CEIC 2011

TCP/IP Protocol Analysis

Gary C. Kessler
Vermont Internet Crimes Against Children Task Force
Gary Kessler Associates/Norwich University

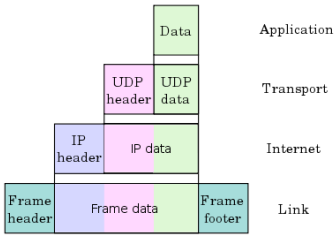
Overview

CEIC 2011

- The TCP/IP Protocol Suite
- Encapsulation and Interpretation
- The Role of Packet Sniffers
  - tcpdump/WinDump
  - Ethereal/WireShark
  - ngrep

CEIC 2011

TCP/IP Protocols, Encapsulation, and Interpretation



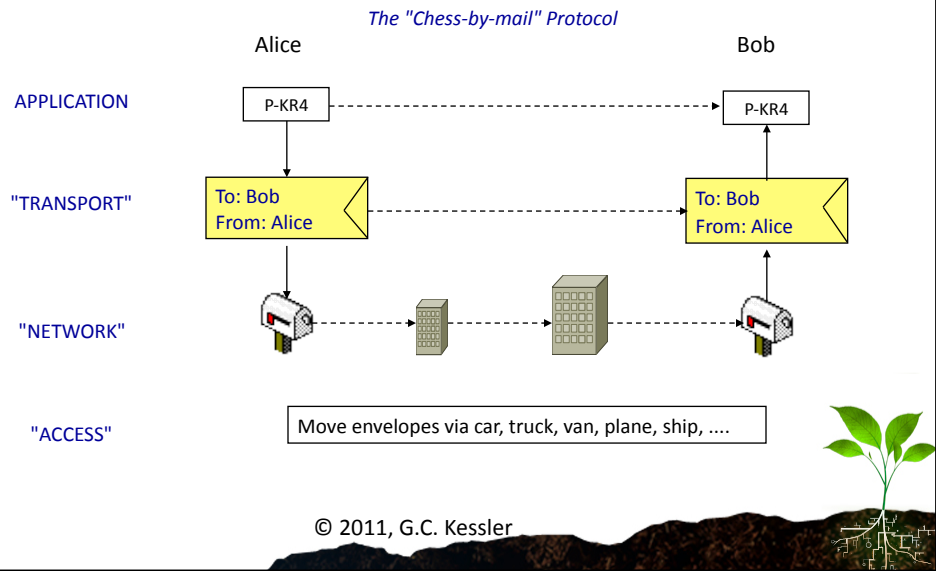
The Internet and TCP/IP

CEIC 2011

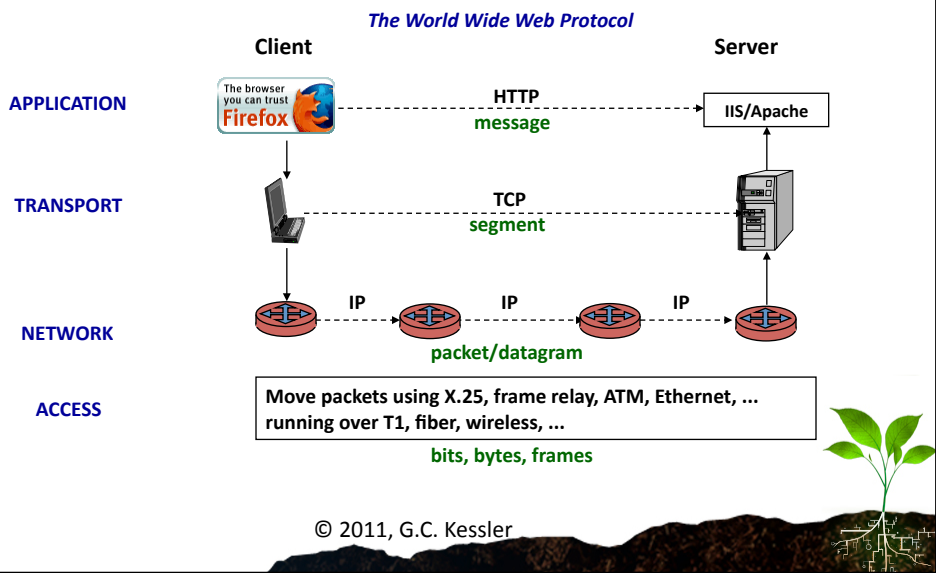
- TCP/IP is
  - The communications protocol suite that holds the Internet together
  - Non-proprietary; supported by all vendors on all software platforms
  - *"We reject kings, presidents, and voting. We believe in rough consensus and running code."* (D. Clark, about the IETF)
  - *The future protocol for voice and video??*

GCK, 1998

A Layered Protocol

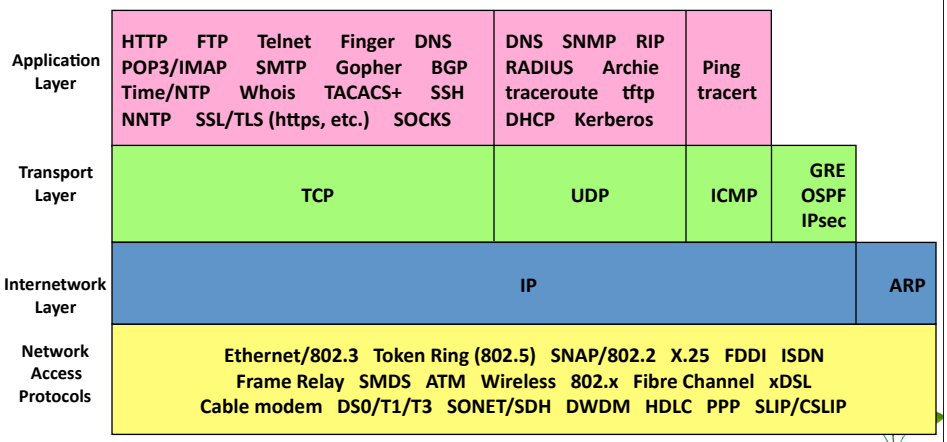


A Layered Protocol Architecture



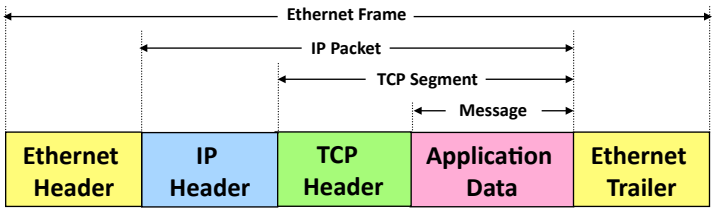
The TCP/IP Protocol Suite

CEIC 2011



How This All Fits Together

CEIC 2011

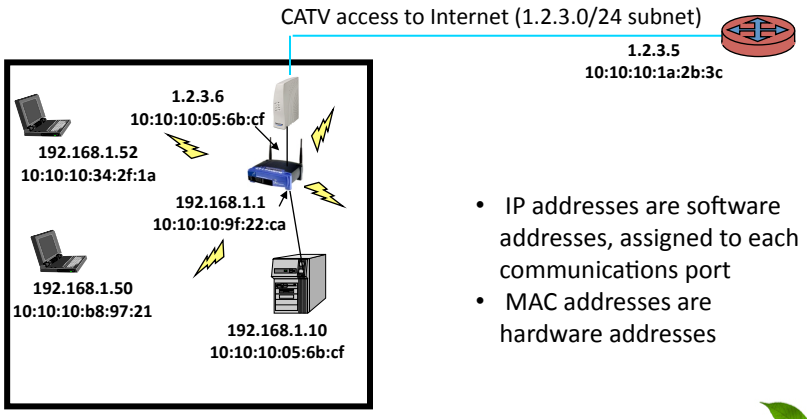


Hex Dump of a POP3 message:

```
00 90 27 8E A0 B2 00 AA 00 BB DE E8 08 00 45 00
00 35 2C 31 40 00 20 06 22 EC 83 6B 02 08 83 6B
02 C8 06 B0 00 6E 23 7D 3B 26 09 A5 90 F7 50 18
21 F6 67 E7 00 00 50 41 53 53 20 73 65 63 72 65
74 0D 0A A0 1B C2 3D
```

MAC and IP Addresses

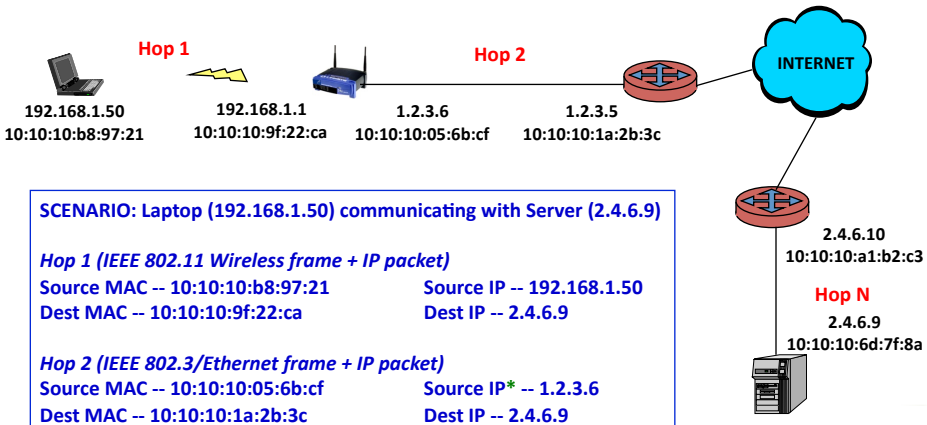
CEIC 2011



- IP addresses are software addresses, assigned to each communications port
- MAC addresses are hardware addresses

MAC and IP Addresses

CEIC 2011



SCENARIO: Laptop (192.168.1.50) communicating with Server (2.4.6.9)

Hop 1 (IEEE 802.11 Wireless frame + IP packet)
Source MAC -- 10:10:10:b8:97:21 Source IP -- 192.168.1.50
Dest MAC -- 10:10:10:9f:22:ca Dest IP -- 2.4.6.9

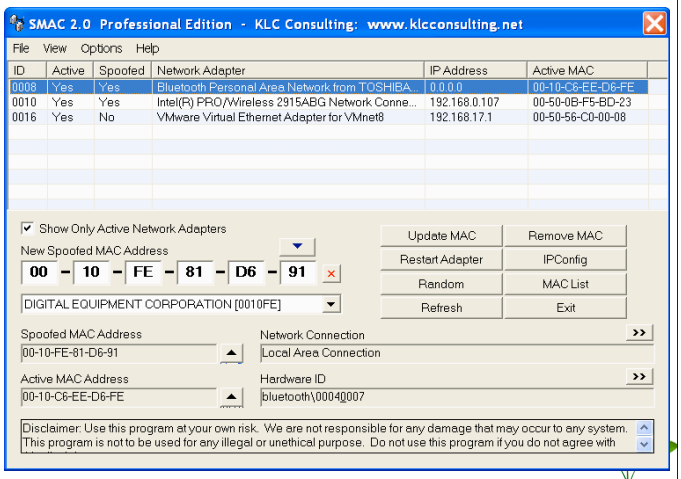
Hop 2 (IEEE 802.3/Ethernet frame + IP packet)
Source MAC -- 10:10:10:05:6b:cf Source IP* -- 1.2.3.6
Dest MAC -- 10:10:10:1a:2b:3c Dest IP -- 2.4.6.9
: (* Note that NAT has occurred)

Hop N (IEEE 802.3/Ethernet frame + IP packet)
Source MAC -- 10:10:10:a1:b2:c3 Source IP -- 1.2.3.6
Dest MAC -- 10:10:10:6d:7f:8a Dest IP -- 2.4.6.9

Sidenote: MAC Addresses

CEIC 2011

- MAC addresses are burned into the hardware
- But they be changed!!



Address Resolution Protocol

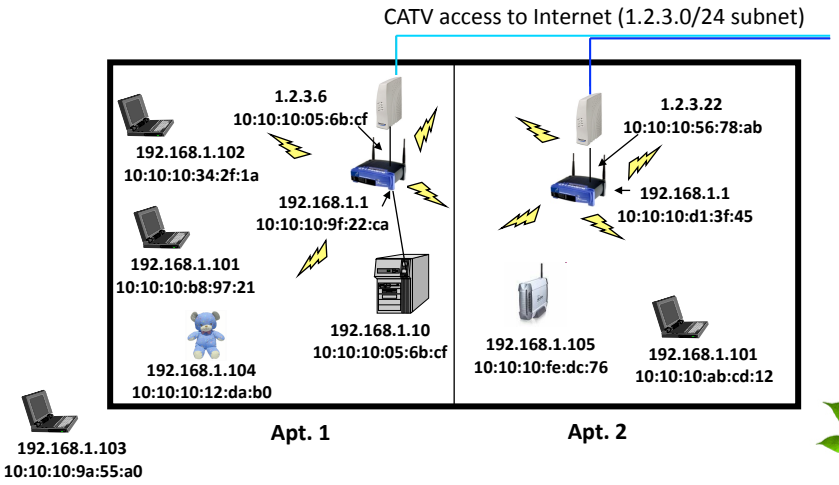
CEIC 2011

- ARP maps software (IP) addresses to hardware (MAC) addresses
- An IP packet on the local network
  - Contains the IP address of the ultimate destination
  - Is carried in a LAN protocol (e.g., Ethernet) frame addressed to the router's MAC address

```
C:\> arp -a
Interface: 192.168.1.100 --- 0x5
  Internet Address      Physical Address      Type
  192.168.1.1           55-55-55-6f-0d-87     dynamic
  192.168.1.40          55-55-55-b2-db-d8     dynamic
  192.168.1.50          55-55-55-70-67-8f     dynamic
  192.168.1.103         55-55-55-6c-01-de     dynamic
C:\>
```

A Common Scenario

CEIC 2011



Packet Decode (Ethernet Frame)

CEIC 2011

Hex	Binary		
00	0000 0000	Destination MAC address (OUI = 0x00-90-27 = Intel)	HEADER
90	1001 0000		
27	0010 0111		
8E	1000 1110		
A0	1010 0000		
B2	1011 0010		
00	0000 0000	Source MAC address (OUI = 0x00-AA-00 = Intel)	
AA	1010 1010		
00	0000 0000		
BB	1011 1011		
DE	1101 1110		
E8	1110 1000		
08	0000 1000	EtherType (IP)	
00	0000 0000		
A0	1010 0000	Frame Check Sequence	TRAILER
1B	0001 1011		
C2	1100 0010		
3D	0011 1101		

IEEE OUI and Company_id Assignments
http://standards.ieee.org/regauth/oui/

IANA Protocol Numbers
http://www.iana.org/assignments/ethernet-numbers

IP Packet Format

CEIC 2011

Bit Number

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

Version	IHL	Type of Service	Total Length
Identification		Flags	Fragment Offset
Time to Live	Protocol		Header Checksum
Source Address			
Destination Address			
Options (optional)			
Higher layer protocol data			

Ref: RFC 791 (STD 5)

Packet Decode (IP Packet)

CEIC 2011

```
Hex  Binary
 45  0100 ....  IP version number (4)
     .... 0101  IP header length (5 32-bit words/20 bytes)
 00  000. ....  Precedence (Routine)
     ...0 ....  Delay (Normal)
     .... 0...  Throughput (Normal)
     .... .0..  Reliability (Normal)
     .... ..0.  Cost (Normal)
     .... ...0  Reserved
 00  0000 0000
 35  0011 0101  Total packet length (53 bytes)
 2C  0010 1100
 31  0011 0001  Packet identifier (11313)
 40  0... ....  Reserved
     .1.. ....  Don't Fragment flag (don't fragment)
     ..0. ....  More Fragments flag (last fragment)
     ...0 0000
 00  0000 0000  Fragment offset (0)
```

Packet Decode (IP Packet, con't.)

