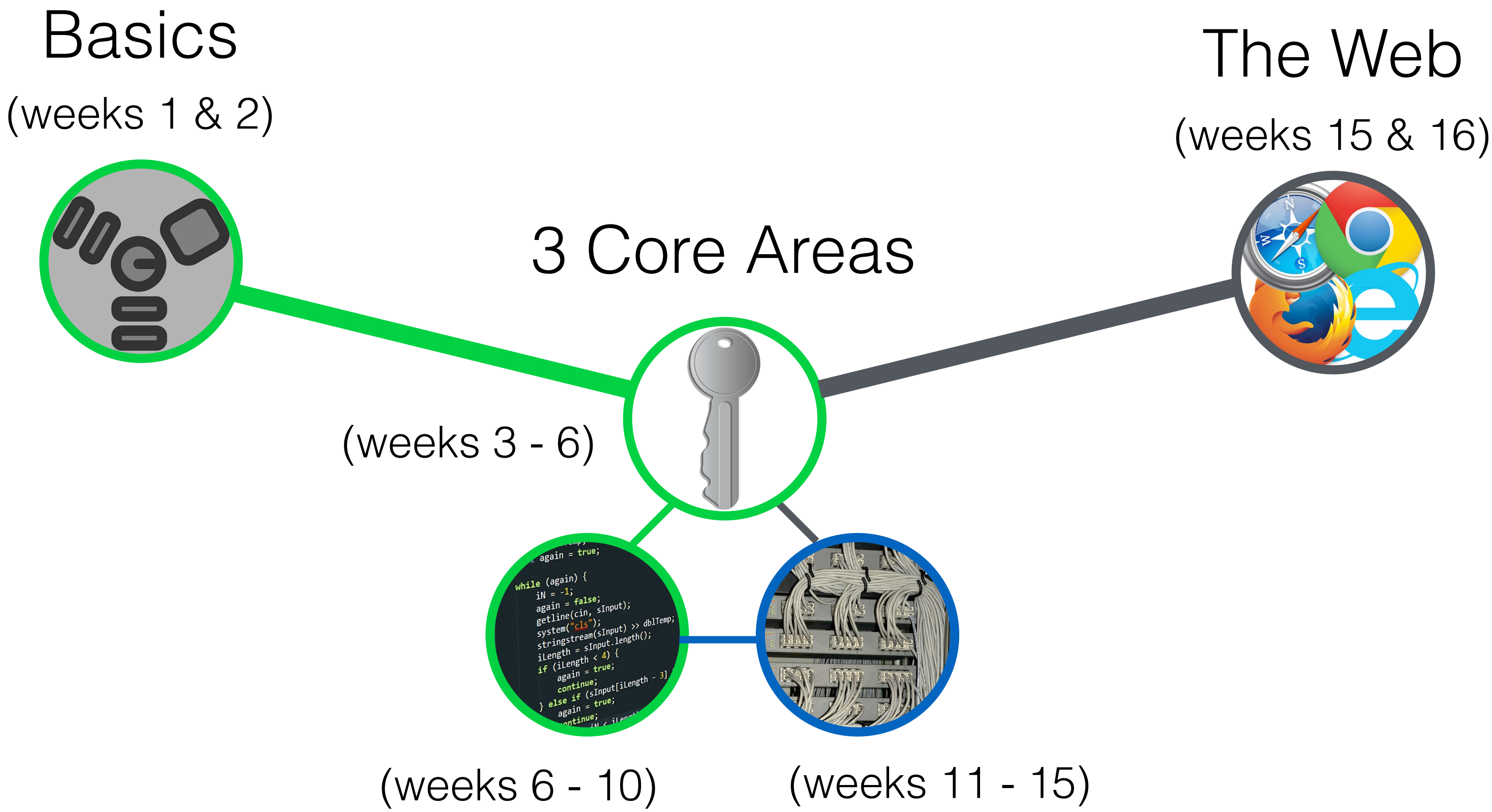
Homework #5 is due tonight at 11:59PM (your timezone)

See **Assignments Page** on the course website for details

Live Tuesday in Class
RC Johnson (PayPal)

(Will not be recorded)

Course Roadmap



Heartbleed

What is it? Serious input validation vulnerability in OpenSSL

Bug introduced: OpenSSL version 1.0.1 on March 14, 2012

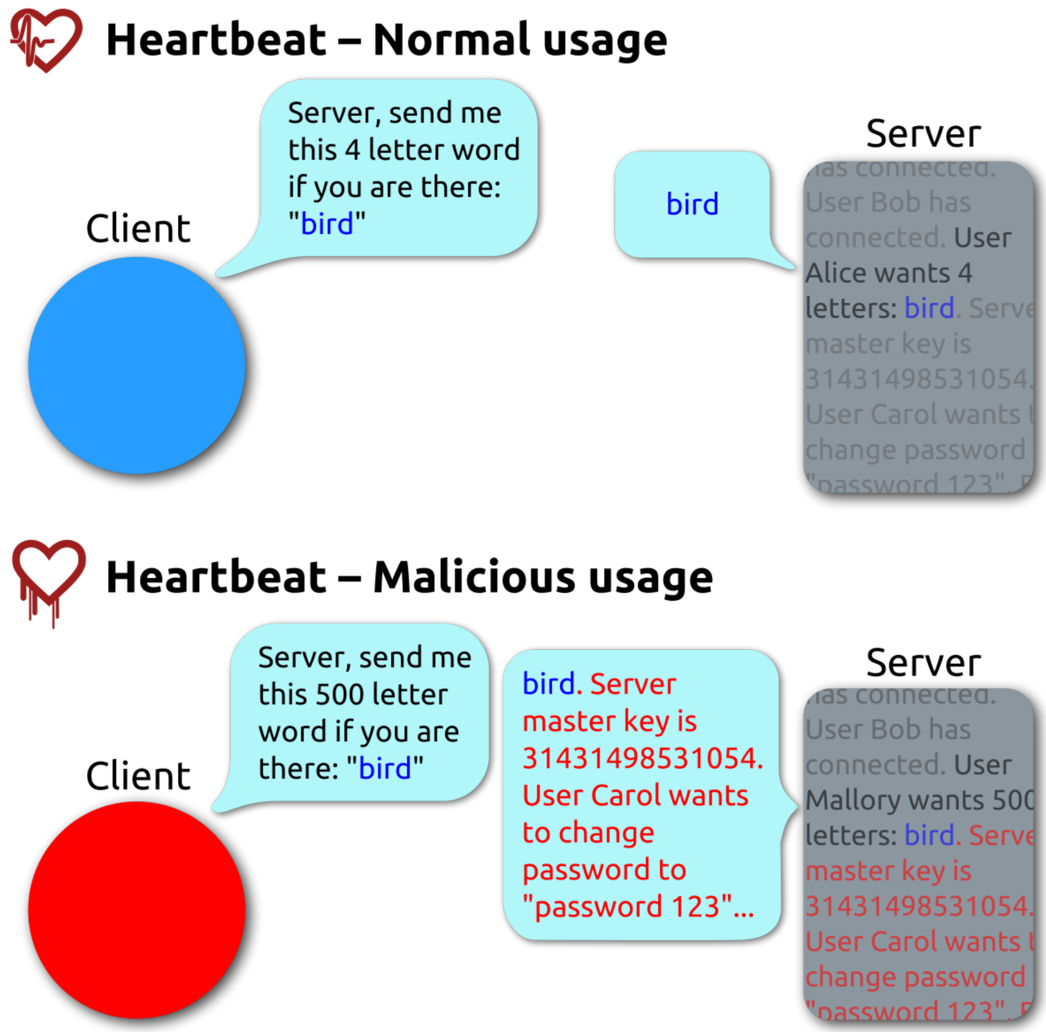
Bug disclosed: April 7th, 2014

Protocol affected: TLS

- Successor to SSL
- Meant to secure network traffic from eavesdropping attacks
- Good example of security software making things less secure

