COMPUTER NETWORKING

Introduction

Part 2
OUTLINE

A. The Internet, TCP/IP, and LANs
B. Ping sweeps, port scans, traceroutes, & OS fingerprinting
C. An introduction to security issues
   a. Attack Points
      – Human access
      – Physical access
      – LAN access
      – Wireless access
      – Remote (Internet) access
   b. Scanning your site
   c. The law
The Anatomy of an Attack

**Introductory Labs**

- **Lab #1**: Survey the University of the Pacific
- **Lab #2**: Survey Dave, UOP’s IT security officer
- **Lab #3**: Use Nmap to ping sweep/port scan your own computer (or the lab’s LAN)
- **Lab #4**: Use Nbtenum to capture user data from your own computer (or from the lab’s LAN)
- **Lab #5**: Use SAMInside to capture your own password hashes. Use SamInside and John to crack the passwords
- **Lab #6**: Use Nessus to vulnerability test your own computer (or the lab’s LAN)
- **Lab #7**: Use Ethereal to capture packets on the lab’s LAN
The Internet, TCP/IP, and LANs

The TCP/IP Protocol Stack

Application (IE, etc.)
Transport (TCP, UDP)
Network (IP)
Data Link (Ethernet)

MAC = Media Access Control
MAC address
IP address
Port number
Message
Segment
Packet
Frame
Physical Layer (the cable)
The Internet, TCP/IP, and LANs

The TCP/IP Protocol Stack

Application (IE, etc.)

Transport (TCP,UDP)

Network (IP)

Data Link (Ethernet)

Port number

IP address

MAC address

MAC = Media Access Control

Physical Layer (the cable)

Message

Segment

Packet

Frame

See next slide
The Transport Layer is responsible for program-to-program communication (therefore, it uses port numbers)

In TCP/IP there are two Transport Layer protocols:

- The Transport Control Protocol (TCP) guarantees that information is received as it was sent – it’s connection-oriented
- The User Datagram Protocol (UDP) performs no end-to-end reliability checks – it’s connectionless

Let’s look at the TCP and UDP headers....
The **UDP Header**

- Compare this header with the **TCP header**
- What’s added?
  - What can TCP do that UDP *cannot* do?
    - **Be reliable!**
  - What can UDP do?

<table>
<thead>
<tr>
<th>0</th>
<th>15</th>
<th>16</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>16-bit source port</td>
<td>16-bit destination port</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16-bit UDP length</td>
<td>16-bit UDP checksum</td>
<td></td>
<td></td>
</tr>
<tr>
<td>data (if any)</td>
<td>8 bytes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TCP Header**

- 16-bit origin port number
- 16-bit destination port number
- 32-bit sequence number
- 32-bit acknowledgement number
- 4-bit length (Header)
- 6-bit Reserved
- 16-bit TCP checksum
- 16-bit urgent pointer
- Option fields (if any)
- Data (if any)
The Option fields are used by some packets for metadata.
Since TCP is connection-oriented, it requires a **3-way handshake** to get started (notice the reserved port numbers).
### TCP 3-way Handshake Example

In the image, we see a network packet capture using a packet sniffer tool, showing a 3-way handshake between two devices:

- **66.249.89.99** = Google
- **192.168.0.206** = Me

The packets are as follows:

1. **SYN** from Google (66.249.89.99) to Me (192.168.0.206)
2. **SYN-ACK** from Me back to Google
3. **ACK** from Google, confirming the connection

This is a typical sequence in establishing a TCP connection.
Using Ethereal to watch the 3-hand handshake

- Run Ethereal.
  - Click on the desktop icon or…
  - Click on Start/All Programs/Ethereal/Ethereal.
- Select Capture/Interfaces (see next slide).