CEIC 2011

```
Hex  Binary
20   0010 0000   Time to live (32)
06   0000 0110   Protocol (TCP)
22   0010 0010
EC   1110 1100   Header checksum
83   1000 0011
6B   0110 1011
02   0000 0010
08   0000 1000   Source address (131.107.2.8)
83   1000 0011
6B   0110 1011
02   0000 0010
C8   1100 1000   Destination address (131.107.2.200)
```


Ports

CEIC 2011

Port No.	Protocol	Application	Port No.	Protocol	Application
7	UDP	echo	80	TCP	http
13	TCP	daytime	110	TCP	pop3
19	UDP	chargen	111	TCP	sunrpc
20	TCP	ftp-data	113	TCP	auth
21	TCP	ftp-control	119	TCP	nntp
22	TCP	ssh	123	UDP	ntp
23	TCP	telnet	137	UDP	netbios-ns
25	TCP	smtp	138	UDP	netbios-dgm
37	UDP	time	139	TCP	netbios-ssn
43	TCP	whois	143	TCP	imap
53	TCP/UDP	dns	161	UDP	snmp
67	UDP	bootps	162	UDP	snmp-trap
68	UDP	bootpc	179	TCP	bgp
69	UDP	tftp	443	TCP	https (http/ssl)
70	TCP	gopher	514	UDP	syslog
79	TCP	finger	520	UDP	rip

TCP Segment Format

CEIC 2011

Bit Number

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

Source Port			Destination Port
Sequence Number			
Acknowledgment Number			
Offset	(reserved)	Flags	Window
Checksum			Urgent Pointer
Options (optional)			
Higher layer protocol data			

Ref: RFC 793 (STD 7)

Packet Decode (TCP Segment)

CEIC 2011

```
Hex  Binary
06   0000 0110
B0   1011 0000  Source port (1712)
00   0000 0000
6E   0110 1110  Destination port (110/POP3)
23   0010 0011
7D   0111 1101
3B   0011 1011
26   0010 0110  Sequence number (595409702)
09   0000 1001
A5   1010 0101
90   1001 0000
F7   1111 0111  Acknowledgement number (161845495)
50   0101 ....  Data offset (5 32-bit words/20 bytes)
     .... 0000  Reserved
```

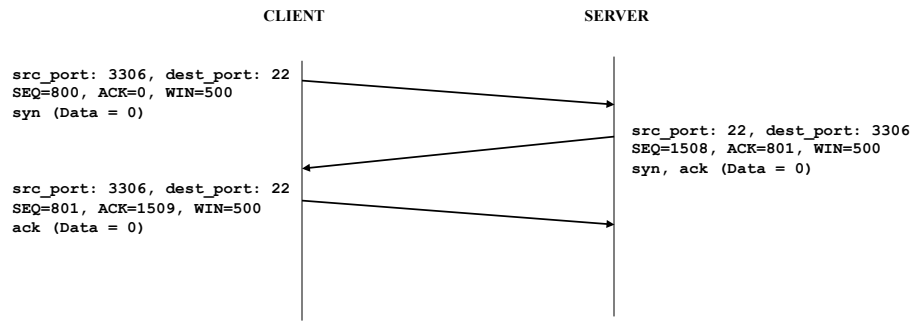
Packet Decode (TCP Segment, con't.)

CEIC 2011

Hex	Binary	
18	00..	Reserved
	..0.	Urgent pointer significant (no)
	...1	Acknowledgement field significant (yes)
	.... 1...	Push data (yes)
	.... .0..	Reset (no)
	.... ..0.	Synchronize sequence numbers (no)
	0	Finish flag (no)
21	0010 0001	
F6	1111 0110	Window size (8694 bytes)
C7	1100 0111	
E7	1110 0111	Checksum
00	0000 0000	
00	0000 0000	Urgent pointer (0)

Normal TCP Connection

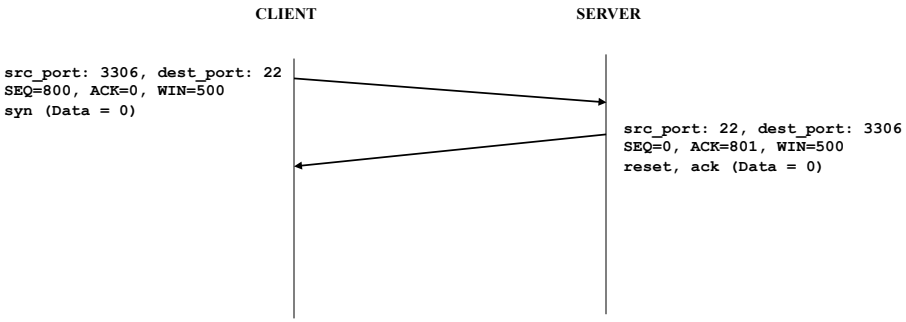
CEIC 2011



CLIENT attempts to connect to SERVER port 22 (SSH). SERVER has an SSH daemon so it is listening on port 22.

SEQ = sequence #	syn = synchronization flag
ACK = acknowledgement #	ack = acknowledgement-valid flag
WIN = Window size	reset = connection-reset flag

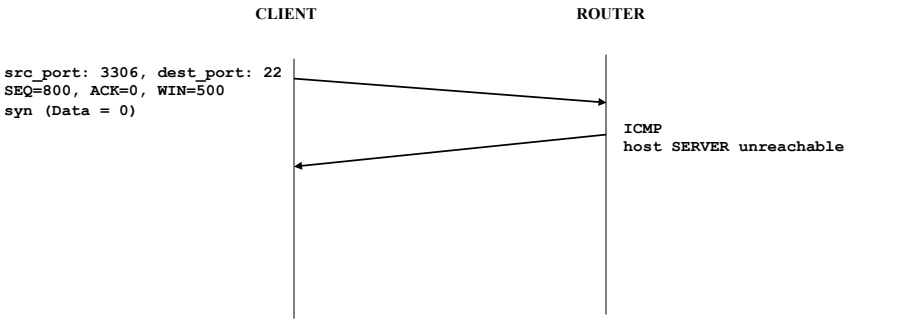
Host Not Listening on TCP Port



CLIENT attempts to connect to SERVER port 22 (SSH). SERVER does not have an SSH daemon, so it is not listening on port 22.

SEQ = sequence #	syn = synchronization flag
ACK = acknowledgement #	ack = acknowledgement-valid flag
WIN = Window size	reset = connection-reset flag

Host Does Not Exist



CLIENT attempts to connect to SERVER port 22 (SSH). SERVER is, for some reason, not "on" the network so its IP address does not exist on any network.

SEQ = sequence #	syn = synchronization flag
ACK = acknowledgement #	ack = acknowledgement-valid flag
WIN = Window size	reset = connection-reset flag

UDP Datagram Format

CEIC 2011

Bit Number

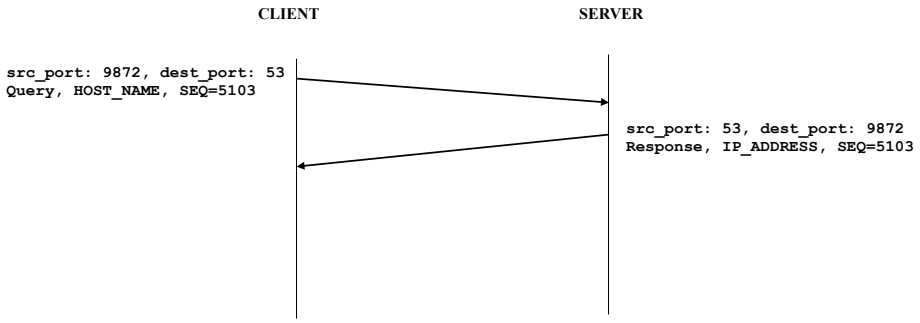
1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

Source Port	Destination Port
Length	Checksum
Higher layer protocol data	

Ref: RFC 768 (STD 6)

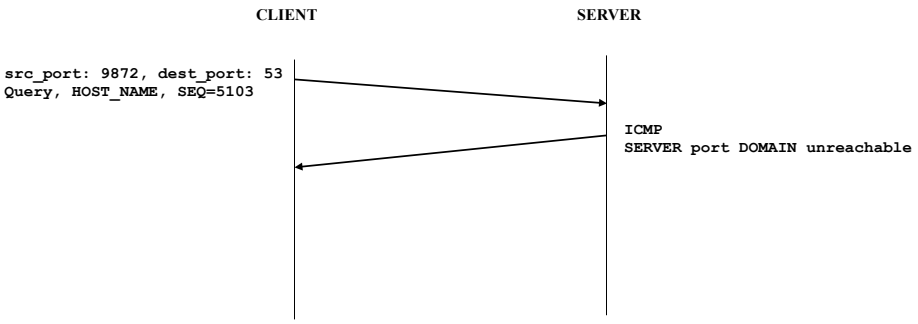
Normal UDP Communication

CEIC 2011



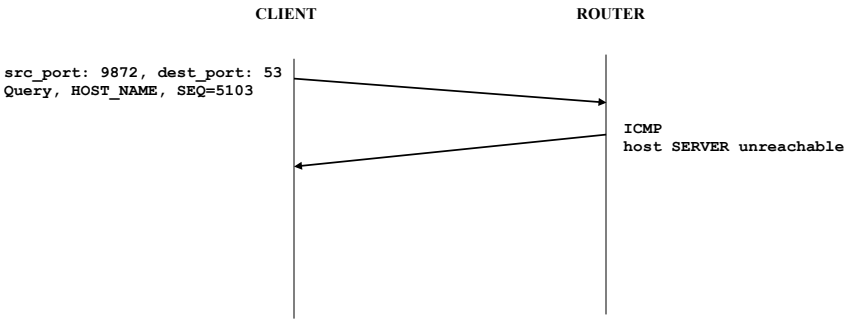
CLIENT attempts to connect to SERVER port 53 (DNS). SERVER runs BIND so it responds to the name query.

Host Not Listening on UDP Port



CLIENT attempts to connect to SERVER port 53 (DNS). SERVER does not run BIND, so it is not listening on port 53.

Host Does Not Exist



CLIENT attempts to connect to SERVER port 53 (DNS). SERVER is, for some reason, not "on" the network so its IP address does not exist on any network.

Packet Decode (POP3 Message)

CEIC 2011

Hex	Binary	
50	0101 0000	ASCII "P"
41	0100 0001	ASCII "A"
53	0101 0011	ASCII "S"
53	0101 0011	ASCII "S"
20	0010 0000	ASCII " "
73	0111 0011	ASCII "s"
65	0110 0101	ASCII "e"
63	0110 0011	ASCII "c"
72	0111 0010	ASCII "r"
65	0110 0101	ASCII "e"
74	0111 0100	ASCII "t"
0D	0000 1101	ASCII <CR>
0A	0000 1010	ASCII <LF>

ICMP Message Format

CEIC 2011

Bit Number

```
                1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
```

Type	Code	Checksum
Message-specific information.....		

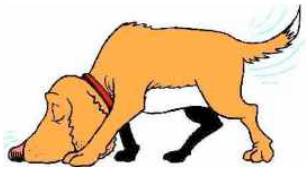
TYPE (CODES)

```
 0  Echo Reply
 3  Destination Unreachable (0=net, 1=host, 2=protocol, 3=port)
 4  Source Quench
 5  Redirect (0=network redirect, 1=host redirect)
 8  Echo Request
11  Time Exceeded (0=TTL exceeded, 1=fragment reassembly time exceeded)
12  Parameter Problem
13  Timestamp
14  Timestamp Reply
17  Address Mask Request
18  Address Mask Reply
```

Ref: RFC 792 (STD 5)

CEIC 2011

Packet Sniffers



Sources of Network Data

CEIC 2011

- IDS and firewall logs
- Server logs
- Network application logs
- Artifacts and remnants of traffic on computer hard drive
- Live traffic using a packet sniffer

What Are Packet Sniffers?