http://heartbleed.com/

Heartbleed bug explained



The bug in OpenSSL

`p` is a pointer to start of message

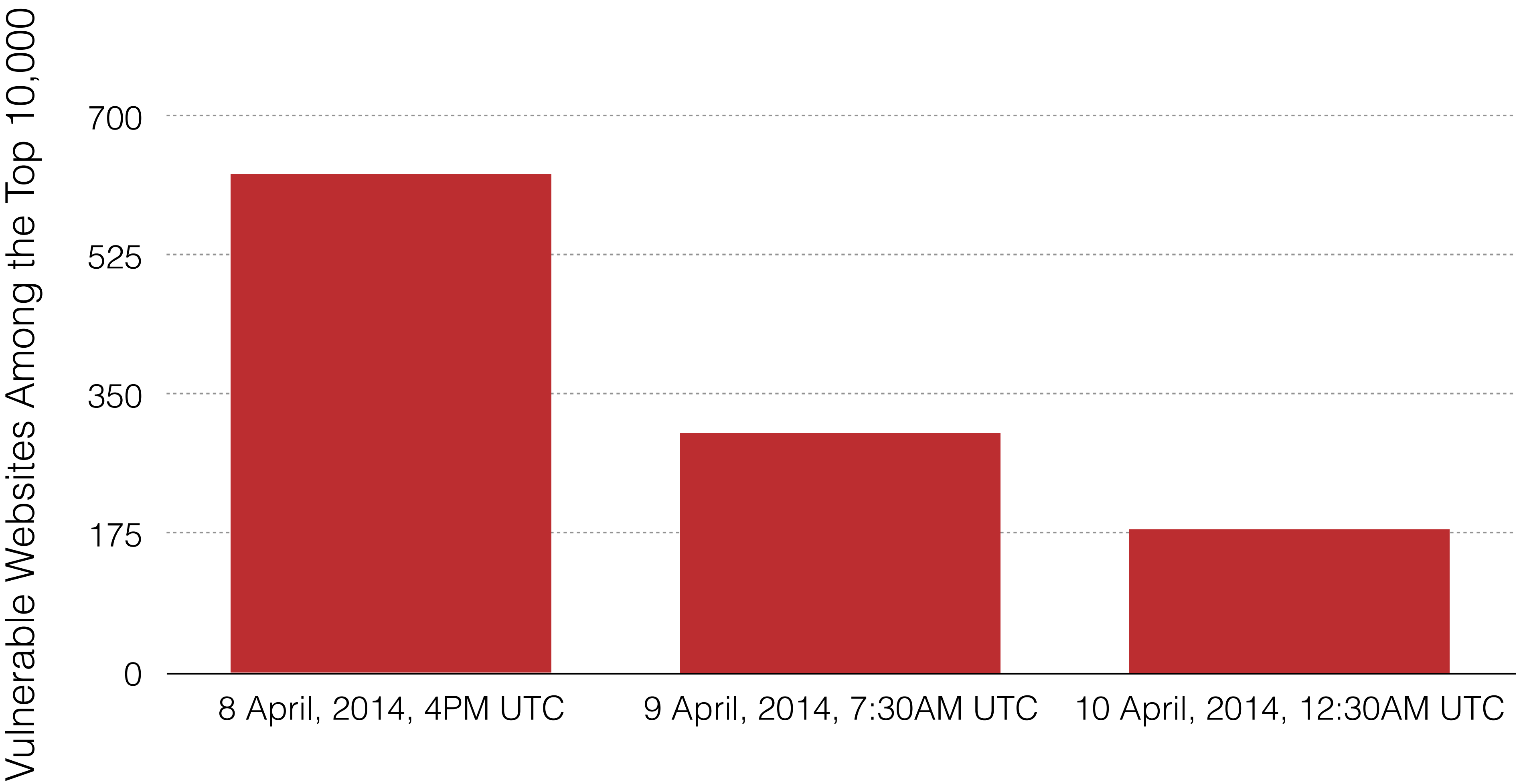
```
/* Read type and payload length first */
hbtype = *p++;
n2s(p, payload);
pl = p;
```

No bounds checking to enforce consistency between `payload_length` and the actual `payload`

The fix:

```
hbtype = *p++;
n2s(p, payload);
if (1 + 2 + payload + 16 > s->s3->rrec.length)
    return 0; /* silently discard per RFC 6520
                 sec. 4 */
pl = p;
```

Scope of the vulnerability



Response from the community

"OpenSSL is not developed by a responsible team."

- Theo de Raadt

Large mistake: OpenSSL developers wrote their own memory management routines, circumventing library protections

"We are building the most important technologies for the global economy on shockingly underfunded infrastructure."

- Dan Kaminsky

An often overlooked downside of the Open Source movement

LibreSSL

http://www.libressl.org/

Fork of OpenSSL by OpenBSD Project

Removed 90,000 lines of C code in its first week

Key changes:

Replaced custom memory calls with ones in the standard library

- Enables ASLR, use of the NX bit, stack canaries

Planning for the future

- Enables compiler flag checks, fixes for 2038 compatibility

Proper use of cryptographic primitives

Removal of old code and insecure features

Brief Review of TCP/IP

Protocols

The basic protocols that underly the Internet are very mature

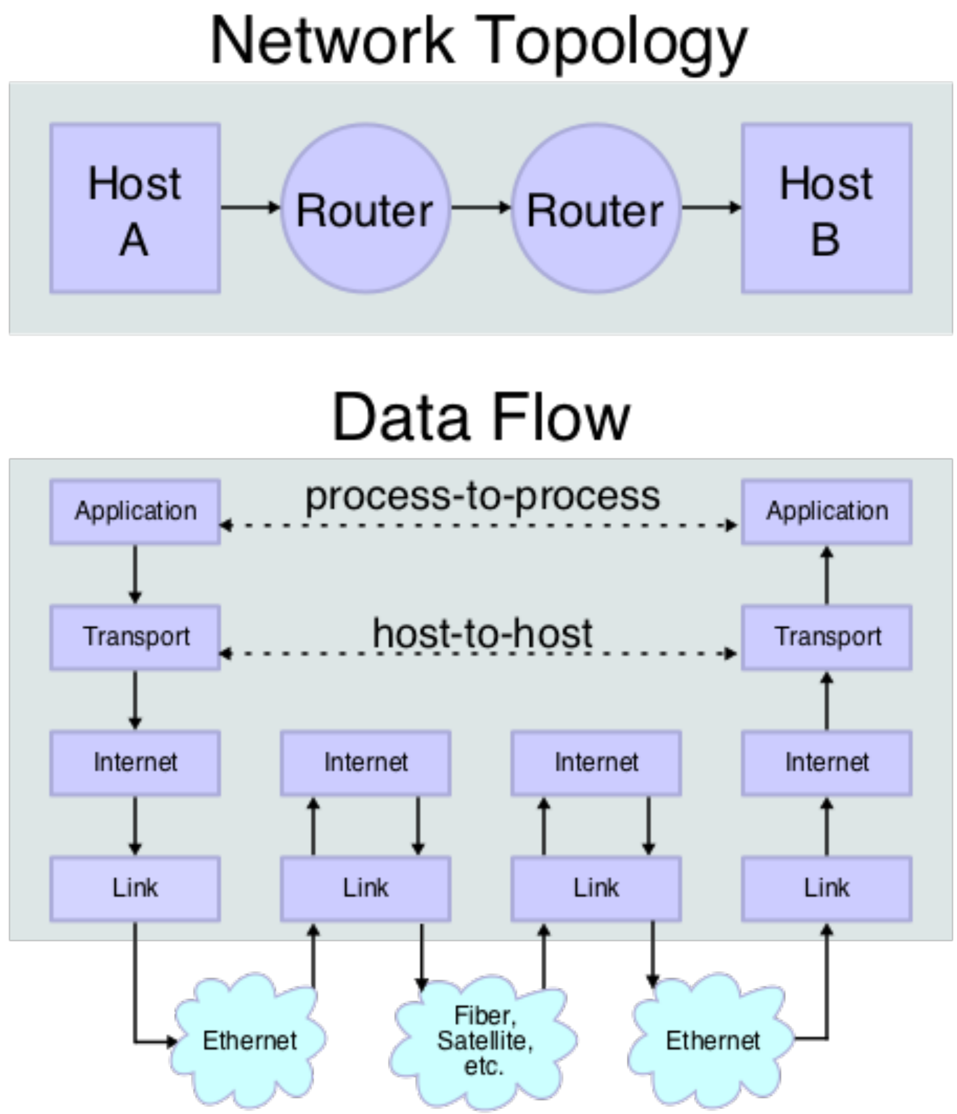
1983: IPv4 deployed to ARPANET

1998: IPv6 formalized by IETF

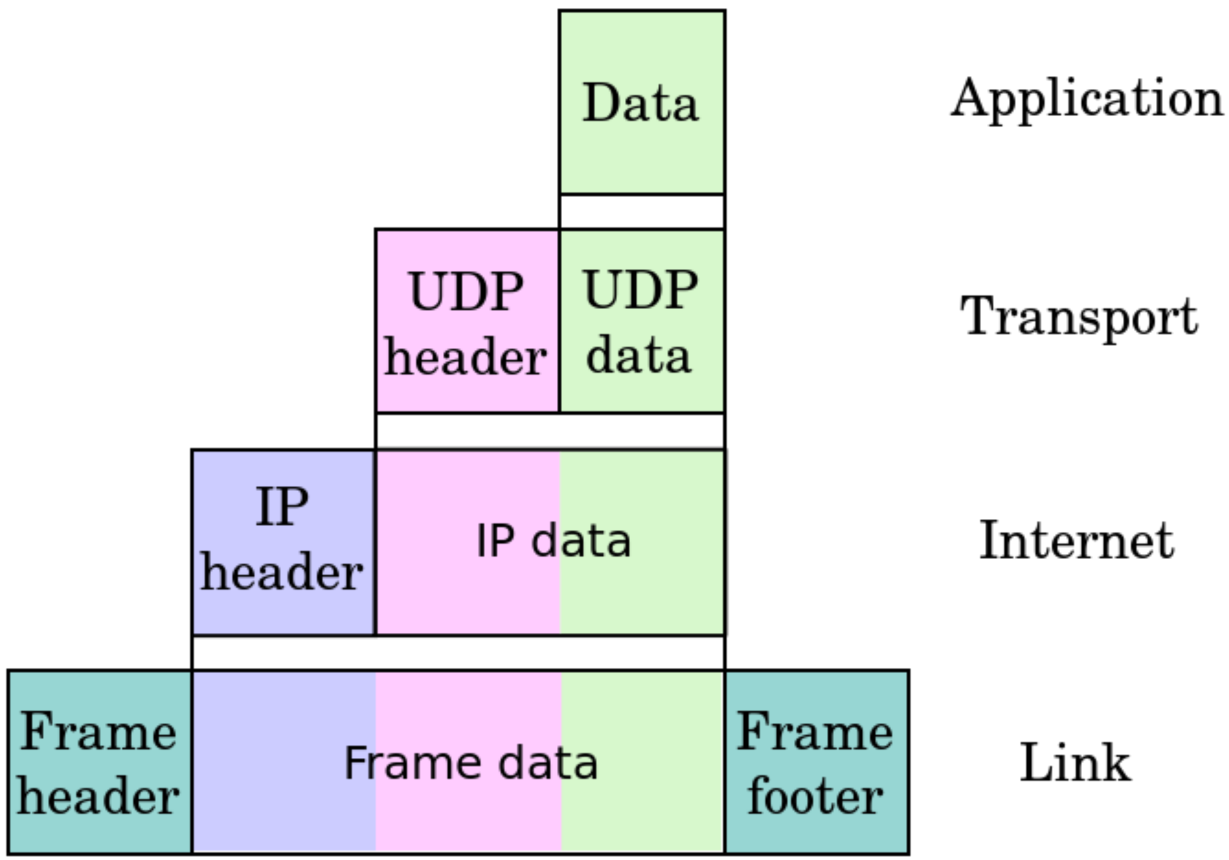
- Still waiting for universal adoption

Most work in networking (and new security vulnerabilities) now stems from emerging applications

Operation of the TCP/IP Protocol Suite



Protocol layers



Encapsulation of application data descending through the layered IP architecture

ARP Header

Internet Protocol (IPv4) over Ethernet ARP packet		
octet offset	**0**	**1**
0	Hardware type (HTYPE)	
2	Protocol type (PTYPE)	
4	Hardware address length (HLEN)	Protocol address length (PLEN)
6	Operation (OPER)	
8	Sender hardware address (SHA) (first 2 bytes)	
10	(next 2 bytes)	
12	(last 2 bytes)	
14	Sender protocol address (SPA) (first 2 bytes)	
16	(last 2 bytes)	
18	Target hardware address (THA) (first 2 bytes)	
20	(next 2 bytes)	
22	(last 2 bytes)	
24	Target protocol address (TPA) (first 2 bytes)	
26	(last 2 bytes)	

IPv4 Header

IPv4 Header Format

<table>
<tr><th>Offsets</th><th>Octet</th><th colspan="8">0</th><th colspan="8">1</th><th colspan="8">2</th><th colspan="8">3</th></tr>
<tr><th>Octet</th><th>Bit</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><th>0</th><th>0</th><td colspan="4">Version</td><td colspan="4">IHL</td><td colspan="6">DSCP</td><td colspan="2">ECN</td><td colspan="16">Total Length</td></tr>
<tr><th>4</th><th>32</th><td colspan="16">Identification</td><td colspan="3">Flags</td><td colspan="13">Fragment Offset</td></tr>
<tr><th>8</th><th>64</th><td colspan="8">Time To Live</td><td colspan="8">Protocol</td><td colspan="16">Header Checksum</td></tr>
<tr><th>12</th><th>96</th><td colspan="32">Source IP Address</td></tr>
<tr><th>16</th><th>128</th><td colspan="32">Destination IP Address</td></tr>
<tr><th>20</th><th>160</th><td colspan="32">Options (if IHL > 5)</td></tr>
</table>