You will repeat this in LAB #3
Using Ethereal to watch the 3-hand handshake

- *The names of the network interfaces will vary!*
- Choose the one with the **192.168.y.x** address
  - If in doubt, ask!  \( y = 0 \text{ or } 1 \)
- Click on the **Prepare** for that interface  \( x = 101 \text{ to } 254 \)

**NOTE:** Ethereal works on most wireless LANs now!
This window appears:

- “Capture packets in promiscuous mode” is set by default
- Check these
- Click on Start
The Ethereal Capture window appears

- Open a website, for example:  
  - www.google.com
- Close the website (click the x)
- Then click on Stop
- The Ethereal results window will appear….
  - See next slide…
<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>21.111795</td>
<td>192.168.0.1</td>
<td>192.168.0.206</td>
<td>DNS</td>
<td>Standard query response, No such name</td>
</tr>
<tr>
<td>27</td>
<td>21.226381</td>
<td>192.168.0.206</td>
<td>66.249.89.99</td>
<td>TCP</td>
<td>1220 &gt; http [SYN] Seq=0 Ack=0 win=16384 Len=0 MSS=1</td>
</tr>
<tr>
<td>28</td>
<td>21.406711</td>
<td>66.249.89.99</td>
<td>192.168.0.206</td>
<td>TCP</td>
<td>http &gt; 1220 [SYN, ACK] Seq=0 Ack=1 win=8190 Len=0 M:</td>
</tr>
<tr>
<td>29</td>
<td>21.406749</td>
<td>192.168.0.206</td>
<td>66.249.89.99</td>
<td>TCP</td>
<td>1220 &gt; http [ACK] Seq=1 Ack=1 win=17640 Len=0</td>
</tr>
<tr>
<td>30</td>
<td>21.406889</td>
<td>192.168.0.206</td>
<td>66.249.89.99</td>
<td>HTTP</td>
<td>GET / HTTP/1.1</td>
</tr>
<tr>
<td>31</td>
<td>21.541267</td>
<td>192.168.0.206</td>
<td>192.168.0.1</td>
<td>DNS</td>
<td>Standard query response, No such name</td>
</tr>
<tr>
<td>32</td>
<td>21.591153</td>
<td>66.249.89.99</td>
<td>192.168.0.206</td>
<td>TCP</td>
<td>[TCP segment of a reassembled PDU]</td>
</tr>
<tr>
<td>33</td>
<td>21.592165</td>
<td>66.249.89.99</td>
<td>192.168.0.206</td>
<td>TCP</td>
<td>1220 &gt; http [ACK] Seq=335 Ack=1679 win=17640 Len=0</td>
</tr>
<tr>
<td>34</td>
<td>21.593146</td>
<td>168.95.192.1</td>
<td>192.168.0.206</td>
<td>DNS</td>
<td>Standard query response, No such name</td>
</tr>
<tr>
<td>35</td>
<td>21.712231</td>
<td>66.249.89.99</td>
<td>192.168.0.206</td>
<td>TCP</td>
<td>HTTP/1.1 200 OK (text/html)</td>
</tr>
<tr>
<td>36</td>
<td>21.713043</td>
<td>66.249.89.99</td>
<td>192.168.0.206</td>
<td>HTTP</td>
<td>1220 &gt; http [ACK] Seq=335 Ack=1679 win=17640 Len=0</td>
</tr>
</tbody>
</table>

**3-way handshake!**
Frame Ethernet Internet Protocol TCP

Select a SYN packet

Then right click in the Middle Frame and select “Expand All”

See next slide…
In Ethereal’s Middle Frame, you will see:

- A “Frame” section
- An “Ethernet” section
  - Here, you will find information found in the MAC header
- An “Internet Protocol” section
  - Here, you find information found in the IP header
- A “User Datagram Protocol” (UDP) or “Transmission Control Protocol” (TCP) section
  - Here, you find information found in the UDP or TCP header
<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>113</td>
<td>162.166811</td>
<td>Cisco-LI_79:4:e:95</td>
<td>01:00:5e:7f:ff:fa</td>
<td>LLC</td>
<td>I P, N(R)=87, N(S)=87</td>
</tr>
<tr>
<td>114</td>
<td>162.17032</td>
<td>Cisco-LI_79:4:e:95</td>
<td>01:00:5e:7f:ff:fa</td>
<td>LLC</td>
<td>S, func=RR, N(R)=87</td>
</tr>
<tr>
<td>115</td>
<td>162.174111</td>
<td>Cisco-LI_79:4:e:95</td>
<td>01:00:5e:7f:ff:fa</td>
<td>LLC</td>
<td>U P, func=Unknown</td>
</tr>
<tr>
<td>116</td>
<td>162.177266</td>
<td>Cisco-LI_79:4:e:95</td>
<td>01:00:5e:7f:ff:fa</td>
<td>LLC</td>
<td>I P, N(R)=81, N(S)=81</td>
</tr>
<tr>
<td>117</td>
<td>162.180984</td>
<td>Cisco-LI_79:4:e:95</td>
<td>01:00:5e:7f:ff:fa</td>
<td>LLC</td>
<td>I, N(R)=89, N(S)=89</td>
</tr>
<tr>
<td>118</td>
<td>162.184711</td>
<td>Cisco-LI_79:4:e:95</td>
<td>01:00:5e:7f:ff:fa</td>
<td>LLC</td>
<td>S F, func=SREJ</td>
</tr>
<tr>
<td>119</td>
<td>162.188341</td>
<td>Cisco-LI_79:4:e:95</td>
<td>01:00:5e:7f:ff:fa</td>
<td>LLC</td>
<td>S P, func=SREJ</td>
</tr>
<tr>
<td>120</td>
<td>177.047120</td>
<td>192.168.1.101</td>
<td>64.12.24.200</td>
<td>LLC</td>
<td>AIM Me AIM Messaging, Chat</td>
</tr>
</tbody>
</table>

Frame 120 (250 bytes on wire, 250 bytes captured)
IEEE 802.11

```
0080 71 05 01 00 04 00 01 00 01 00 01 00 06 00 00 00 00 00
0090 00 06 d7 20 64 61 64 20 6e 65 64 64 73 20 74 6f 6f
00a0 20 70 72 61 63 64 61 63 66 69 63 6f 69 63 69 64 65 66
00b0 67 20 61 66 61 69 66 6e 20 73 6f 20 68 65 69 63 69 64
00c0 65 64 73 20 6d 65 20 74 6f 20 73 6f 20 73 6f 20 73 6f
00d0 20 6d 65 73 61 63 64 61 63 66 69 63 6f 69 63 69 64 65
00e0 67 20 61 66 61 69 66 6e 20 73 6f 20 68 65 69 63 69 64
00f0 65 64 73 20 6d 65 20 74 6f 20 73 6f 20 73 6f 20 73 6f
```

IM: q............. e...
my dad needs to
practic e hackin
g again so he ne
eds me t o send a
message ... so w
hat are you up t
o?....... =.

P: 153 D: 153 M: 0
<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>135</td>
<td>206.12/53</td>
<td>192.168.1.1</td>
<td>239.255.255.250</td>
<td>SSDP</td>
<td>NOTIFY * HTTP/1.</td>
</tr>
<tr>
<td>136</td>
<td>207.18685</td>
<td>192.168.1.101</td>
<td>64.12.24.200</td>
<td>AIM</td>
<td>Keep Alive</td>
</tr>
<tr>
<td>137</td>
<td>207.18693</td>
<td>192.168.1.101</td>
<td>216.155.193.151</td>
<td>TCP</td>
<td>[TCP segment of</td>
</tr>
<tr>
<td>138</td>
<td>207.28129</td>
<td>64.12.24.200</td>
<td>192.168.1.101</td>
<td>TCP</td>
<td>5190 &gt; 1065 [ACK</td>
</tr>
<tr>
<td>139</td>
<td>207.37828</td>
<td>216.155.193.151</td>
<td>192.168.1.101</td>
<td>TCP</td>
<td>5050 &gt; 1073 [ACK</td>
</tr>
<tr>
<td>140</td>
<td>207.516751</td>
<td>64.12.24.200</td>
<td>192.168.1.101</td>
<td>AIM</td>
<td>Bi Oncoming Buddy:</td>
</tr>
<tr>
<td>141</td>
<td>207.61757</td>
<td>64.12.24.200</td>
<td>192.168.1.101</td>
<td>TCP</td>
<td>[TCP ACKed Lost</td>
</tr>
<tr>
<td>142</td>
<td>207.62702</td>
<td>64.12.24.200</td>
<td>192.168.1.101</td>
<td>AIM</td>
<td>Lo AIM Location, Us</td>
</tr>
</tbody>
</table>

```
Webpage
```

```
S COLTS- SUPERBOW
L XLI CH AMPs! <b>r>
nt face= "Lucida"
Handwriting" siz=
e=2 colo r=#00800
0 back=# fffffffffff
ome peopl e are s
o special that o
nce they enter y
our life, it bec
omes rich er, ful