CEIC 2011

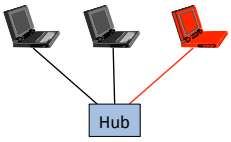
- Packet sniffers are hardware devices or software applications that read network traffic
  - Passive (and difficult to detect if used surreptitiously)
  - Can capture all broadcast traffic
    - Particular threat on wireless or hubbed networks
    - With ARP spoofing, can capture switched LAN traffic
- Most common software sniffers use libpcap (Linux/Unix) or WinPcap (Windows)
  - CLI: tcpdump, WinDump, ngrep
  - GUI: WireShark, Ethereal, Analyzer, Iris

Caution When Using a

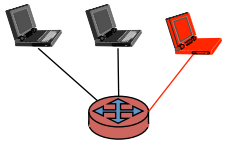
CEIC 2011

- Packet sniffers work best when they can "see" all traffic on the network

BUS - Broadcast



HUB - Broadcast



ROUTER/SWITCH – Administrative Port

WLAN – Promiscuous WLAN NIC

Installation

CEIC 2011

- See *http://www.garykessler.net/download/tcpip/sniffers.html*
  - **Install libpcap/WinPcap first!**
  - Installation instructions can be found at the sites referenced above

tcpdump and WinDump

CEIC 2011

- tcpdump is a libpcap-based packet sniffer
  - Runs on any Linux/Unix system with a NIC that can operate in promiscuous mode
  - Comes standard on many Linux distributions
- WinDump is a WinPcap-based sniffer
  - Windows port of tcpdump
- Both are command line utilities

TCP Handshake & Data Exchange

CEIC 2011

```
1.  14:05:27.083238 altamont.sover.net.1057 > ftp.ex.edu.21:
     S 1484414:1484414(0) win 8192 <mss 536,nop,nop,sackOK> (DF)
2.  14:05:27.250587 ftp.ex.edu.21 > altamont.sover.net.1057:
     S 3037133697:3037133697(0) ack 1484415 win 9112 <mss 536>
     (DF)
3.  14:05:27.259810 altamont.sover.net.1057 > ftp.ex.edu.21:
     . ack 1 win 8576 (DF)
4.  14:05:27.564336 ftp.ex.edu.21 > altamont.sover.net.1057:
     P 1:88(87) ack 1 win 9112 (DF) [tos 0x10]
5.  14:05:27.727461 altamont.sover.net.1057 > ftp.ex.edu.21:
     . ack 88 win 8489 (DF)
```


Handshake, Part 1

CEIC 2011

```
1.  14:05:27.083238 altamont.sover.net.1057 > ftp.ex.edu.21:
     S 1484414:1484414(0) win 8192 <mss 536,nop,nop,sackOK> (DF)
```

- Timestamp: `14:05:27.083238`
- Source IP and Port: `altamont.sover.net.1057` (IP or hostname)
- Destination IP and Port: `ftp.ex.edu.21` (IP or hostname)
- Flags: `S` (SYN)
- Initial Sequence Number and Bytes of Data in Payload: `1484414:1484414(0)` (setting sequence number for byte counting by client)
- Window Size in Bytes: `win 8192`
- TCP Options:
  - Maximum Segment Size in Bytes: `mss 536`
  - No Option Byte Pads: `nop,nop`
  - Selective Acknowledgment Allowed: `sackOK`
- Don't Fragment Bit Present: `(DF)`

Handshake, Part 2

CEIC 2011

```
2.  14:05:27.250587 ftp.ex.edu.21 > altamont.sover.net.1057:
     S 3037133697:3037133697(0) ack 1484415 win 9112 <mss 536> (DF)
```

- Source IP and Port: `ftp.ex.edu.21`
- Destination IP and Port: `ftp.client.edu.1057`
- Flags: `S` (SYN) and `ack`
  - **`ack 1484415`** means ftp.ex.edu expects the next data byte from ftp.client.edu to be numbered 1484415
  - **`1484415`** is altamont.sover.net's initial synchronization number 1484414 + 1
- Initial Sequence Number and Bytes in Data Payload: `3037133697:3037133697(0)`
- Window Size in Bytes: `win 9112`
- TCP Options:
  - Maximum Segment Size in Bytes: `mss 536`
- Don't Fragment Bit Present: `(DF)`

Handshake, Part 3

CEIC 2011

```
3.  14:05:27.259810 altamont.sover.net.1057 > ftp.ex.edu.21:
     . ack 1 win 8576 (DF)
```

- Flags: `. ack 1`
  - **`.`** (dot) means neither the SYN, FIN, RST, nor PSH flags are set
  - **`ack`** means that the ACK flag is set
  - **`1`** is a "relative sequence number" for easier reading, so **`ack 1`** is equivalent to **`ack 3037133698`**
  - 3037133698 is ftp.ex.edu's initial sequence number 3037133697 + 1
  - Although not a single byte of payload data has been passed from the altamont.sover.net to ftp.ex.edu, the client's sequence numbers have incremented from 3037133697 to 3037133698
- TCP Window Size in Bytes: `win 8576`
- Don't Fragment Bit Present: `(DF)`

Data Exchange

CEIC 2011

```
4.  14:05:27.564336 ftp.ex.edu.21 > altamont.sover.net.1057:
     P 1:88(87) ack 1 win 9112 (DF) [tos 0x10]
5.  14:05:27.727461 altamont.sover.net.1057 > ftp.ex.edu.21:
     . ack 88 win 8489 (DF)
```

- Packet 4: `P 1:88(87) ack 1`
  - **`P`** is PSH (Push) flag
  - **`1:88(87)`** means that there are 87 bytes of data in the packet starting at sequence number 1; byte 88 will be sent next
  - **`ack 1:`** ftp.ex.edu expects the next byte from altamont.sover.net to be number 1
  - **`[tos 0x10]`** denotes IP Type of Service is "minimize delay"
- Packet 5: `. ack 88`
  - **`.`** (dot) means neither the SYN, FIN, RST, nor PSH flags are set
  - **`ack 88`** means altamont.sover.net expects byte 88 next from ftp.ex.edu, acknowledging receipt of bytes 1 to 87

TCP Connection Termination

CEIC 2011

Packets 6 to 12 deleted for brevity...

```
13.  14:05:35.774099 altamont.sover.net.1057 > ftp.ex.edu.21:
      P 31:37(6) ack 183 win 8394 (DF)
14.  14:05:35.895233 ftp.ex.edu.21 > altamont.sover.net.1057:
      P 183:197(14) ack 37 win 9112 (DF) [tos 0x10]
15.  14:05:35.903492 ftp.ex.edu.21 > altamont.sover.net.1057:
      F 197:197(0) ack 37 win 9112 (DF) [tos 0x10]
16.  14:05:35.906748 altamont.sover.net.1057 > ftp.ex.edu.21:
      . ack 198 win 8380 (DF)
17.  14:05:35.917049 altamont.sover.net.1057 > ftp.ex.edu.21:
      F 37:37(0) ack 198 win 8380 (DF)
18.  14:05:35.921478 ftp.ex.edu.21 > altamont.sover.net.1057:
      . ack 38 win 9112 (DF) [tos 0x10]
```

Last Exchange of Data

CEIC 2011

```
13.  14:05:35.774099 altamont.sover.net.1057 > ftp.ex.edu.21:
      P 31:37(6) ack 183 win 8394 (DF)
14.  14:05:35.895233 ftp.ex.edu.21 > altamont.sover.net.1057:
      P 183:197(14) ack 37 win 9112 (DF) [tos 0x10]
```

- Packet 13: `31:37(6) ack 183`
  - **`31:37(6):`** 6 bytes of data in packet starting at 31; byte 37 will be sent next
  - **`ack 183`:** altamont.sover.net next expecting byte 183 from ftp.ex.edu
- Packet 14: `183:197(14) ack 37`
  - **`183:197 (14):`** 14 bytes of data in packet starting at 183; byte 197 will be next
  - **`ack 37:`** ftp.ex.edu expects next byte from altamont.sover.net to be number 37

Graceful Shutdown

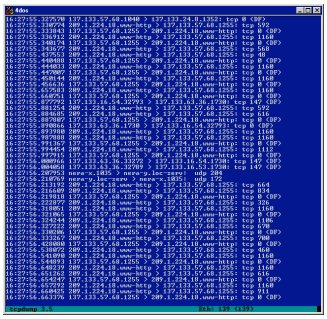
CEIC 2011

```
15.  14:05:35.903492 ftp.ex.edu.21 > altamont.sover.net.1057:
      F 197:197(0) ack 37 win 9112 (DF) [tos 0x10]
16.  14:05:35.906748 altamont.sover.net.1057 > ftp.ex.edu.21:
      . ack 198 win 8380 (DF)
17.  14:05:35.917049 altamont.sover.net.1057 > ftp.ex.edu.21:
      F 37:37(0) ack 198 win 8380 (DF)
18.  14:05:35.921478 ftp.ex.edu.21 > altamont.sover.net.1057:
      . ack 38 win 9112 (DF) [tos 0x10]
```

- Packet 15 from ftp.ex.edu has the FIN flag (F) set, meaning that there is no more data to send. This packet contains no data.
- Packet 16 is altamont's acknowledgement; the connection is now half-closed.
- Packet 17 from altamont has the FIN flag set to shut down this direction.
- Packet 18 is ftp.ex.edu's acknowledgement and the TCP connection is now closed.

CEIC 2011

tcpdump/WinDump Tutorial



WinDump Command Line

CEIC 2011

DOS box

```
C:\MYPROG~1\WinDump>windump -i 6
windump: listening on \Device\NPF_{14033A57-673E-4DE5-B6C9-95D6B09867DC}
15:15:17.818804 arp who-has 192.168.1.1 tell Altamont.sbtnvt.adelphia.net
15:15:17.819183 arp reply 192.168.1.1 is-at 00:06:25:dc:25:91
15:15:17.819195 IP Altamont.sbtnvt.adelphia.net.1029 > nscache1.bur.adelphia.net
.53:  26196+ A? www.cisco.com. (31)
15:15:17.832581 IP nscache1.bur.adelphia.net.53 > Altamont.sbtnvt.adelphia.net.1
029:  26196 1/2/2 A www.cisco.com (115)
15:15:17.837722 IP Altamont.sbtnvt.adelphia.net > www.cisco.com: icmp 40: echo r
equest seq 256
15:15:17.938038 IP www.cisco.com > Altamont.sbtnvt.adelphia.net: icmp 40: echo r
eply seq 256
15:15:18.711902 IP Altamont.sbtnvt.adelphia.net.1029 > nscache1.bur.adelphia.net
.53:  52052+ PTR? 1.1.168.192.in-addr.arpa. (42)
15:15:18.721700 IP nscache1.bur.adelphia.net.53 > Altamont.sbtnvt.adelphia.net.1
029:  52052 NXDomain 0/1/0 (119)
15:15:18.786885 IP Altamont.sbtnvt.adelphia.net.137 > 192.168.1.1.137: UDP, leng
th: 50
15:15:18.787019 IP ALTAMONT.137 > 192.168.1.1.137: UDP, length: 50
15:15:18.787120 IP ALTAMONT.137 > 192.168.1.1.137: UDP, length: 50
15:15:18.787551 IP 192.168.1.1 > Altamont.sbtnvt.adelphia.net: icmp 36: 192.168.
1.1 udp port 137 unreachable
15:15:18.787893 IP 192.168.1.1 > ALTAMONT: icmp 36: 192.168.1.1 udp port 137 unr
eachable
15:15:18.788252 IP 192.168.1.1 > ALTAMONT: icmp 36: 192.168.1.1 udp port 137 unr
```

Some tcpdump/WinDump Switches

CEIC 2011

`-D`	Obtain list of network interfaces
`-e`	Display link layer (Ethernet) header
`-F`	Filter display/capture
`-h`	Help
`-i`	Get input form specified interface
`-n`	Enable name resolution
`-r`	Get input from a file
`-s`	Snaplen (Frame display size)
`-w`	Send output to a file
`-x`	Hexadecimal output

Hexadecimal Output (-x)

CEIC 2011

```
gck# tcpdump -x
11:55:52.069484 192.168.143.5 >
  192.168.143.101: icmp: echo request
        4500 003c 064b 0000 4001 bc12 c0a8 8f05
        c0a8 8f65 0800 620a 850a 0000 6162 6364
          6566 6768 696a 6b6c 6d6e 6f70 7172 7374
          7576 7761 6263
```

IP header

ICMP Header

Protocol = 0x01 (ICMP)

Embedded protocol data: `abcdefghijklmnopqrstuvwabc`

(Note: These are from the 32 bytes sent by a Windows ping)

snaplen

CEIC 2011

- snaplen defines the number of bytes captured
  - Default = 68
  - But why do we see only 54 bytes of data?

```
4500 003c 064b 0000 4001 bc12 c0a8 8f05   (16)
c0a8 8f65 0800 620a 850a 0000 6162 6364   (32)
6566 6768 696a 6b6c 6d6e 6f70 7172 7374   (48)
7576 7761 6263                            (54)
```

- Answer: Ethernet header = 14B (not displayed)

Setting snaplen (-s)

CEIC 2011

- Ethernet maximum frame size = 1514 B

```
gck# tcpdump -x -s 1514
11:55:52.069484 192.168.143.5 >
  192.168.143.101: icmp: echo request
   4500 003c 064b 0000 4001 bc12 c0a8 8f05 (16)
   c0a8 8f65 0800 620a 850a 0000 6162 6364 (32)
   6566 6768 696a 6b6c 6d6e 6f70 7172 7374 (48)
   7576 7761 6263 6465 6667 6869           (60)
```

IP packet length = 0x3c = 60B

Embedded protocol data: `abcdefghijklmnopqrstuvwabcdefghi`

(Note: These are the entire 32 bytes sent by a Windows ping)

Displaying the Frame Header (-e)

CEIC 2011

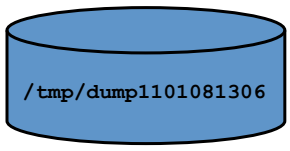
```
gck# tcpdump -e
20:55:48.520619 0:10:b5:39:c6:93
  0:10:b5:19:25:9a ip 74 192.168.143.5 >
  192.168.143.101: icmp: echo request
```