IPv6 Header

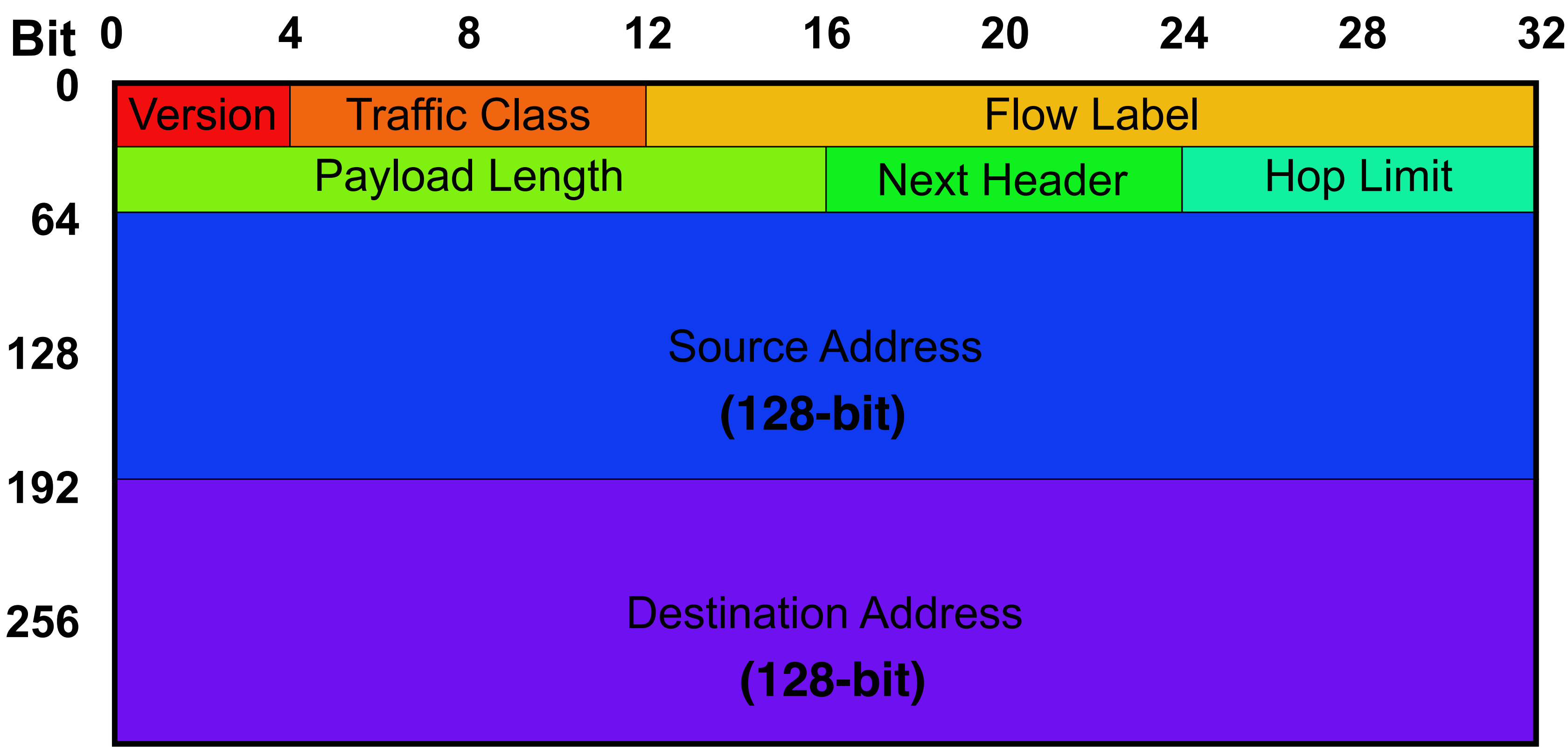
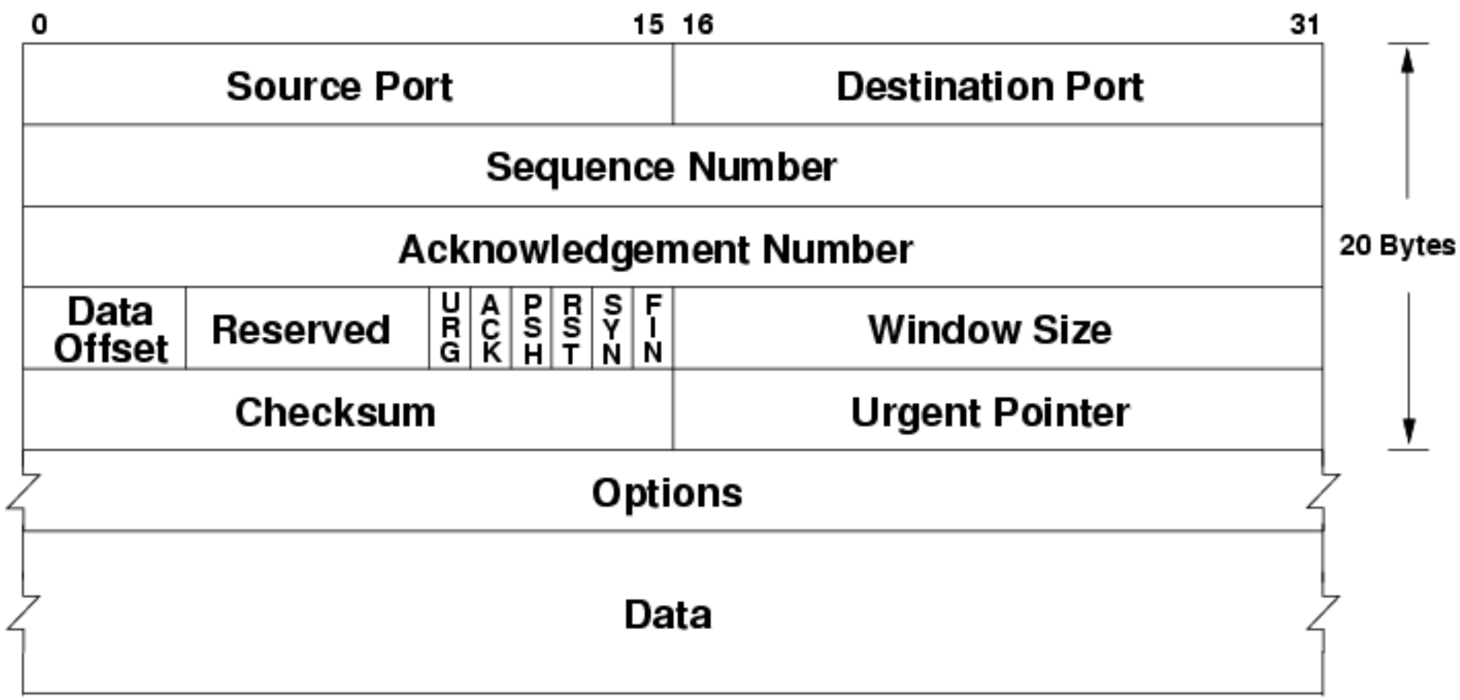
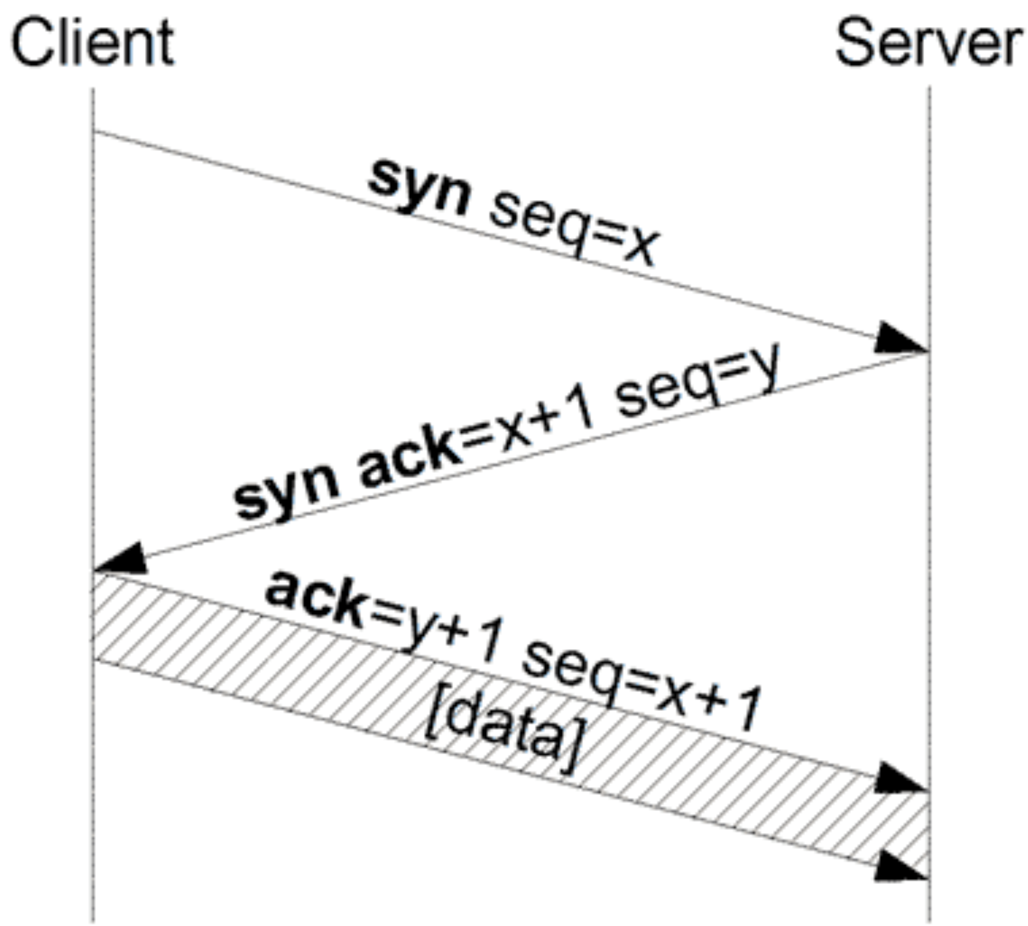


Diagram of the IPv6 packet header.