```
The TCP/IP Protocol Stack

- Application (IE, etc.)
- Transport (TCP, UDP)
- Network (IP)
- Data Link (Ethernet)

The Internet, TCP/IP, and LANs

Could be a UDP header

MAC = Media Access Control
The Internet and TCP/IP

Deals with fragmentation – not a big deal

<table>
<thead>
<tr>
<th>4-bit version</th>
<th>4-bit header length</th>
<th>8-bit Type-of-Service</th>
<th>16-bit total length</th>
</tr>
</thead>
<tbody>
<tr>
<td>16-bit identifier</td>
<td>3-bit flags</td>
<td>13-bit fragment offset</td>
<td></td>
</tr>
</tbody>
</table>

- **ICMP** = Internet Control Message Protocol
- 01H = ICMP
- 06H = TCP
- 11H = UDP
- Plus others...

**IP Header**

- 20 Bytes
- 8-bit Time-To-Live
- 8-bit Protocol
- 16-bit header checksum
- 32-bit origin IP address
- 32-bit destination IP address
- Option fields (if any)
- Data (if any)
Let’s look at a real Internet packet…
### The Slammer Packet

<table>
<thead>
<tr>
<th>4500</th>
<th>0194</th>
<th>aa13</th>
<th>0000</th>
<th>0111</th>
<th>386d</th>
<th>c0e4</th>
<th>8b2f</th>
</tr>
</thead>
<tbody>
<tr>
<td>e2b9</td>
<td>a70b</td>
<td>0408</td>
<td>059a</td>
<td>0180</td>
<td>5b6a</td>
<td>0401</td>
<td>0101</td>
</tr>
<tr>
<td>0101</td>
<td>0101</td>
<td>0101</td>
<td>0101</td>
<td>0101</td>
<td>0101</td>
<td>0101</td>
<td>0101</td>
</tr>
<tr>
<td>0101</td>
<td>0101</td>
<td>0101</td>
<td>0101</td>
<td>0101</td>
<td>0101</td>
<td>0101</td>
<td>0101</td>
</tr>
<tr>
<td>0101</td>
<td>0101</td>
<td>0101</td>
<td>0101</td>
<td>0101</td>
<td>0101</td>
<td>0101</td>
<td>0101</td>
</tr>
<tr>
<td>0101</td>
<td>0101</td>
<td>0101</td>
<td>0101</td>
<td>0101</td>
<td>0101</td>
<td>0101</td>
<td>0101</td>
</tr>
<tr>
<td>0101</td>
<td>0101</td>
<td>0101</td>
<td>0101</td>
<td>0101</td>
<td>0101</td>
<td>0101</td>
<td>0101</td>
</tr>
</tbody>
</table>

#### 20-byte IP Header

#### 8-byte UDP Header

0101... overflows a buffer

04 = Start of data

It's all really just 1s and 0s!

Note that the packet is terminated with a 00 byte (NULL character)
4500 0194 aa13 0000 0111 386d c0e4 8b2f

e2b9 a70b 0408 059a 0180 5b6a 04

20-byte IP Header

8-byte UDP Header

04 = Start of data

4 = IPv4

5 = 5 x 4 = 20-byte header length

00 = TOS (not used)

0194 = packet length in hex

aa13 = fragment identifier

0000 = fragment flags/offset

01 = TTL

11 = protocol (UDP)

386d = header checksum

c0e4 8b2f = source IP address

e2b9 a70b = destination IP addr
The Internet, TCP/IP, and LANs

**ENCAPSULATION**

Think: matryoshka doll!