Source MAC Address (6B):	0:10:b5:39:c6:93
Destination MAC Address (6B):	0:10:b5:19:25:9a
Protocol Type (2B):	ip (0x0800)
Frame Length:	74B (calculated)

Writing/Reading Raw Data (-r, -w)

CEIC 2011

```
gck# tcpdump -w /tmp/dump1101081306
```




```
gck# tcpdump -r /tmp/dump1101081306
```

Filtering Data (-F)

CEIC 2011

Only display TCP traffic:

```
gck# tcpdump 'tcp'
```

```
gck\windump> windump "tcp"
```

Display traffic per the filter expression in file *tcp.filter*:

```
gck# tcpdump -F /home/gck/tcp.filter
```

Enabling Name/Port Resolution (-n)

CEIC 2011

```
gck# tcpdump
12:09:45.987469 27-5.example.edu.1105 >
  doggie.example.edu.ftp: S
  2175981658:2175981658(0) win 32120 <mss
  1460,sackOK,timestamp 4855695> (DF)
gck# tcpdump -n
12:10:32.404176 192.233.5.27.1106 >
  198.112.67.32.21: S 2222946726:2222946726
  (0) win 32120 <mss 1460,sackOK,timestamp
  4860337> (DF)
```

Changing Interfaces (-D, -i)

CEIC 2011

- tcpdump/WinDump will default to use the lowest numbered, currently available interface (not including the loopback interface)

In Unix/Linux:
- Use `ifconfig -a` or `tcpdump -D` to get interface information
- Use `tcpdump -i int_name` to use a particular interface
- E.g.: `tcpdump -i ppp0`

In Windows:
- Use `windump -D` to get interface information
- Use `windump -i int_number` to use a particular interface
- E.g.: `windump -i 2`

IP Header Length

CEIC 2011

IP Header Length: 5 → Five 32-bit words = 5*4 bytes = 20 bytes

```
4500 0054 064b 0000 4001 bc12 c0a8 8f05    (16)
c0a8 8f65 0800 620a 850a 0000 889f 4b39    (32)
510f 0100 0809 0a0b 0c0d 0e0f 1011 1213    (48)
1415 1617 1819                             (54)
```

IP Datagram Length

CEIC 2011

IP Datagram Length: 0x54 → $(5 * 16^1) + (4 * 16^0) = 80 + 4 = 84$ bytes

```
4500 0054 064b 0000 4001 bc12 c0a8 8f05    (16)
c0a8 8f65 0800 620a 850a 0000 889f 4b39    (32)
510f 0100 0809 0a0b 0c0d 0e0f 1011 1213    (48)
1415 1617 1819                             (54)
```

IP Options

CEIC 2011

```
20:55:48.520619 192.168.143.5 > 192.168.143.101:
  icmp: echo request
```

IP Header Length: 0xe = 14 → 14 32-bit words = 14*4 bytes = 56 bytes

IP option type: 0x44 = 68 = timestamp option

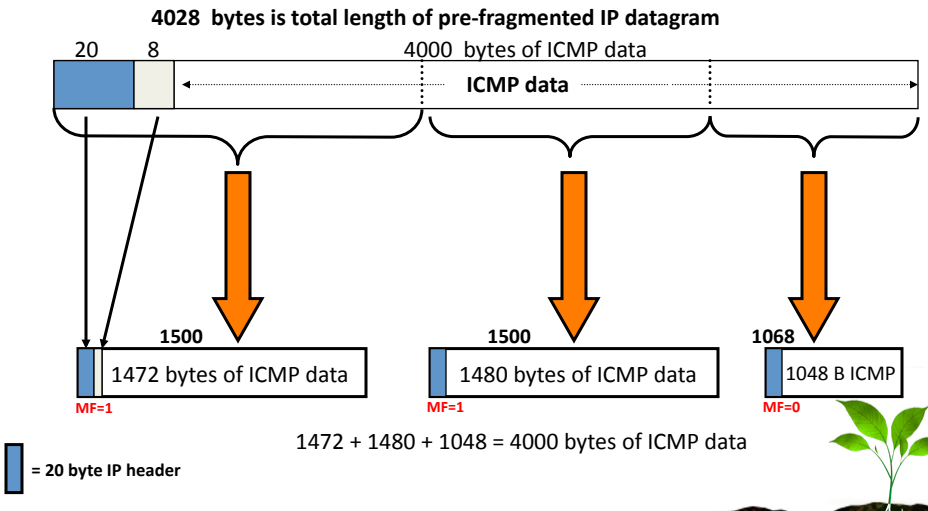
IP option length: *0x24* = 36 bytes

```
4e00 004c 05d9 0000 4001 d970 c0a8 8f05    (16)
c0a8 8f65 4424 0d03 c0a8 8f05 0033 1828    (32)
c0a8 8f65 0000 0000 c0a8 8f65 0000 0000    (48)
c0a8 8f05 0000 0000 0800 21a2 bd02 0100    (64)
0033 1828 0000 0000 0000 0000              (76)
```

Fragmentation

- Maximum IP datagram size = 65,535 (2^{16}-1)
- Maximum fragment offset = 8,191 (2^{13}-1)
- Fragment Offset field measures in units of 8 (2^3) bytes
- Fragment Identification field used to associate fragments for reassembly
- More Fragments flag indicates when reassembly is complete

Fragmentation Example: Echo



Fragmentation-Related Values

CEIC 2011

	---- FRAGMENT -----			ORIGINAL
	1	2	3	DATAGRAM
Fragment Ident.	21223	21223	21223	
Fragment Offset	0	1480	2960	0
Offset in 8B units	(0)	(185)	(370)	(0)
More Fragments flag	1	1	0	0
Packet length	1500	1500	1068	4028
IP Header info	20	20	20	20
ICMP Header info	8	0	0	8
ICMP data	1472	1480	1048	4000

tcpdump and Fragmentation

CEIC 2011

```
altamont.sover.net > foo.ex.edu: icmp: echo request (frag
21223:1480@0+)
altamont.sover.net> foo.ex.edu: (frag 21223:1480@1480+)
altamont.sover.net> foo.ex.edu: (frag 21223:1048@2960)
```

Fragment Identification

Number of bytes in fragment

Offset into the original datagram

More Fragments flag

tcpdump and Fragmentation (hex)

CEIC 2011

```
altamont.sover.net > foo.ex.edu: icmp: echo request (frag
21223:1480@0+)
   4500 05dc 52e7 2000 4001 a83d c0a8 8f05
   c0a8 8f65 0800 f827 2a05 0300 6162 6364
                    :
altamont.sover.net > foo.ex.edu: (frag 21223:1480@1480+)
   4500 05dc 52e7 20b9 4001 13f2 c0a8 8f05
   c0a8 8f65 7374 7576 7778 797a 7b7c 7d7e
                    :
altamont.sover.net > foo.ex.edu: (frag 21223:1048@2960)
   4500 042c 52e7 0172 4001 70f1 c0a8 8f05
   c0a8 8f65 6768 696a 6b6c 6d6e 6f70 7172
                    :
```


TCP Header Length

CEIC 2011

```
15:43:40.705372 altamont.sover.net.63220 >
  foo.ex.edu.netbios-ssn: S
  776342897:776342897(0) win 3072
   4500 0028 e34f 0000 3a06 e534 c0a8 8f05
   c0a8 8f65 f6f4 008b 2e46 0d71 0000 0000
   5002 0c00 b85f 0000
```

TCP header length = 5 * 4 = 20 bytes

Found in high-order nibble of 13th byte (byte #12) in the TCP header

TCP Header Length With Options

CEIC 2011

```
15:48:24.620314 verbo.3088 > win98.netbios-ssn: S
  1212214992:1212214992(0) win 32120 <mss
  1460,sackOK,timestamp 7748460 0,nop,wscale 0> (DF)
   4500 003c 11a8 4000 4006 70c8 c0a8 8f05
   c0a8 8f65 0c10 008b 4840 eed0 0000 0000
   a002 7d78 92b4 0000 0204 05b4 0402 080a
   0076 3b6c 0000 0000 0103 0300
```

TCP header length = 10 * 4 = 40 bytes
Options:
- 2 (MSS), length = 4 bytes, value = 1460 (0x5b4)
- 4 (SACK permitted), length = 2 bytes
- 8 (timestamp), length = 10 bytes, values = 7748460 (0x00763b6c), 0 (0x00000000)
- 1 (No Op.)
- 3 (Windows scale), length = 3 bytes, value = 0

Dissecting a UDP Packet

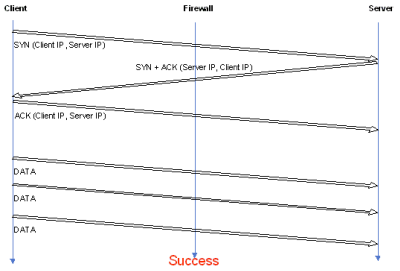
CEIC 2011

```
4500 0060 893c 0000 8011 f904 c0a8 8f05
c0a8 8f65 0089 0089 004c 1fd7 0d42 4000
0001 0000 0000 0001 2041 4241 4346 5046
5045 4e46 4445 4346 4345 5046 4846 4445
4646 5046 5041 4341 4200 0020 0001 c00c
0020 0001 0004 93e0 0006 8000 ccf0 8f65
```

IP Header Length (5*4) = 20 bytes
  Protocol Id. = 0x11 = 17 = UDP
UDP Header is 8 bytes in length
  Source port and *destination port* = 0x89 = 137 = netbios.name
  Datagram length = 0x4c = 76 bytes
Next 68 bytes are the contents of the UDP datagram (Ref. RFC 1002)

CEIC 2011

Analyzing Traffic



CEIC 2011

Suspicious tcpdump Output?

```
13:58:18.387461 foo.example.net.1565 > holmes.ftp: S 2115515674:2115515674(0)
   win 32120 <mss 1460,sackOK,timestamp 126466936 0,nop,wscale 0> (DF)
13:58:18.392988 holmes.ftp > foo.example.net.1565: S 3478333904:3478333904(0)
   ack 2115515675 win 10136 <nop,nop,timestamp 126470710 126466936,nop,wscale
   0,nop,nop,sackOK,mss 1460> (DF)
14:00:46.999055 foo.example.net.4238 > watson.ftp: S 2262117252:2262117252(0)
   win 32120 <mss 1460,sackOK,timestamp 126481796 0,nop,wscale 0> (DF)
14:00:47.029487 watson.ftp > foo.example.net.4238: S 26347543:26347543(0) ack
   2262117253 win 8760 <mss 1460,nop,nop,sackOK> (DF)
```

OS fingerprinting with TCP option set (Linux & Windows)

Suspicious tcpdump Output?

CEIC 2011

```
16:25:00.711083 holmes.39272 > doggie.example.edu.135: udp 0
16:25:00.711923 holmes.39272 > doggie.example.edu.140: udp 0
16:25:00.712543 holmes.39272 > doggie.example.edu.136: udp 0
16:25:00.712703 doggie.example.edu > holmes: icmp: doggie.example.edu udp port 140 unreachable
16:25:00.713223 doggie.example.edu > holmes: icmp: doggie.example.edu udp port 136 unreachable
16:25:00.714060 holmes.39272 > doggie.example.edu.netbios-dgm: udp 0
16:25:00.714763 holmes.39272 > doggie.example.edu.130: udp 0
16:25:00.715275 holmes.39272 > doggie.example.edu.netbios-ns: udp 0
16:25:00.715337 doggie.example.edu > holmes: icmp: doggie.example.edu udp port 130 unreachable
16:25:00.715822 holmes.39272 > doggie.example.edu.132: udp 0
16:25:00.716797 holmes.39272 > doggie.example.edu.netbios-ssn: udp 0
16:25:00.716858 doggie.example.edu > holmes: icmp: doggie.example.edu udp port 132 unreachable
16:25:00.717692 holmes.39272 > doggie.example.edu.134: udp 0
16:25:00.718276 holmes.39272 > doggie.example.edu.133: udp 0
16:25:00.718388 doggie.example.edu > holmes: icmp: doggie.example.edu udp port 134 unreachable
16:25:00.718895 doggie.example.edu > holmes: icmp: doggie.example.edu udp port 133 unreachable
16:25:00.802856 holmes.39272 > doggie.example.edu.131: udp 0
16:25:01.024999 holmes.39272 > doggie.example.edu.135: udp 0
16:25:01.025761 holmes.39272 > doggie.example.edu.netbios-dgm: udp 0
16:25:01.026355 holmes.39272 > doggie.example.edu.netbios-ns: udp 0
16:25:01.026930 holmes.39272 > doggie.example.edu.netbios-ssn: udp 0
16:25:01.028203 doggie.example.edu > holmes: icmp: doggie.example.edu udp port 131 unreachable
16:25:01.844952 holmes.39272 > doggie.example.edu.netbios-ssn: udp 0
16:25:01.845647 holmes.39272 > doggie.example.edu.netbios-ns: udp 0
16:25:01.846286 holmes.39272 > doggie.example.edu.netbios-dgm: udp 0
16:25:01.846932 holmes.39272 > doggie.example.edu.135: udp 0
```

UDP port scan

Suspicious tcpdump Output?

CEIC 2011

```
20:55:21.594616 holmes.1099 > watson.80: . ack 1 win 512
20:55:22.116776 holmes.1097 > watson.80: . ack 1 win 512
20:55:22.539044 holmes.1116 > watson.80: . ack 1 win 512
20:55:22.996164 holmes.1032 > watson.80: . ack 1 win 512
20:55:23.215656 holmes.1047 > watson.80: . ack 1 win 512
20:55:23.475400 holmes.1101 > watson.80: . ack 1 win 512
20:55:23.987302 holmes.1116 > watson.80: . ack 1 win 512
20:55:24.098277 holmes.1099 > watson.80: . ack 1 win 512
20:55:24.503761 holmes.1047 > watson.80: . ack 1 win 512
20:55:24.976001 holmes.1112 > watson.80: . ack 1 win 512
20:55:25.216777 holmes.1097 > watson.80: . ack 1 win 512
20:55:25.664981 holmes.1115 > watson.80: . ack 1 win 512
20:55:25.992876 holmes.1115 > watson.80: . ack 1 win 512
20:55:26.437888 holmes.1119 > watson.80: . ack 1 win 512
20:55:26.899912 holmes.1100 > watson.80: . ack 1 win 512
```

Odd set of source port numbers...