TCP Header

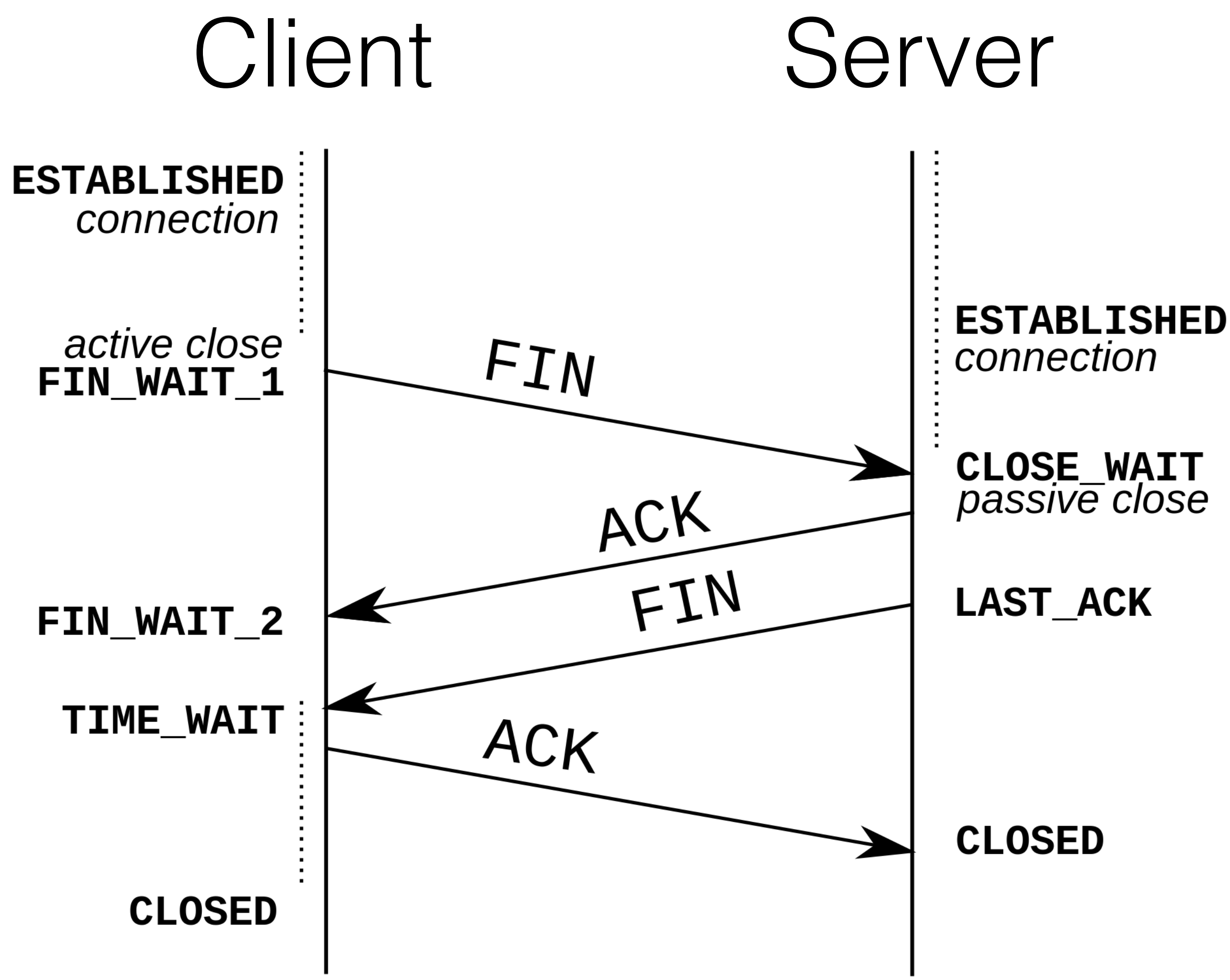

0
15
16
31
Source Port
Destination Port
Sequence Number
Acknowledgement Number
Data Offset
Reserved
URG
ACK
PSH
RST
SYN
FIN
Window Size
Checksum
Urgent Pointer
Options
Data
20 Bytes

TCP Connections

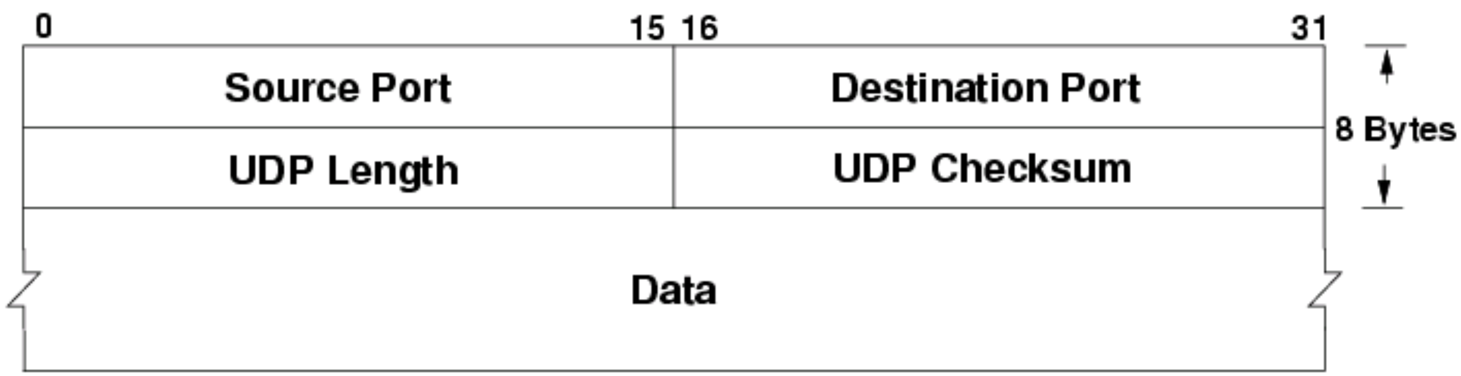


Tcp-handshake.png

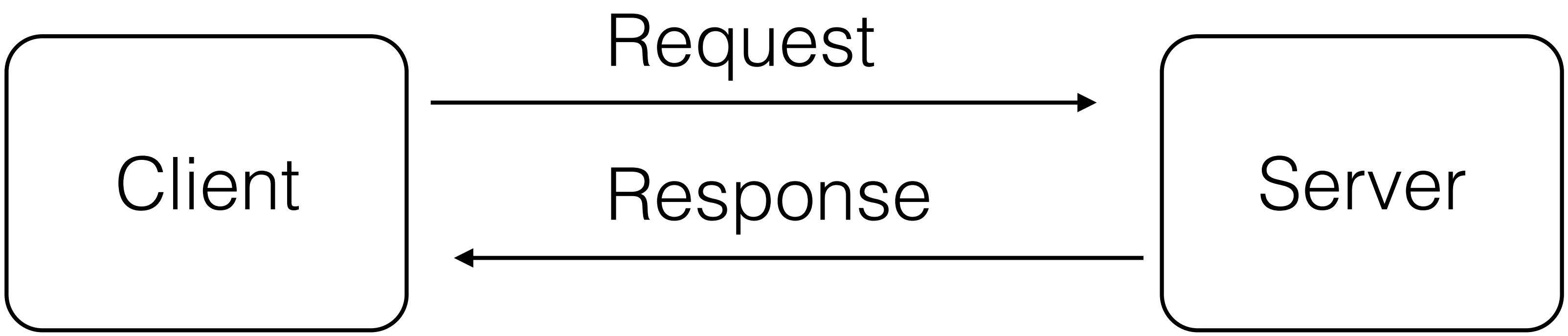
TCP Connections



UDP Header


0
15 16
31
Source Port
Destination Port
UDP Length
UDP Checksum
8 Bytes
Data

UDP Request / Response Regime



ICMP Header

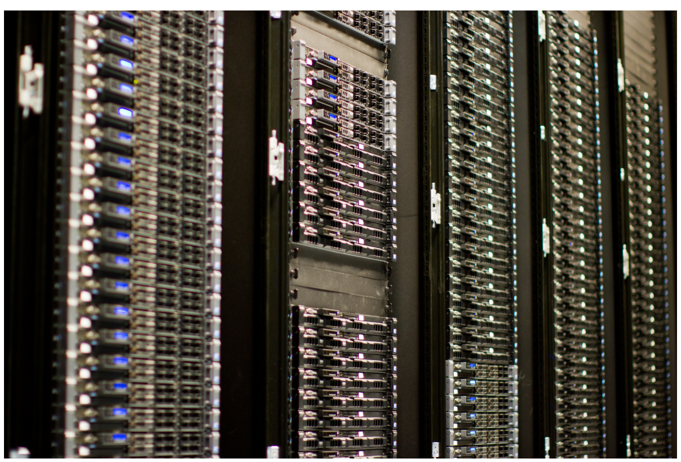
8-bit ICMP Type	8-bit ICMP Code	16-bit ICMP Checksum
ICMP Contents (dependent on type and code)		