- **GOING DOWN**
  - The layer above adds a header and sends the result to the lower layer.

- **GOING UP**
  - The layer below strips off the header and sends the result to the higher layer.
Ping Sweeps
Port Scans
Traceroutes
OS fingerprinting
Why do we need to know about ping sweeps, port scans, traceroutes, and OS fingerprinting?

Because that the Bad Guys can use...

- Ping sweeps to determine what computers are up and running
- Port scans to determine what programs are listening for connections
- Traceroutes to determine what’s between this network and me
- OS fingerprinting to determine what OSs your computer are using
Ping sweeps, port scans, traceroutes, OS fingerprinting

You can see if a single box is up by simply pinging it:

DNS (Domain Name Service) at work!

```
C:\>ping www.google.com

Pinging www.google.akadns.net [64.233.161.99] with 32 bytes of data:

Reply from 64.233.161.99: bytes=32 time=8ms TTL=248
Reply from 64.233.161.99: bytes=32 time=7ms TTL=248
Reply from 64.233.161.99: bytes=32 time=7ms TTL=248
Reply from 64.233.161.99: bytes=32 time=7ms TTL=248

Ping statistics for 64.233.161.99:
   Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
   Approximate round trip times in milli-seconds:
     Minimum = 7ms, Maximum = 8ms, Average = 7ms
```
Ping Sweeps

- Uses ICMP’s ping command to send an “echo request”
- The destination IP address is incremented through a range of addresses
- Protocol = 1 (ICMP)
- Increments destination IP address
- ICMP type here
  8 = echo (request)
  0 = echo reply
  11 = TTL ⇒ 0
  3 = dest. unreach.

ICMP = Internet Control Message Protocol
Port Scans

- Port scanners send a series of packets while incrementing the destination port number (e.g. from 1 to 1023)

- Scans can include both TCP & UDP ports by setting the Protocol byte in the IP header (see previous slide)

**GOAL:** What services are running?
**Nmap**
The most popular tool used to scan a network for active IP address and open ports is **Nmap**. It is available, for free, for DOS & Linux.

1. **Ping sweep**
   > nmap 192.168.0.2-10

2. **Ping sweep + port scan**
   > nmap -p 1-3389 192.168.0.2-10

3. **Ping sweep + port scan + OS fingerprinting**
   > nmap -O -p 1-3389 192.168.0.2-10

1. This **ping sweeps** IP addresses 192.168.0.1 to 192.168.0.10
2. And **port scans** 1 to 3389 (-p 1-3389) at each live address
3. And **fingerprints** the operating systems (-O)
Port Scans Legal, Judge Says

*December 18, 2000, SecurityFocus.com*

A federal court [United States District Court, Southern District of Florida] found that scanning a network doesn't cause damage, or threaten public health and safety.

Judge Thomas Thrash found that the value of time spent investigating a port scan can not be considered damage. "The statute clearly states that the damage must be an impairment to the integrity and availability of the network," wrote the judge, who found that a port scan impaired neither.

"It says you can't create your own damages by investigating something that would not otherwise be a crime," says hacker defense attorney Jennifer Granick. "It's a good decision for computer security researchers."
Traceroutes

- Use the TTL (Time To Live) field in the IP header
- Set TTL to 1, 2, 3… and sends the packet to the destination IP
- When a router decrements the TTL to zero, it returns a “TTL exceeded message to the originator (source IP address) and (usually) identifies itself
- Note: On Windows boxes, it’s called “tracert”

**GOAL: Where’s my target? Where is this box that my personal firewall says is scanning me? What is the address of the target’s edge router?**
Ping sweeps, port scans, traceroutes, OS fingerprinting

First, a DNS request

C:\Documents and Settings\Name>tracert www.google.com

Tracing route to www.l.google.com [64.233.161.99] over a maximum of 30 hops:

1  4 ms  9 ms  5 ms  192.168.1.1
2 34 ms  26 ms 27 ms ip68-227-192-1.dc.dc.cox.net [68.227.192.1]
3 19 ms  12 ms 16 ms 68.100.1.113
4 22 ms  18 ms 22 ms ip68-100-0-65.dc.dc.cox.net [68.100.0.65]
5 32 ms  26 ms 27 ms mrfdds9j02gex070003.rd.dc.cox.net [68.100.0.113]
6 13 ms  19 ms 17 ms mrfdbbrc02-pos0101.rd.dc.cox.net [68.1.1.6]
7 38 ms  42 ms 35 ms asbhbbrj01-pos020100.r2.as.cox.net [68.1.1.1]
8 41 ms  88 ms 16 ms 68.105.30.118
9 34 ms  27 ms 20 ms 216.239.47.153
10 37 ms  29 ms 27 ms 216.239.48.198
11 36 ms  42 ms 40 ms 64.233.161.99

Trace complete.
Visual-Route ($50) presents the information in a more readable format.
OS Fingerprinting

• The process of identifying the target’s OS
  – Passive fingerprinting involves simply examining the target’s packets
    • Different OSs often use different default values in the TCP and IP header fields
  – Active fingerprinting involves measuring the target’s response to unexpected or mal-formed packets
    • Different OSs often respond differently
Why OS Fingerprint?

• Every OS has its own unique vulnerabilities (like locks).
• No point in going after /etc/passwd on a Windows NT box
  – NT/2K/XP/W2003 passwords are stored in a SAM (Security Account Manager) file
• And no point in going after an IIS-specific exploit on a Linux box
  – IIS (Internet Information Service), a Web server, is a Windows-only program
• **ALSO:** We can trace a moving target (when its IP address is changing) by knowing that target’s TCP/IP fingerprint
  – Perhaps its packets are captured at an ISP or at an Internet backbone router
  – A traceroute can help geolocate the target
OS Fingerprinting

- The standards (RFCs) don’t cover every possibility; therefore, the writers of TCP/IP stacks must decide:
  - What the **TCP** window size should be
    - The RFC (Request For Comments) dictates nothing
  - What **TCP** options, if any, should be used
    - The RFC dictates nothing
  - What the **IP** 16-bit identifier value in the IP header should be
    - The RFC dictates only that it uniquely identify a series of fragments; some increment it by 1 with each packet
  - What the **IP** TTL (Time To Live) value must be
    - The RFC dictates only that it be large enough to get to the destination
OS Fingerprinting

TCP Header
- 16-bit origin port number
- 16-bit destination port number
- 32-bit sequence number
- 32-bit acknowledgement number
- 16-bit window size
- 16-bit TCP checksum
- 16-bit urgent pointer
- Option fields (if any)
- Data (if any)

IP Header
- 4-bit version
- 4-bit header length
- 8-bit Type-of-Service
- 16-bit total length
- 16-bit identifier
- 3-bit flags
- 13-bit fragment offset
- 8-bit Protocol
- 16-bit header checksum
- 32-bit origin IP address
- 32-bit destination IP address
- Option fields (if any)
- Data (if any)

See next two slides…
OS Fingerprinting

- Example: 16-bit Window Size
  - Windows 98: 8192
  - Windows 2000: 16384
  - Windows XP: 64240
  - Linux 2.2: 32120
  - Linux 2.4: 5840
OS Fingerprinting

- Example: 8-bit Time-To-Live
  - Windows 9x/NT: 32
  - Windows 2K/XP: 128
  - Digital Unix: 60
  - Linux 2.2.x: 64
  - Solaris 2.x: 255
- Average decrement via Internet is 30, maximum is 60
• Now that we know what…
  • Ping sweeps are
  • Port scans are
  • Traceroutes are
  • OS fingerprinting is
• Let’s move on…
A. The Internet, TCP/IP, and LANs
B. Ping sweeps, port scans, traceroutes, & OS fingerprinting
C. An introduction to security issues  ➦ We’re Here!
   a. Attack Points
      – Human access
      – Physical access
      – LAN access
      – Wireless access
      – Remote (Internet) access
   b. Scanning your site
   c. The law