Possible Covert Communication

CEIC 2011

- Assume exploited Web server....
- Look at source port number
  - Subtract 1000 from source port value, convert remainder to ASCII value
  - Result: `cat /etc/passwd`
- Merely an example; who knows what it is? But this might raise some eyebrows...

Suspicious tcpdump Output?

CEIC 2011

```
13:21:45.010117 holmes.4033 > watson.220: S 93266:93266(0) win 8192
13:21:45.011128 holmes.4003 > watson.ftp: S 92918:92918(0) win 8192
13:21:45.012014 holmes.4005 > watson.telnet: S 92946:92946(0) win 8192
13:21:45.013095 holmes.4004 > watson.22: S 92932:92932(0) win 8192
13:21:45.014107 holmes.4019 > watson.110: S 93094:93094(0) win 8192
13:21:45.015865 holmes.4010 > watson.63: S 93016:93016(0) win 8192
13:21:45.016763 holmes.4021 > watson.nntp: S 93106:93106(0) win 8192
13:21:45.018001 holmes.4016 > watson.80: S 93076:93076(0) win 8192
13:21:45.018456 holmes.4017 > watson.92: S 93154:93154(0) win 8192
13:21:45.018997 holmes.4034 > watson.396: S 93280:93280(0) win 8192
13:21:45.019562 holmes.4031 > watson.215: S 93238:93238(0) win 8192
13:21:45.020017 holmes.4002 > watson.17: S 92912:92912(0) win 8192
```

TCP port scan

Suspicious tcpdump Output?

CEIC 2011

```
13:21:45.012014 foo.example.com.1090 > 198.112.67.27.80: S 92946:92946(0) win 8192
13:21:45.013095 foo.example.com.1092 > 198.112.67.28.80: S 92932:92932(0) win 8192
13:21:45.014107 foo.example.com.1093 > 198.112.67.29.80: S 93094:93094(0) win 8192
13:21:45.015865 foo.example.com.1095 > 198.112.67.30.80: S 93016:93016(0) win 8192
13:21:45.016763 foo.example.com.1096 > 198.112.67.31.80: S 93106:93106(0) win 8192
13:21:45.018001 foo.example.com.1097 > 198.112.67.32.80: S 93076:93076(0) win 8192
13:21:45.018456 foo.example.com.1100 > 198.112.67.33.80: S 93154:93154(0) win 8192
13:21:45.018997 foo.example.com.1102 > 198.112.67.34.80: S 93280:93280(0) win 8192
```

Port 80 site scan

Suspicious tcpdump Output?

CEIC 2011

```
foo.example.com.45820 > 192.168.209.5.23: S 4195942931:4195942935(4) win 4096
foo.example.com.45820 > 192.168.216.5.23: S 4195944723:4195944727(4) win 4096
foo.example.com.52526 > 172.16.68.5.23: S 357331986:357331990(4) win 4096
foo.example.com.45820 > 192.168.183.5.23: S 4196001810:4196001814(4) win 4096
foo.example.com.52526 > 172.16.248.5.23: S 357312531:357312535(4) win 4096
foo.example.com.45820 > 192.168.205.5.23: S 4196007442:4196007446(4) win 4096
foo.example.com.52526 > 172.16.250.5.23: S 357313043:357313047(4) win 4096
foo.example.com.52526 > 172.16.198.5.23: S 357365266:357365270(4) win 4096
foo.example.com.52526 > 172.16.161.5.23: S 357355794:357355798(4) win 4096
```

Static source port & syn segments with data

Suspicious tcpdump Output?

CEIC 2011

```
10:08:23.472378 state.example.net.1739 > watson.22: S 72549644:72549644(0) win
   8192 (DF)
10:08:25.009256 state.example.net.1739 > watson.22: S 72549644:72549644(0) win
   8192 (DF)
10:08:26.504518 state.example.net.1739 > watson.22: S 72549644:72549644(0) win
   8192 (DF)
10:08:28.006168 state.example.net.1739 > watson.22: S 72549644:72549644(0) win
   8192 (DF)

17:14:18.726864 foo.example.net.62555 > watson.80: S 20583734:20583734(0) win
   8192 <mss 1380> (DF)
17:14:21.781140 foo.example.net.62555 > watson.80: S 20583734:20583734(0) win
   8192 <mss 1380> (DF)
17:14:27.776662 foo.example.net.62555 > watson.80: S 20583734:20583734(0) win
   8192 <mss 1380> (DF)
17:14:39.775929 foo.example.net.62555 > watson.80: S 20583734:20583734(0) win
   8192 <mss 1380> (DF)
```

TCP connection retries (Windows vs. Linux)

Suspicious tcpdump Output?

CEIC 2011

```
16:03:40.763603 foo.example.com.39344 > watson.80: S
   523285584:523285584(0) win 8760 (DF)
16:03:41.919170 foo.example.com.39345 > watson.80: S
   523517577:523517577(0) win 8760 (DF)
16:03:53.348706 foo.example.com.39378 > watson.80: S
   528418601:528418601(0) win 8760 (DF)
16:03:53.491895 foo.example.com.39379 > watson.80: S
   528509044:528509044(0) win 8760 (DF)
```

Multiple connection requests

Suspicious tcpdump Output?

CEIC 2011

```
[root@altamont gck]# tcpdump 'icmp'
12:03:36.016502 doggie.example.edu > 192.0.2.7: icmp: echo reply (DF)
12:03:46.016502 doggie.example.edu > 192.0.2.7: icmp: echo reply (DF)
12:04:37.016502 doggie.example.edu > 192.0.2.7: icmp: echo reply (DF)
12:04:47.006502 doggie.example.edu > 192.0.2.7: icmp: echo reply (DF)
12:05:38.016502 doggie.example.edu > 192.0.2.7: icmp: echo reply (DF)
12:05:48.016502 doggie.example.edu > 192.0.2.7: icmp: echo reply (DF)
12:06:39.006502 doggie.example.edu > 192.0.2.7: icmp: echo reply (DF)
12:06:49.006502 doggie.example.edu > 192.0.2.7: icmp: echo reply (DF)
12:07:40.006502 doggie.example.edu > 192.0.2.7: icmp: echo reply (DF)
12:07:50.006502 doggie.example.edu > 192.0.2.7: icmp: echo reply (DF)
12:08:41.006502 doggie.example.edu > 192.0.2.7: icmp: echo reply (DF)
12:08:51.006502 doggie.example.edu > 192.0.2.7: icmp: echo reply (DF)
12:09:42.016502 doggie.example.edu > 192.0.2.7: icmp: echo reply (DF)
12:09:52.016502 doggie.example.edu > 192.0.2.7: icmp: echo reply (DF)
12:10:43.016502 doggie.example.edu > 192.0.2.7: icmp: echo reply (DF)
12:10:53.016502 doggie.example.edu > 192.0.2.7: icmp: echo reply (DF)
12:11:44.016502 doggie.example.edu > 192.0.2.7: icmp: echo reply (DF)
```

Echo Replies without Echo Requests

Suspicious tcpdump Output?

CEIC 2011

```
[root@altamont gck]# tcpdump 'icmp' -x
12:27:09.016502 doggie.example.edu > 192.0.2.7: icmp: echo reply (DF)
                                4500 0414 0000 4000 4001 cdf9 c670 431e
                                cc59 9307 0000 9ca3 1a0a 0000 0000 0000
                                0000 0000 0000 0000 0000 0000 0000 0000
                                736b 696c 6c7a 0000 0000 0000 0000 0000
                                0000 0000 0000 0000 0000 0000 0000 0000
                                0000
12:28:00.016502 doggie.example.edu > 192.0.2.7: icmp: echo reply (DF)
                                4500 0414 0000 4000 4001 cdf9 c670 431e
                                cc59 9307 0000 9ca3 1a0a 0000 0000 0000
                                0000 0000 0000 0000 0000 0000 0000 0000
                                736b 696c 6c7a 0000 0000 0000 0000 0000
                                0000 0000 0000 0000 0000 0000 0000 0000
                                0000
12:28:10.016502 doggie.example.edu > 192.0.2.7: icmp: echo reply (DF)
                                4500 0414 0000 4000 4001 cdf9 c670 431e
                                cc59 9307 0000 9ca3 1a0a 0000 0000 0000
                                0000 0000 0000 0000 0000 0000 0000 0000
                                736b 696c 6c7a 0000 0000 0000 0000 0000
                                0000 0000 0000 0000 0000 0000 0000 0000
                                0000
```

0x73-6b-69-6c-6c-7a = "skillz" (Stacheldraght)

Suspicious tcpdump Output?

CEIC 2011

```
foo.example.com > watson: icmp: echo request (frag 56980:1480@0+)
foo.example.com > watson: (frag 56980:1480@1480+)
foo.example.com > watson: (frag 56980:1480@2960+)
foo.example.com > watson: (frag 56980:1480@4440+)
foo.example.com > watson: (frag 56980:1480@5920+)
                        :
                        :
foo.example.com > watson: (frag 56980:1480@59200+)
foo.example.com > watson: (frag 56980:1480@60680+)
foo.example.com > watson: (frag 56980:1480@62160+)
foo.example.com > watson: (frag 56980:1480@63640+)
foo.example.com > watson: (frag 56980:1480@65120)
```

Ping of Death

Suspicious tcpdump Output?

CEIC 2011

```
foo.example.com.137 > watson.137: udp 28 (frag 242:36@0+)
foo.example.com > watson: (frag 242:4@24)
```

Teardrop attack (fragment overlap)

Suspicious tcpdump Output?

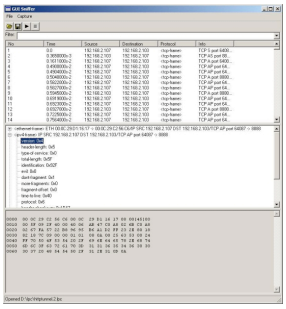
CEIC 2011

```
granite:~ $ ssh root:secret@doggie.example.edu
[root@networking gck]# tcpdump
09:31:15.747498 doggie.example.edu.ssh > granite.sover.net.841: P 691789802:691789850(48) ack 3378360866 win 8576 <nop,nop,timestamp
35003172 37891099> (DF) [tos 0x10]
09:31:15.747646 doggie.example.edu.ssh > granite.sover.net.841: P 48:208(160) ack 1 win 8576 <nop,nop,timestamp 35003172 37891099> (DF)
[tos 0x10]
09:31:15.968039 granite.sover.net.841 > doggie.example.edu.ssh: . 1:1(0) ack 208 win 8760 <nop,nop,timestamp 37891100 35003172> (DF) [tos
0x10]
09:31:15.968103 doggie.example.edu.ssh > granite.sover.net.841: P 208:1360(1152) ack 1 win 8576 <nop,nop,timestamp 35003194 37891100>
(DF) [tos 0x10]
09:31:15.968610 doggie.example.edu.ssh > granite.sover.net.841: P 1360:1712(352) ack 1 win 8576 <nop,nop,timestamp 35003194 37891100>
(DF) [tos 0x10]
09:31:15.968904 doggie.example.edu.ssh > granite.sover.net.841: P 1712:1920(208) ack 1 win 8576 <nop,nop,timestamp 35003194 37891100>
(DF) [tos 0x10]
09:31:16.029210 granite.sover.net.841 > doggie.example.edu.ssh: . 1:1(0) ack 208 win 8760 <nop,nop,timestamp 37891100 35003172> (DF) [tos
0x10]
09:31:16.029303 doggie.example.edu.ssh > granite.sover.net.841: P 1920:2128(208) ack 1 win 8576 <nop,nop,timestamp 35003200 37891100>
(DF) [tos 0x10]
09:31:16.029224 granite.sover.net.841 > doggie.example.edu.ssh: . 1:1(0) ack 208 win 8760 <nop,nop,timestamp 37891100 35003172> (DF) [tos
0x10]
09:31:16.029247 granite.sover.net.841 > doggie.example.edu.ssh: . 1:1(0) ack 1920 win 7048 <nop,nop,timestamp 37891100 35003194> (DF)
[tos 0x10]
09:31:16.029979 doggie.example.edu.ssh > granite.sover.net.841: P 2128:2800(672) ack 1 win 8576 <nop,nop,timestamp 35003200 37891100>
(DF) [tos 0x10]
09:31:16.030289 doggie.example.edu.ssh > granite.sover.net.841: P 2800:3008(208) ack 1 win 8576 <nop,nop,timestamp 35003200 37891100>
(DF) [tos 0x10]
09:31:16.067973 granite.sover.net.841 > doggie.example.edu.ssh: . 1:1(0) ack 2128 win 8760 <nop,nop,timestamp 37891100 35003200> (DF)
[tos 0x10]
09:31:16.068046 doggie.example.edu.ssh > granite.sover.net.841: P 3008:3216(208) ack 1 win 8576 <nop,nop,timestamp 35003204 37891100>
(DF) [tos 0x10]
09:31:16.068168 granite.sover.net.841 > doggie.example.edu.ssh: . 1:1(0) ack 3008 win 7880 <nop,nop,timestamp 37891100 35003200> (DF)
[tos 0x10]
09:31:16.068651 doggie.example.edu.ssh > granite.sover.net.841: P 3216:3728(512) ack 1 win 8576 <nop,nop,timestamp 35003204 37891100>
(DF) [tos 0x10]
09:31:16.068951 doggie.example.edu.ssh > granite.sover.net.841: P 3728:3936(208) ack 1 win 8576 <nop,nop,timestamp 35003204 37891100>
(DF) [tos 0x10]
09:31:16.069236 doggie.example.edu.ssh > granite.sover.net.841: P 3936:4144(208) ack 1 win 8576 <nop,nop,timestamp 35003204 37891100>
(DF) [tos 0x10]
```