Network Eavesdropping

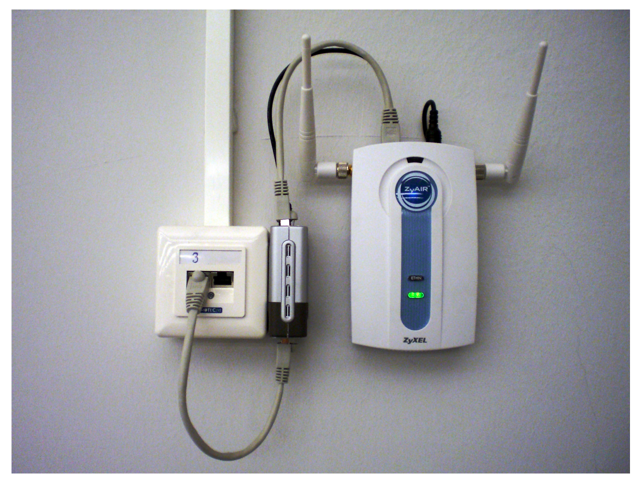
Good places for eavesdropping

At the router (capture traffic from multiple networks)

Personal computers



Multi-user servers



Unprotected Wireless APs

tcpdump

- Trusty Unix packet sniffer (command line interface)
- Requires root privilege to capture network traffic on an interface

List interfaces on which tcpdump can listen:

```
# tcpdump -D
1.eth0
2.eth1
3.usbmon1 (USB bus number 1)
4.usbmon2 (USB bus number 2)
5.usbmon3 (USB bus number 3)
6.usbmon4 (USB bus number 4)
7.usbmon5 (USB bus number 5)
8.usbmon6 (USB bus number 6)
9.usbmon7 (USB bus number 7)
10.usbmon8 (USB bus number 8)
11.any (Pseudo-device that captures on all interfaces)
12.lo
```

tcpdump

Listen on interface eth0 (first ethernet device)

```
# tcpdump -i eth0
tcpdump: verbose output suppressed, use -v or -vv for full
protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size
65535 bytes
16:25:12.840541 IP dhcp-140-247-178-194.fas.harvard.edu.
5900 > c-73-168-122-160.hsd1.in.comcast.net.61227: Flags
[.], seq 11760:13208, ack 1, win 272, options [nop,nop,TS
val 4196170389 ecr 1186915574], length 1448
16:25:12.840545 IP dhcp-140-247-178-194.fas.harvard.edu.
5900 > c-73-168-122-160.hsd1.in.comcast.net.61227: Flags
[P.], seq 13208:13216, ack 1, win 272, options
[nop,nop,TS val 4196170389 ecr 1186915574], length 8
16:25:12.840690 IP dhcp-140-247-178-194.fas.harvard.edu.
5900 > c-73-168-122-160.hsd1.in.comcast.net.61227: Flags
[P.], seq 13216:13680, ack 1, win 272, options
[nop,nop,TS val 4196170389 ecr 1186915574], length 464
```

…

`tcpdump`

Targeted sniffing is often more useful.

Capture packets to a particular destination:

```
tcpdump -n dst host 192.168.1.1
```

Capture packets from a particular source:

```
tcpdump -n src host 192.168.1.1
```

Capture packets to / from a particular host:

```
tcpdump -n host 192.168.1.1
```

Capture packets to / from a particular network:

```
tcpdump -n net 192.168.1.0/24
```

tcpdump

Capture packets related to a specific port:

```
tcpdump -n port 22
```

Capture packets related to a range of tcp ports:

```
tcpdump -n tcp portrange 1-1023
```

Capture packets to a range of udp ports:

```
tcpdump -n udp portrange 1-1023
```

Capture ARP packets:

```
tcpdump -v arp
```

Capture ICMP packets:

```
tcpdump -v icmp
```

DNS resolution

```
$ host nd.edu
```

```
# tcpdump -n udp port 53
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
15:29:01.393146 IP 140.247.178.194.38502 > 140.247.233.163.53: 39751+ A? nd.edu. (24)
15:29:01.393157 IP 140.247.178.194.38502 > 140.247.233.194.53: 39751+ A? nd.edu. (24)
15:29:01.393162 IP 140.247.178.194.38502 > 128.103.1.7.53: 39751+ A? nd.edu. (24)
15:29:01.394643 IP 140.247.233.163.53 > 140.247.178.194.38502: 39751 1/0/0 A 52.6.129.12 (40)
15:29:01.395155 IP 140.247.178.194.7260 > 140.247.233.163.53: 13012+ AAAA? nd.edu. (24)
15:29:01.396338 IP 140.247.233.163.53 > 140.247.178.194.7260: 13012 0/1/0 (68)
15:29:01.396791 IP 140.247.178.194.43203 > 140.247.233.163.53: 29425+ MX? nd.edu. (24)
15:29:01.397814 IP 140.247.233.163.53 > 140.247.178.194.43203: 29425 2/0/0 MX mail-mx3-prod-v.cc.nd.edu. 50, MX mail-mx4-prod-v.cc.nd.edu. 50 (91)
```

`nc`

Swiss army knife of socket tools

`$ nc 192.168.0.1 80` ← **raw connection to web server**

- Outbound or inbound connections, TCP or UDP, to or from any ports
- Full DNS forward/reverse checking, with appropriate warnings
- Ability to use any local source port
- Ability to use any locally configured network source address
- Built-in port-scanning capabilities, with randomization
- Built-in loose source-routing capability
- Slow-send mode, one line every N seconds
- Hex dump of transmitted and received data
- Tunneling mode which permits user-defined tunneling

Web session

```
$ nc www.google.com 80
HEAD / HTTP/1.0

HTTP/1.0 200 OK
Date: Sun, 14 Feb 2016 21:11:52 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=ISO-8859-1
P3P: CP="This is not a P3P policy! See https://www.google.com/
support/accounts/answer/151657?hl=en for more info."
Server: gws
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
Set-Cookie:
NID=76=ci6ojYCjvFMJM8zqU8gwyYKk88yr0B9iSpOAcwKEk5k2NgLm7IOAcO6pw9i
Jxbb8w9MsET2p-J-i0V0b2VBdvnTU8H6XZIlqqw6dT9ZxwcXw9-
Tb8EvcyTYQLRUiesU6_YAd8Ualo3fanw; expires=Mon, 15-Aug-2016
21:11:52 GMT; path=/; domain=.google.com; HttpOnly
Accept-Ranges: none
Vary: Accept-Encoding
```