GCK's self-DoS

CEIC 2011

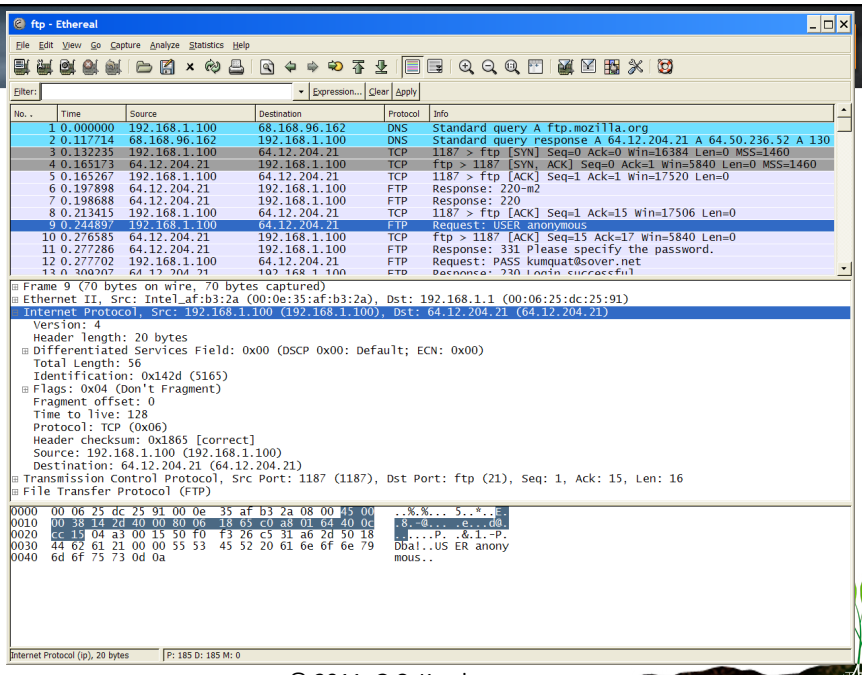
GUI Packet Sniffers

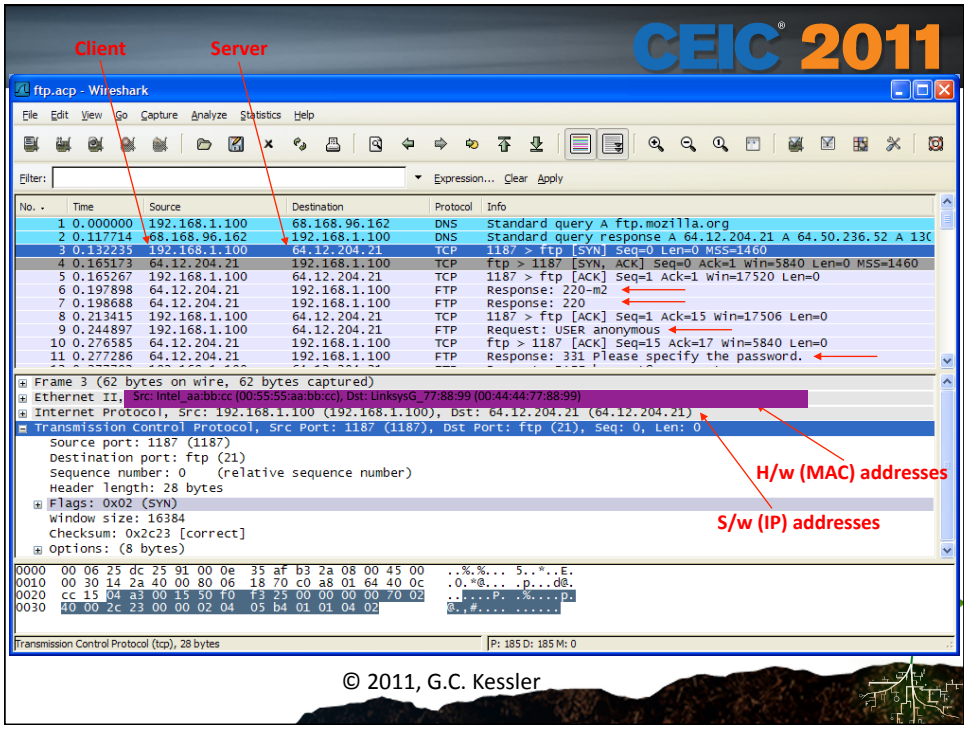


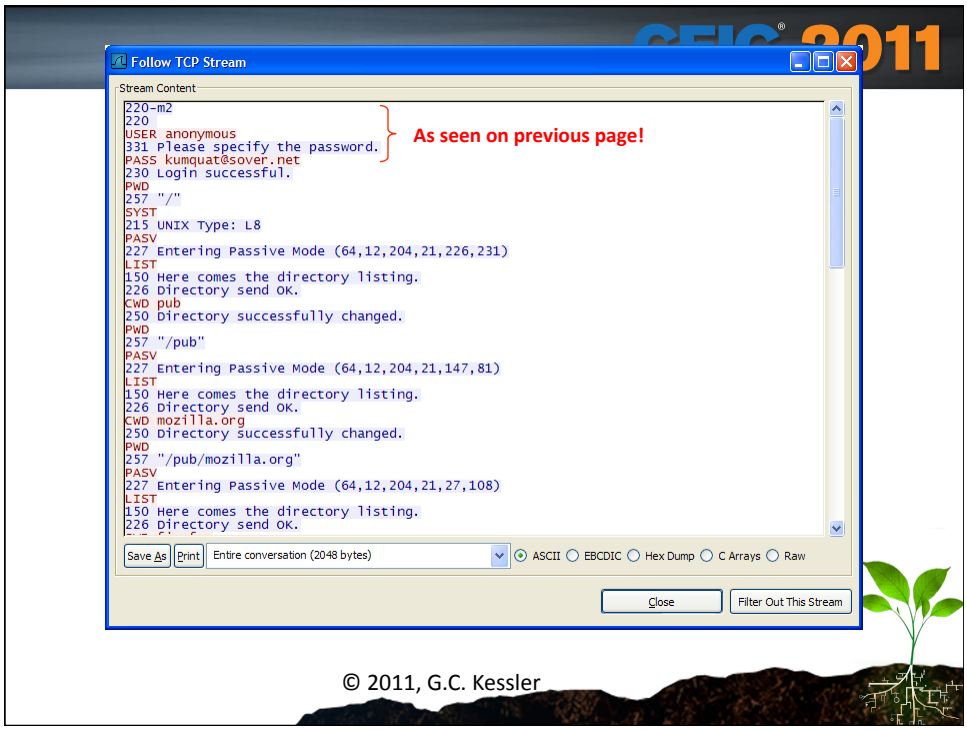
WireShark

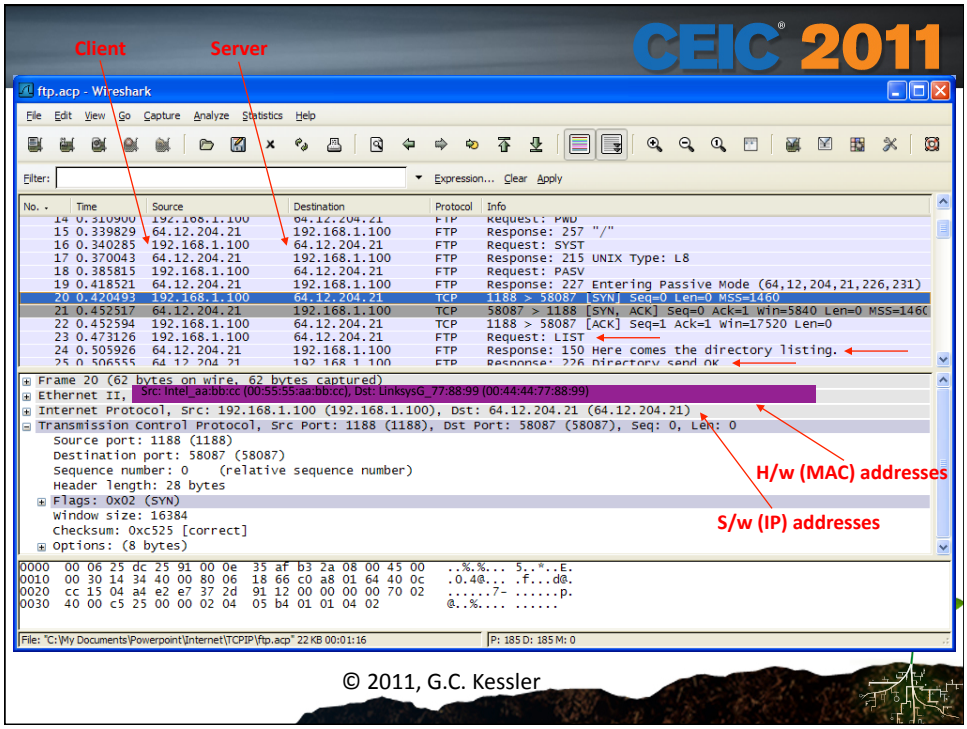
CEIC 2011

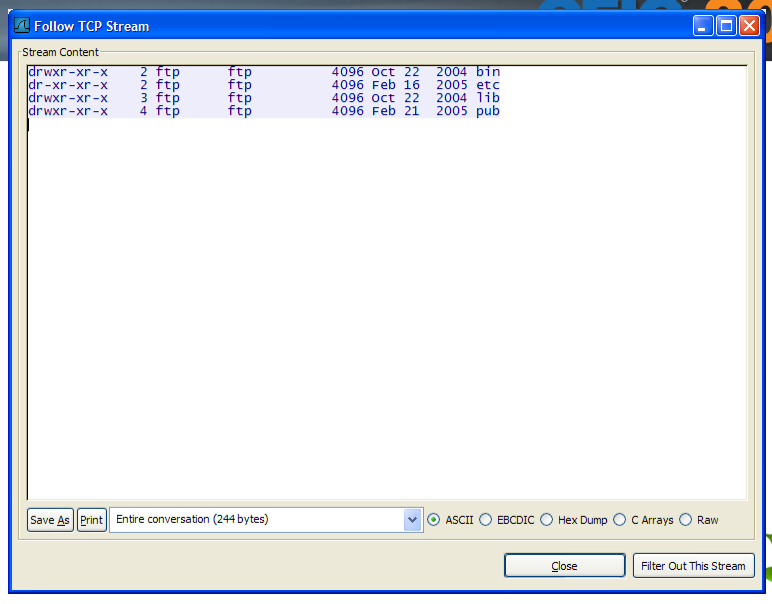
- GUI packet sniffer for both Linux and Windows *(formerly Ethereal)*
  - Handles TCP/IP, NetBIOS, and *many* more protocol
  - Interprets displayed packets
  - Features include ability to filter capture and/or display, and packet stream reassembly
  - *http://www.wireshark.org/*
- Most commonly used open source packet sniffer



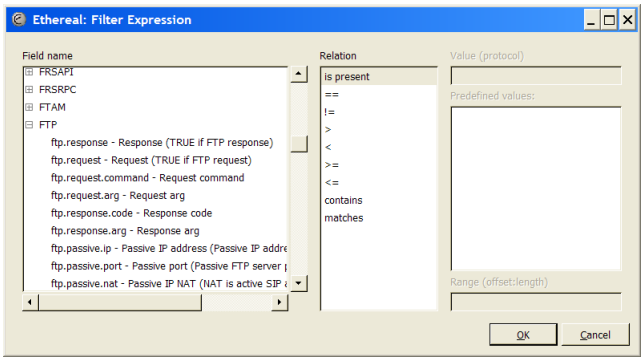


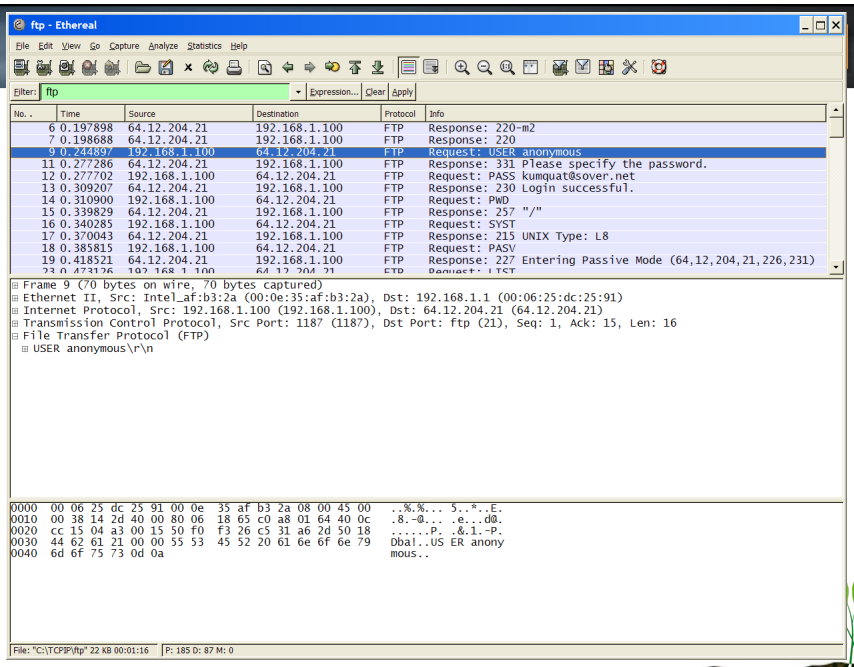






Filtering the Display

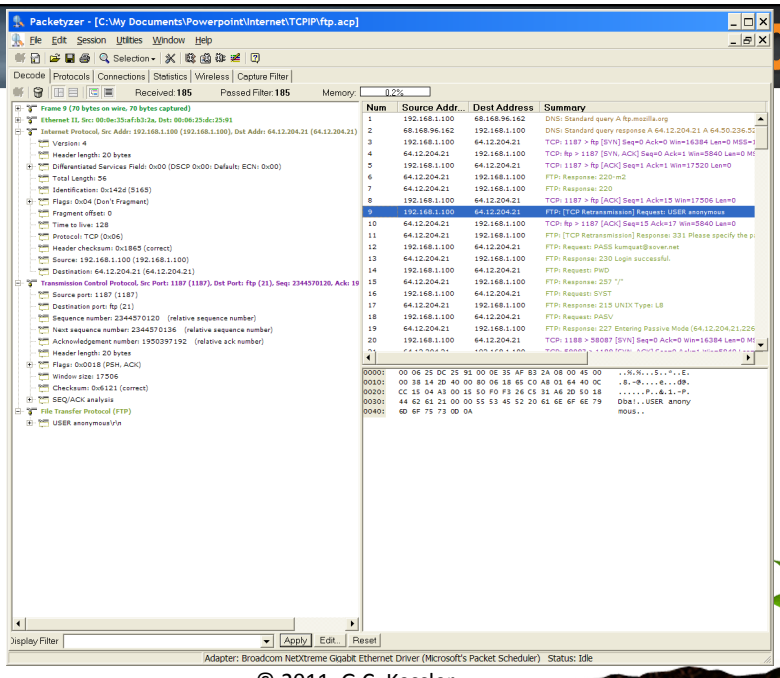


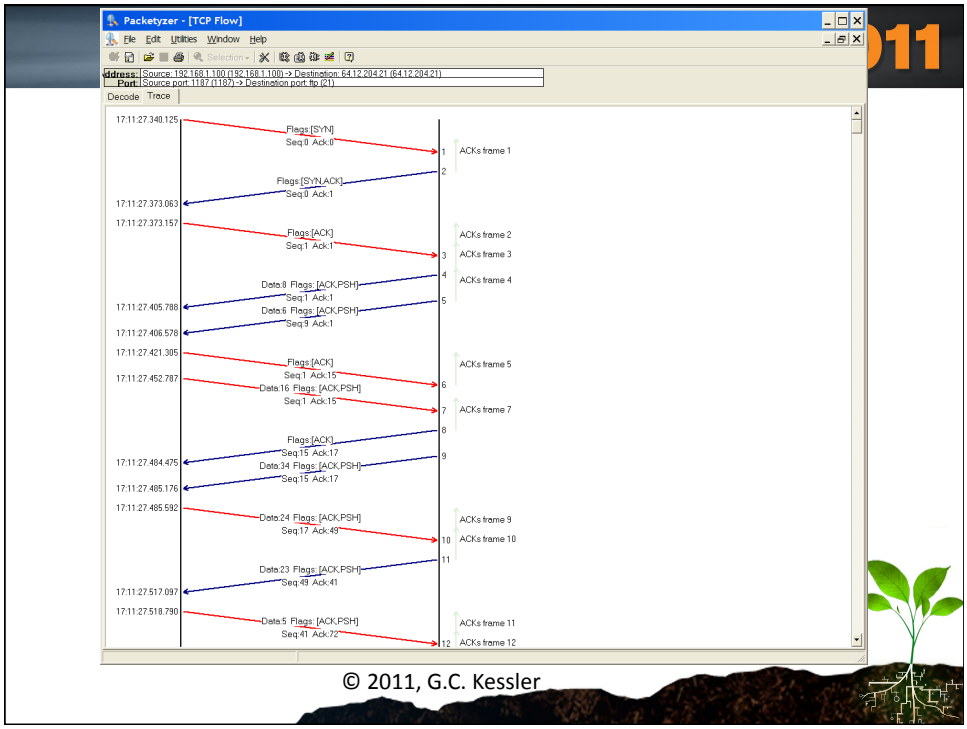


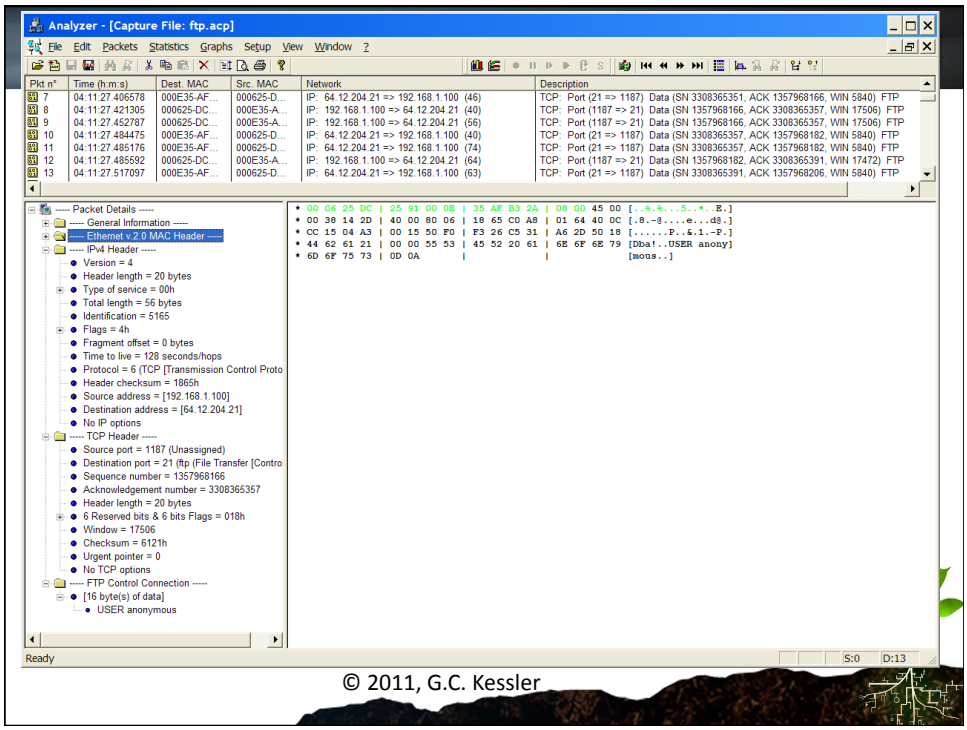
Other GUI Packet Sniffers

CEIC 2011

- Packetyzer
  - Free packet analysis tool
  - *http://www.paglo.com/opensource/packetyzer*
- Analyzer
  - Open source protocol analysis tool
  - *http://analyzer.polito.it/*
- Iris
  - Commercial, very powerful sniffer
  - *http://www.eeye.com/html/products/iris/*



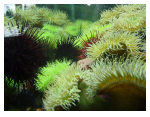




Sample Capture Files

CEIC 2011

- FTP
- Secure FTP
- HTTP
- HTTP Over SSL
- SMTP
- POP v3
- Port scan
- Telnet
- SSH
- Ping (Linux)
- Ping (Windows)
- Traceroute
- Tracert
- Phishing



http://www.garykessler.net/download/tcpip/capture_files.zip

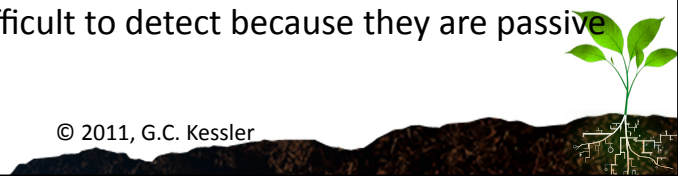
CEIC 2011

Summary, Additional Information, and Appendices

Conclusions

CEIC 2011

- Use of packet sniffers and ability to perform protocol analysis requires understanding of networking and the underlying protocols
- Essential skill for infosec incident responders, network administrators, and network forensic analysts
- Many open source tools available
  - May also be used by hackers on your network and are very difficult to detect because they are passive



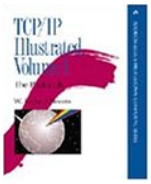
Summary

CEIC 2011

- The TCP/IP Protocol Suite
- Encapsulation and Interpretation
- The Role of Packet Sniffers
  - tcpdump/WinDump
  - Ethereal/WireShark
  - ngrep

For More Information...

CEIC 2011



- *Guide to TCP/IP*, Chappell & Tittel
- *TCP/IP Illustrated, Vol. 1*, Stevens
- "An Overview of TCP/IP Protocols and the Internet"
  - *http://www.garykessler.net/library/tcpip.html*
- Web 2.0 for packets
  - *http://www.pcapr.net/*

Downloads

CEIC 2011

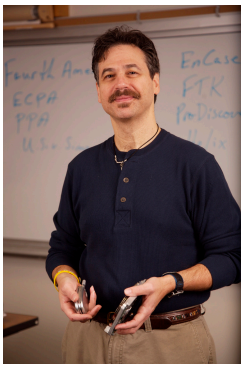
- Protocol analysis software
  - Analyzer *http://analyzer.polito.it/*
  - WireShark *http://www.wireshark.org/*
  - ngrep *http://ngrep.sourceforge.net/download.html*
  - Packetyzer *http://sourceforge.net/projects/packetyzer/*
  - WinDump *http://www.winpcap.org/windump/*
  - WinPcap*http://www.winpcap.org/*
- *http://www.garykessler.net/download/tcpip/*filename
  - Sample capture files *capture_files.zip*
  - TCP/IP pocket guide *tcpip_prg.pdf*

Speaker Contact Information

Gary C. Kessler, Ph.D., CCE, CISSP
GARY KESSLER ASSOCIATES
2 Southwind Drive
Burlington, VT 05401

mobile: +1 802-238-8913
e-mail: gck@garykessler.net
gkessler@bpdvt.org
gkessle1@norwich.edu
Skype: *gary.c.kessler*

http://www.garykessler.net
http://www.vtinternetcrimes.org
http://infoassurance.norwich.edu



Acronyms & Abbreviations

ACK	Acknowledgement flag (TCP)
ANSI	American National Standards Institute
ARP	Address Resolution Protocol (IETF)
ASCII	American Standard Code for Information Interchange (ANSI)
ATM	Asynchronous Transfer Mode
B	Byte (an *octet*, 8 bits)
BGP	Border Gateway Protocol (IETF)
CATV	Community access (cable) television
CLI	Command line utility
CSLIP	Compressed SLIP (IETF)
DF	Don't Fragment flag (IP)
DHCP	Dynamic Host Configuration Protocol (IETF)
DIG	Domain Internet Groper
DNS	Domain Name System (IETF)
DWDM	Dense Wavelength Division Multiplexing
FDDI	Fiber Distributed Data Interface (ANSI)
FIN	Finish flag (TCP)
FTP	File Transfer Protocol (IETF)
GRE	Generic Routing Encapsulation (IETF)
GUI	Graphical user interface
HDLC	High-level Data Link Control (OSI)
HTML	Hypertext Markup Language (IETF)
HTTP	Hypertext Transfer Protocol (IETF)
https	HTTP over SSL
ICMP	Internet Control Message Protocol (IETF)
IEEE	Institute of Electrical and Electronic Engineers
IETF	Internet Engineering Task Force
IHL	Internet Header Length field (IP)
IMAP	Internet Message Access Protocol (IETF)
IP	Internet Protocol (IETF)
IPsec	Secure IP (IETF)
ISDN	Integrated Services Digital Network
ISN	Initial sequence number (TCP)
ITU-T	International Telecommunication Union – Telecommunication Standardization Sector
LAN	Local area network
MAC	Media Access Control (IEEE)
MF	More Fragments flag (IP)
MS	Microsoft
MSS	Maximum segment size (TCP)
NAT	Network Address Translation (IETF)
NetBIOS	Network Basic Input/Output System (MS)
NIC	Network interface card
NNTP	Network News Transfer Protocol (IETF)
NTP	Network Time Protocol (IETF)
OSPF	Open Shortest Path First (IETF)
OUI	Organizationally Unique Identifier (IEEE)
POP3	Post Office Protocol version 3 (IETF)
PPP	Point-to-Point Protocol (IETF)

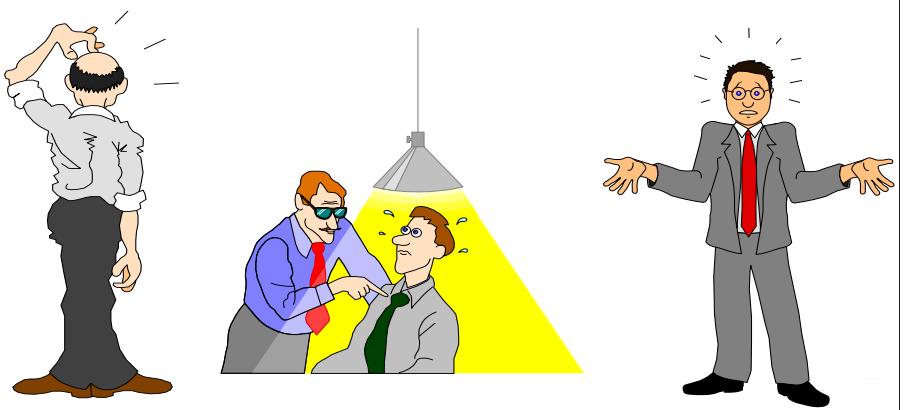
Acronyms & Abbreviations (con't.)

CEIC 2011

PSH	Push flag (TCP)
RADIUS	Remote Authentication Dial In User Service
RFC	Request for Comments (IETF)
RIP	Routing Information Protocol (IETF)
RST	Reset flag (TCP)
SACK	Selective Acknowledgement (TCP)
SDH	Synchronous Digital Hierarchy (ITU-T)
SLIP	Serial Line Interface Protocol (IETF)
SMDS	Switched Multimegabit Data Service
SMTP	Simple Message Transfer Protocol (IETF)
SNAP	Subnetwork Access Protocol (IEEE)
SNMP	Simple Network Management Protocol (IETF)
SONET	Synchronous Optical Network (ANSI)
SSH	Secure Shell
SSL	Secure Sockets Layer (Netscape)
STD	Standard series of RFCs (IETF)
SYN	Synchronization flag (TCP)
TACACS+	Terminal Access Controller Access Control System plus
TCP	Transmission Control Protocol (IETF)
TFTP	Trivial FTP (IETF)
TLS	Transaction Layer Security (IETF)
TTL	Time-to-Live field (IP)
UDP	User Datagram Protocol (IETF)
URG	Urgent flag (TCP)
xDSL	Digital Subscriber Line technology family

Questions? Comments? Queries?