Web session (3-way handshake)

```
# tcpdump -n tcp port 80

tcpdump: verbose output suppressed, use -v or -vv for full protocol
decode

listening on eth0, link-type EN10MB (Ethernet), capture size 65535
bytes

16:04:55.589296 IP 140.247.178.194.37612 > 4.53.56.118.80: Flags
[S], seq 108294392, win 14600, options [mss 1460,sackOK,TS val
4217466076 ecr 0,nop,wscale 7], length 0

16:04:55.590276 IP 4.53.56.118.80 > 140.247.178.194.37612: Flags
[S.], seq 3684023987, ack 108294393, win 28960, options [mss
1460,sackOK,TS val 741439636 ecr 4217466076,nop,wscale 7], length 0

16:04:55.590298 IP 140.247.178.194.37612 > 4.53.56.118.80: Flags
[.], ack 1, win 115, options [nop,nop,TS val 4217466076 ecr
741439636], length 0
```

Web session (data transmission)

```
16:05:00.023831 IP 140.247.178.194.37612 > 4.53.56.118.80: Flags
[P.], seq 1:17, ack 1, win 115, options [nop,nop,TS val 4217467185
ecr 741439636], length 16

16:05:00.024328 IP 4.53.56.118.80 > 140.247.178.194.37612: Flags
[.], ack 17, win 227, options [nop,nop,TS val 741444070 ecr
4217467185], length 0

16:05:00.163811 IP 140.247.178.194.37612 > 4.53.56.118.80: Flags
[P.], seq 17:18, ack 1, win 115, options [nop,nop,TS val 4217467220
ecr 741444070], length 1

16:05:00.164245 IP 4.53.56.118.80 > 140.247.178.194.37612: Flags
[.], ack 18, win 227, options [nop,nop,TS val 741444210 ecr
4217467220], length 0

16:05:00.234189 IP 4.53.56.118.80 > 140.247.178.194.37612: Flags
[P.], seq 1:626, ack 18, win 227, options [nop,nop,TS val 741444280
ecr 4217467220], length 625

16:05:00.234195 IP 140.247.178.194.37612 > 4.53.56.118.80: Flags
[.], ack 626, win 124, options [nop,nop,TS val 4217467237 ecr
741444280], length 0
```

Web session (termination)

16:05:00.234356 IP 4.53.56.118.80 > 140.247.178.194.37612: Flags [**F**.], seq 626, ack 18, win 227, options [nop,nop,TS val 741444280 ecr 4217467220], length 0

16:05:00.234401 IP 140.247.178.194.37612 > 4.53.56.118.80: Flags [**F**.], seq 18, **ack** 627, win 124, options [nop,nop,TS val 4217467237 ecr 741444280], length 0

16:05:00.234958 IP 4.53.56.118.80 > 140.247.178.194.37612: Flags [.], **ack** 19, win 227, options [nop,nop,TS val 741444281 ecr 4217467237], length 0

Security vulnerability: plaintext passwords

Without encryption, passwords are trivial to recover with sniffing.

Large problem for custom web apps and mobile applications

- Less of a concern for common protocols these days

Several protocols still in widespread use are susceptible to this: HTTP, POP3, SNMP, and FTP

Example: ftp

```
# tcpdump -X -n tcp port 21
```

```
16:21:33.236417 IP 140.247.178.194.50384 > 69.163.224.12.21: Flags [P.],
seq 1:14, ack 27, win 115, length 13
    0x0000:  4510 0035 87c0 4000 4006 4d89 8cf7 b2c2  E..5..@.@.M.....
    0x0010:  45a3 e00c c4d0 0015 5af2 b588 99e8 a61b  E.......Z.......
    0x0020:  5018 0073 6591 0000 5553 4552 2077 616c  P..se...USER.wal
    0x0030:  7465 720d 0a                             ter..
```

```
16:21:35.252626 IP 140.247.178.194.50384 > 69.163.224.12.21: Flags
[P.], seq 14:27, ack 61, win 115, length 13
    0x0000:  4510 0035 87c2 4000 4006 4d87 8cf7 b2c2  E..5..@.@.M.....
    0x0010:  45a3 e00c c4d0 0015 5af2 b595 99e8 a63d  E.......Z......=
    0x0020:  5018 0073 6591 0000 5041 5353 2066 6f6f  P..se...PASS.foo
    0x0030:  6261 720d 0a                             bar..
```

Security vulnerability: online behavior profiling

- Even if passwords are encrypted, you can learn a lot about a user's behavior
    - Blackmail
    - Research for future social engineering attack against the user
- Learn about network topology
    - Servers (internal / external)
    - Clients (low-hanging fruit)