CEIC 2011



CEIC 2011

APPENDIX 1

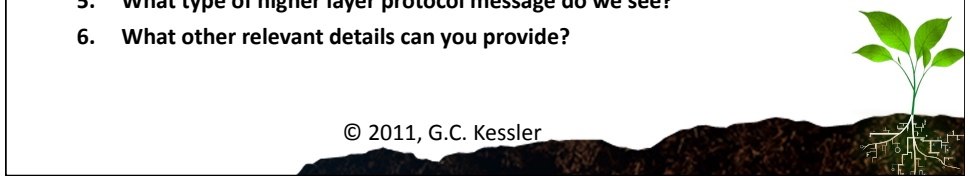
Packet Interpretation Exercises

Exercise 1

CEIC 2011

```
4500 0054 b02f 0000 4001 122e 91f0 8a05
1830 6532 0800 836a cf1b 0000 ecec c339
0150 0900 0809 0a0b 0c0d 0e0f 1011 1213
1415 1617 1819 1a1b 1c1d 1e1f 2021 2223
2425 2627 2829 2a2b 2c2d 2e2f 3031 3233
```

1. **What is the length of the IP header?**
2. **What are the source and destination IP addresses?**
3. **What is the total length of the IP datagram?**
4. **What is the higher layer protocol?**
5. **What type of higher layer protocol message do we see?**
6. **What other relevant details can you provide?**



Exercise 1 Answers

CEIC 2011

```
4500 0054 b02f 0000 4001 122e 91f0 8a05
1830 6532 0800 836a cf1b 0000 ecec c339
0150 0900 0809 0a0b 0c0d 0e0f 1011 1213
1415 1617 1819 1a1b 1c1d 1e1f 2021 2223
2425 2627 2829 2a2b 2c2d 2e2f 3031 3233
```

1. What is the length of the IP header? 20 bytes
2. What are the source and destination IP addresses? 145.240.138.5 and 24.48.101.50
3. What is the higher layer protocol? ICMP
4. What type of higher layer protocol message do we see? Echo
5. What is the total length of the IP datagram? 84 bytes

Exercise 2

CEIC 2011

```
4500 0030 df3c 4000 8006 633f ccf0 8f65
ccf0 8f05 0b64 0015 48f3 05b1 0000 0000
7002 2000 50b6 0000 0204 05b4 0101 0402
```

1. What is the length of the IP header?
2. What are the source and destination IP addresses?
3. What is the total length of the IP datagram?
4. What is the higher layer protocol?
5. What type of higher layer protocol message do we see?
6. What other relevant details can you provide?

Exercise 2 Answers

CEIC 2011

```
4500 0030 df3c 4000 8006 633f ccf0 8f65
ccf0 8f05 0b64 0015 48f3 05b1 0000 0000
7002 2000 50b6 0000 0204 05b4 0101 0402
```

1. What is the length of the IP header? 20 bytes
2. What are the source and destination IP addresses? 204.240.143.101 and 204.240.143.5
3. What is the higher layer protocol? TCP
4. What type of higher layer protocol message do we see? FTP (21/tcp)
5. What is the total length of the IP datagram? 48 bytes
6. If this packet is really torn apart, we might guess that this is the beginning of a 3-way handshake because the ACK field = *0* and there appears to be no data; this is confirmed when we see that the Flags field has the SYN bit on. The long header length (0x7) also tells us that there are options attached.

CEIC 2011

APPENDIX 2

tcpdump/Windump Filters

tcpdump Filters

CEIC 2011

- Allows you to obtain only items of interest from tcpdump output
- Can apply filter to any field in the IP datagram
- Uses standard Berkeley Packet Filter (BPF) format
  - Available on almost all packet sniffers

tcpdump Filter Formats

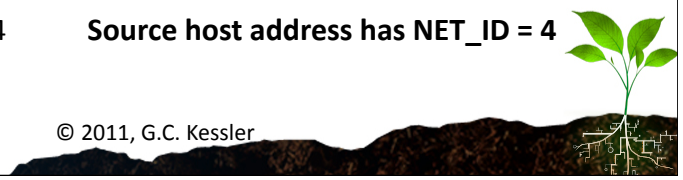
CEIC 2011

- Two different formats for a tcpdump filter
  - *protocol* [*offset*:*length*] *relation value*

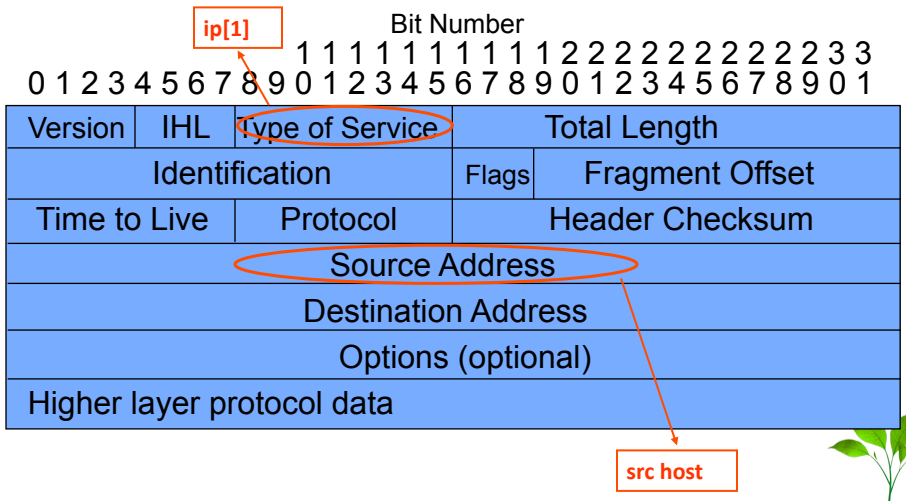
ip[9] = 1	**Protocol = ICMP**
tcp[2:2] < 20	**Destination port <20**
udp[4:2] != 0	**UDP length ≠ 0**
icmp[0] = 8	**ICMP Echo message**

  - *variable value*

port 23	**Source or destination port = 23**
dst host 1.2.3.4	**Destination host = 1.2.3.4**
src net 4	**Source host address has NET_ID = 4**



Location Specification



tcpdump Filter Macros

host	select record if either the source or destination host matches this IP address
net	select record if either the source or destination subnet matches
port	select record if either the source or destination port matches
src host	select record if the source host address matches
dst host	select record if the destination host address matches
src net	select record if the source subnet matches
dst net	select record if the destination subnet matches
src port	select record if the source port matches
dst port	select record if the destination port matches
icmp	select record if the IP Protocol field (ip[9]) has a value of 1
tcp	select record if the IP Protocol field has a value of 6
udp	select record if the IP Protocol field has a value of 17 (0x11)

Filters and Bit Masking

CEIC 2011

- Smallest tcpdump filter unit is a byte
- Reference bits within a byte by masking bits
- Examples:
  - IP version number
  - IP Header Length
  - TCP flags

IP Header Length

CEIC 2011

The IP Header Length is in the four low-order bits of the first byte of an IP packet. ANDing the byte with 00001111 yields the IHL value and masks out the rest of the byte.

```
ip[0]:    0 1 0 0 0 1 0 1    0x45
mask:     0 0 0 0 1 1 1 1    0x0f
          ---------------    ----
AND       0 0 0 0 0 1 0 1    0x05
```

The partial tcpdump filter expression: `ip[0] & 0x0f`

E.g., keep packets only if the IHL is greater than 20B:

```
tcpdump 'ip[0] & 0x0f > 5'
```

IP Version

CEIC 2011

The IP Version is in the four high-order bits of the first byte of an IP packet. ANDing the byte with 11110000 yields the version and masks out the rest of the byte.

```
ip[0]:    0 1 0 0 0 1 0 1    0x45
mask:     1 1 1 1 0 0 0 0    0xf0
          ---------------    ----
AND       0 1 0 0 0 0 0 0    0x40
```

The partial tcpdump filter expression: `ip[0] & 0xf0`

E.g., keep packets only if the version number is 6:

```
tcpdump 'ip[0] & 0xf0 = 0x60'
```

TCP Flag Bits

CEIC 2011

- Located in the 14th byte of the TCP header
  - aka `tcp[13]`
- Tells much about the state of a given TCP segment
- Commonly examined in filters

Reserved	URG	ACK	PSH	RST	SYN	FIN

Masking the TCP Flag Bits

CEIC 2011

	2^3	2^2	2^1	2^0	2^3	2^2	2^1	2^0	
	Reserved		URG	ACK	PSH	RST	SYN	FIN	
SYN-bit mask	0	0	0	0	0	0	1	0	0x02
ACK-bit mask	0	0	0	1	0	0	0	0	0x10
Low-order byte mask	0	0	0	0	1	1	1	1	0x0f

To get all packets with the SYN-bit set:

```
tcpdump 'tcp[13] & 0x02 = 2'
```

To get all packets with the ACK-bit set:

```
windump "tcp[13] & 0x10 = 16"
```

To get all packets with any of the low-order flags set:

```
tcpdump 'tcp[13] & 0x0f != 0'
```


Checking For Multiple Bits Set

CEIC 2011

- Check for either the SYN bit, FIN bit, or both are set:
  - `tcp[13] & 0x03 != 0`
- Check for only both being set
  - `tcp[13] & 0x03 = 3`

Advanced Filters

CEIC 2011

- More complex filters can be created by combining individual filters with Boolean `and`, `or`, and `not` operators

```
(tcp and (tcp[13] & 0x0f != 0) and not
  port 25 and not port 20) or
(ip and not (tcp or igrp or dst port
  520))
```


CEIC 2011

APPENDIX 3

ngrep

ngrep

CEIC 2011

- Can use expressions to control display and analysis of tcpdump/WinDump traffic
  - ngrep = *network get regular expression*
- Reads and displays packets in tcpdump format

ngrep

CEIC 2011

```
C:\windump> ngrep -I miscellaneous.acp "GET" "port 80"
input: miscellaneous.acp
filter: (ip or ip6) and ( port 80 )
match: GET
####
T 192.168.1.102:1381 -> 216.93.159.180:80 [AP]
  GET /syllabus/index.html HTTP/1.1..Host: digitalforensics.champlain.edu..Us
  er-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.9) Gecko/2
  0050711 Firefox/1.0.5..Accept: text/xml,application/xml,application/xhtml+x
  ml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5..Accept-Language: e
  n-us,en;q=0.5..Accept-Encoding: gzip,deflate..Accept-Charset: ISO-8859-1,ut
  f-8;q=0.7,*;q=0.7..Keep-Alive: 300..Connection: keep-alive..Referer: http:/
  /digitalforensics.champlain.edu/..Cookie: __utma=267634908.433925735.111841
  3029.1124165129.1125604057.4; __utmz=267634908.1124161444.2.2.utmccn=(organ
  ic)|utmcsr=google|utmctr=bookstore|utmcmd=organic; WebCTTicket=username%3Dk
  esslerg%26password%3DL0GhXVSZfErE2%26expiry%3D1126116666%26hash%3D5526cd346
  a6ff221c7dbcf7f43446bb7..If-Modified-Since: Wed, 07 Sep 2005 15:40:56 GMT..
  If-None-Match: "103276-24f3-431f0a08"....
###########
T 192.168.1.102:1382 -> 128.9.160.27:80 [AP]
  GET / HTTP/1.1..Host: www.rfc-editor.org..User-Agent: Mozilla/5.0 (Windows;
   U; Windows NT 5.1; en-US; rv:1.7.9) Gecko/20050711 Firefox/1.0.5..Accept:
  text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q
  =0.8,image/png,*/*;q=0.5..Accept-Language: en-us,en;q=0.5..Accept-Encoding:
   gzip,deflate..Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7..Keep-Alive:
  300..Connection: keep-alive....
##############
```

Some ngrep Switches

CEIC 2011

`-i`	Ignore case in filter/regex
`-d`	Read from specified WinPcap interface
`-F`	Read filter from specified file
`-h`	Help
`-I`	Obtain input from file
`-L`	List WinPcap interfaces
`-n`	Look at only specified number of packets
`-O`	Send output to file
`-x`	Display in hexadecimal and ASCII
`-X`	Interpret match expression as a hexadecimal string

Finding Strings With ngrep

CEIC 2011

- ngrep has a variety of options with which to find strings within the packet stream
  - Fixed ASCII string search

    ```
    ngrep "gck"
    ```
  - Regular expression search (see *http://www.regular-expressions.info/*)

    ```
    ngrep -i "GET.*\.htm" "port 80"
    ```
  - Hex string search

    ```
    ngrep -X "0x90909090"
    ```

CEIC 2011

APPENDIX 4

Base Conversion Chart

CEIC 2011

Decimal	Hexadecimal	Binary	Decimal	Hexadecimal	Binary
0	0x00	00 0000	16	0x10	01 0000
1	0x01	00 0001	17	0x11	01 0001
2	0x02	00 0010	18	0x12	01 0010
3	0x03	00 0011	19	0x13	01 0011
4	0x04	00 0100	20	0x14	01 0100
5	0x05	00 0101	21	0x15	01 0101
6	0x06	00 0110	22	0x16	01 0110
7	0x07	00 0111	23	0x17	01 0111
8	0x08	00 1000	24	0x18	01 1000
9	0x09	00 1001	25	0x19	01 1001
10	0x0a	00 1010	26	0x1a	01 1010
11	0x0b	00 1011	27	0x1b	01 1011
12	0x0c	00 1100	28	0x1c	01 1100
13	0x0d	00 1101	29	0x1d	01 1101
14	0x0e	00 1110	30	0x1e	01 1110
15	0x0f	00 1111	31	0x1f	01 1111
			32	0x20	10 0000