Wireshark

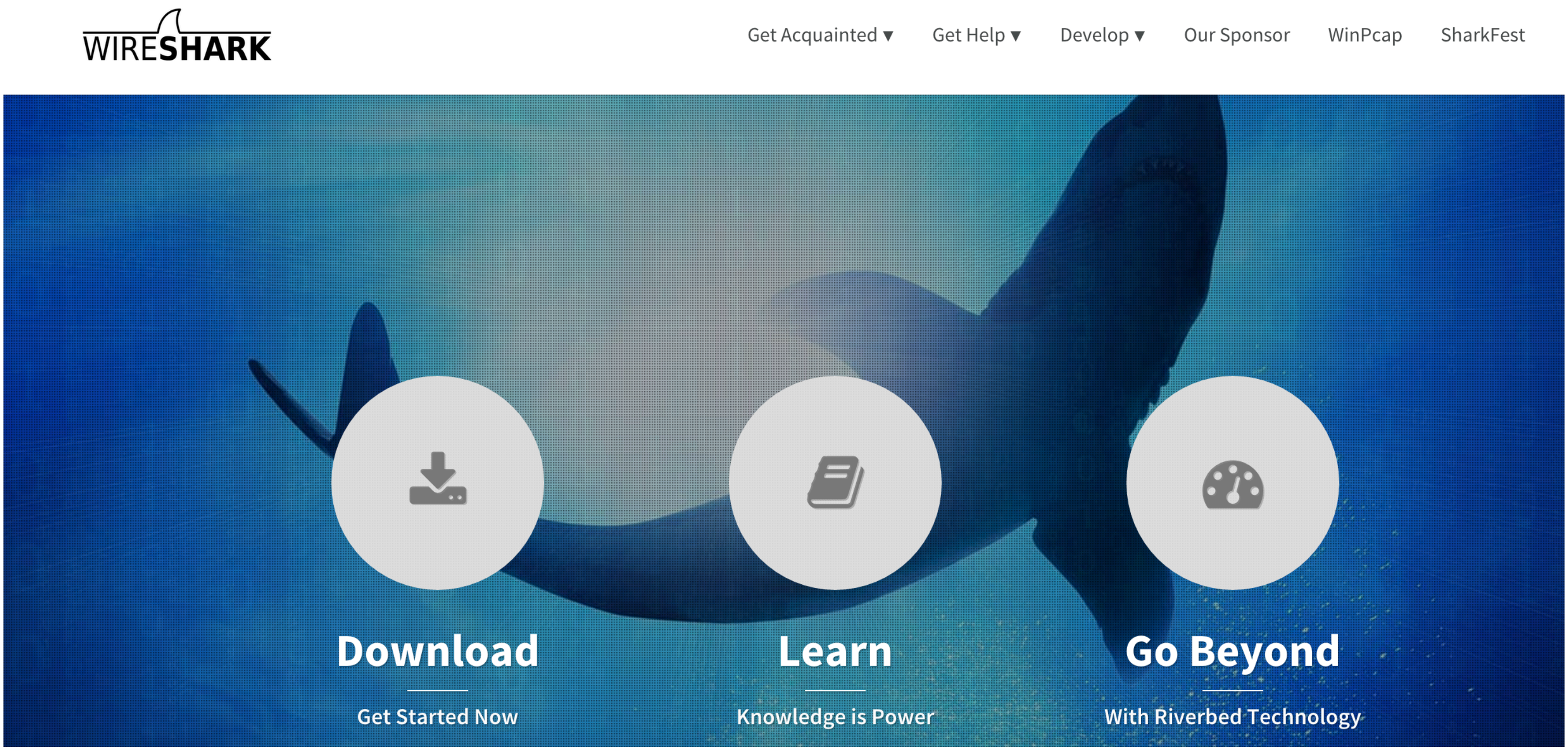
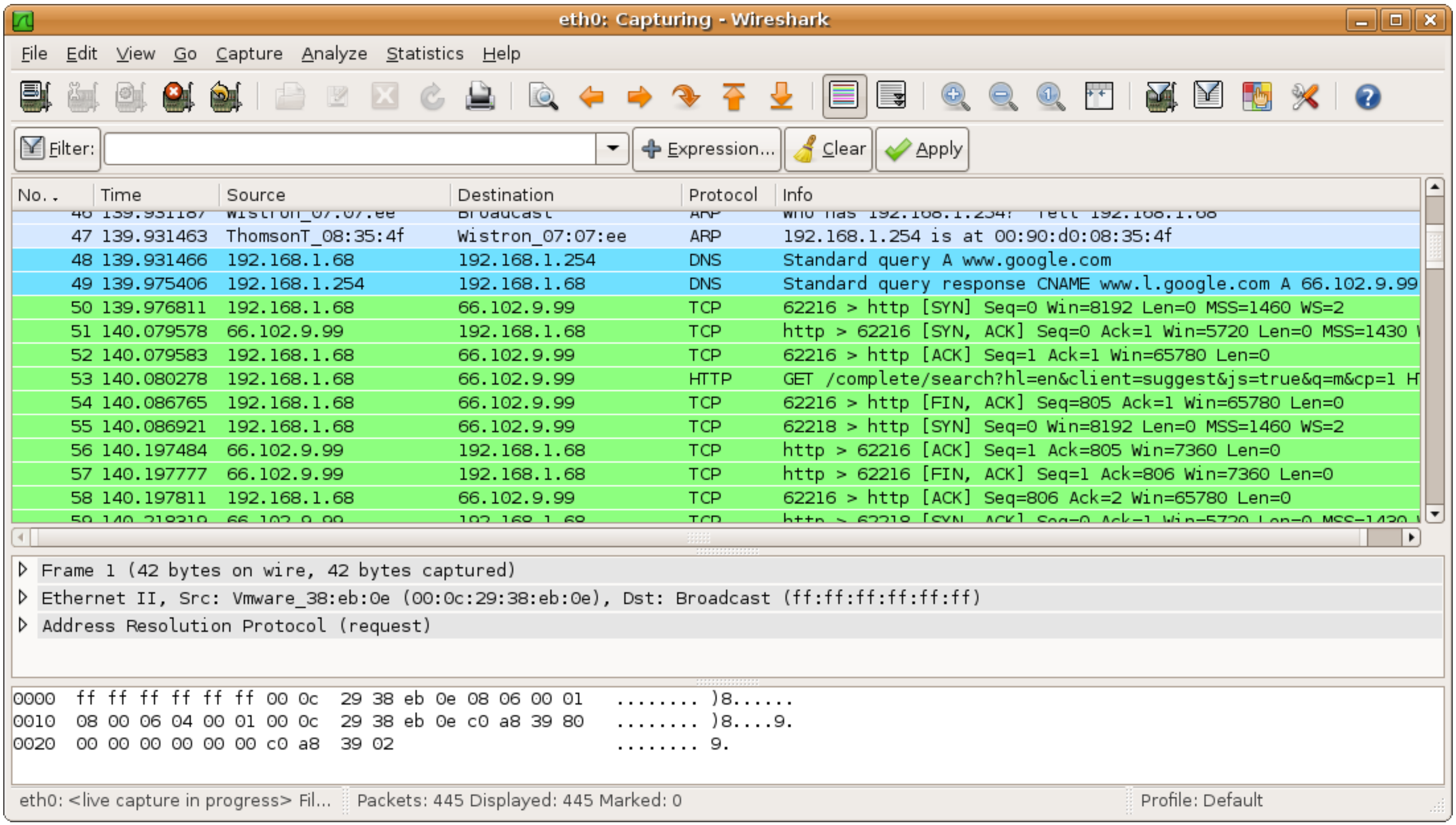


Image credit: https://www.wireshark.org

User-friendly front-end for packet sniffing

Wireshark


eth0: Capturing - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression...
Clear
Apply
No. Time Source Destination Protocol Info
47 139.931463 ThomsonT_08:35:4f Wistron_07:07:ee ARP 192.168.1.254 is at 00:90:d0:08:35:4f
48 139.931466 192.168.1.68 192.168.1.254 DNS Standard query A www.google.com
49 139.975406 192.168.1.254 192.168.1.68 DNS Standard query response CNAME www.l.google.com A 66.102.9.99
50 139.976811 192.168.1.68 66.102.9.99 TCP 62216 > http [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=2
51 140.079578 66.102.9.99 192.168.1.68 TCP http > 62216 [SYN, ACK] Seq=0 Ack=1 Win=5720 Len=0 MSS=1430
52 140.079583 192.168.1.68 66.102.9.99 TCP 62216 > http [ACK] Seq=1 Ack=1 Win=65780 Len=0
53 140.080278 192.168.1.68 66.102.9.99 HTTP GET /complete/search?hl=en&client=suggest&js=true&q=m&cp=1 H
54 140.086765 192.168.1.68 66.102.9.99 TCP 62216 > http [FIN, ACK] Seq=805 Ack=1 Win=65780 Len=0
55 140.086921 192.168.1.68 66.102.9.99 TCP 62218 > http [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=2
56 140.197484 66.102.9.99 192.168.1.68 TCP http > 62216 [ACK] Seq=1 Ack=805 Win=7360 Len=0
57 140.197777 66.102.9.99 192.168.1.68 TCP http > 62216 [FIN, ACK] Seq=1 Ack=806 Win=7360 Len=0
58 140.197811 192.168.1.68 66.102.9.99 TCP 62216 > http [ACK] Seq=806 Ack=2 Win=65780 Len=0
Frame 1 (42 bytes on wire, 42 bytes captured)
Ethernet II, Src: Vmware_38:eb:0e (00:0c:29:38:eb:0e), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Address Resolution Protocol (request)
0000 ff ff ff ff ff ff 00 0c 29 38 eb 0e 08 06 00 01 ....)8......
0010 08 00 06 04 00 01 00 0c 29 38 eb 0e c0 a8 39 80 ....)8....9.
0020 00 00 00 00 00 00 c0 a8 39 02 .... 9.
eth0: <live capture in progress> Fil...
Packets: 445 Displayed: 445 Marked: 0
Profile: Default

PCAP Interface

`libpcap` in Unix; `WinPcap` in Windows

C API, with wrapper support for many other languages:

- Python
- Ruby
- Rust
- Java
- C#
- Go

Powering many tools:

`tcpdump`

`libpcap` (device handling)

```
/* Define the device */
dev = pcap_lookupdev(errbuf);
if (dev == NULL) {
    fprintf(stderr, "Couldn't find default device: %s\n",
            errbuf);
    return(2);
}

/* Find the properties for the device */
if (pcap_lookupnet(dev, &net, &mask, errbuf) == -1) {
    fprintf(stderr, "Couldn't get netmask for device %s:
            %s\n", dev, errbuf);
    net = 0;
    mask = 0;
}
```

http://www.tcpdump.org/pcap.html

`libpcap` (session setup)

```
/* Open the session in promiscuous mode */
handle = pcap_open_live(dev, BUFSIZ, 1, 1000, errbuf);
if (handle == NULL) {
    fprintf(stderr, "Couldn't open device %s: %s\n", dev,
            errbuf);
    return(2);
}

/* Compile and apply the filter */
if (pcap_compile(handle, &fp, filter_exp, 0, net) == -1) {
    fprintf(stderr, "Couldn't parse filter %s: %s\n",
            filter_exp, pcap_geterr(handle));
    return(2);
}

if (pcap_setfilter(handle, &fp) == -1) {
    fprintf(stderr, "Couldn't install filter %s: %s\n",
            filter_exp, pcap_geterr(handle));
    return(2);
}
```

`libpcap` (capture and close)

```
/* Grab a packet */
packet = pcap_next(handle, &header);

/* Print its length */
printf("Captured a packet with length of
       [%d]\n", header.len);

/* And close the session */
pcap_close(handle);
return(0);
```