Network Attacks, Part 1

CS 161: Computer Security
Prof. Vern Paxson

TAs: Jethro Beekman, Mobin Javed, Antonio Lupher, Paul Pearce & Matthias Vallentin

http://inst.eecs.berkeley.edu/~cs161/

February 7, 2013
Announcements / Game Plan

• Homework #1 out now, due next week (Friday 2/15, 10:00PM)
  – Turn in electronically
• We expect Project #1 to ship around the end of next week

• Goal for today: a look at network attacks
  – With a focus on network layers 1-4
• To start: what general goals do we have communication/network security?
**General Communication Security Goals: CIA**

- **Confidentiality:**
  - No one can *read* our data / communication unless we want them to

- **Integrity**
  - No one can *manipulate* our data / processing / communication unless we want them to

- **Availability**
  - We can *access* our data / conduct our processing / use our communication capabilities when we want to

- **Also:** no *additional* traffic other than ours …
Layers 1 & 2: General Threats?

Framing and transmission of a collection of bits into individual messages sent across a single “subnetwork” (one physical technology)

Encoding bits to send them over a single physical link e.g. patterns of voltage levels / photon intensities / RF modulation
Physical/Link-Layer Threats: Eavesdropping

- Also termed *sniffing*
- For subnets using *broadcast* technologies (e.g., WiFi, some types of Ethernet), get it for “free”
  - Each attached system ’s NIC (= Network Interface Card) can capture any communication on the subnet
  - Some handy tools for doing so
    - tcpdump / windump (low-level ASCII printout)
TCPDUMP: Packet Capture & ASCII Dumper

demo 2 % tcpdump -r all.trace2
reading from file all.trace2, link-type EN10MB (Ethernet)
21:39:37.772565 IP 10.0.1.9.62137 > all-systems.mcast.net.canon-bjnp2: UDP, length 16
523449627, win 65535, options [mss 1460,nop,wscale 3,nop,nop,TS val 429017455 ecr 0,sack
OK,eol], length 0
3585654832, ack 2523449628, win 14480, options [mss 1460,sackOK,TS val 1765826995 ecr 42
9017455,nop,wscale 9], length 0
, win 65535, options [nop,nop,TS val 429017456 ecr 1765826995], length 0
1:525, ack 1, win 65535, options [nop,nop,TS val 429017456 ecr 1765826995], length 524
25, win 31, options [nop,nop,TS val 1765827012 ecr 429017456], length 0
1:535, ack 525, win 31, options [nop,nop,TS val 1765827083 ecr 429017456], length 534
35, win 65535, options [nop,nop,TS val 429017457 ecr 1765827083], length 0
21:39:44.838031 IP 10.0.1.9.54277 > 10.0.1.255.canon-bjnp2: UDP, length 16
21:39:44.838213 IP 10.0.1.9.62896 > all-systems.mcast.net.canon-bjnp2: UDP, length 16
Physical/Link-Layer Threats: *Eavesdropping*

- Also termed *sniffing*

- For subnets using **broadcast** technologies (e.g., WiFi, some types of Ethernet), get it for “free”
  - Each attached system’s NIC (= Network Interface Card) can capture any communication on the subnet
  - Some handy tools for doing so
    - tcpdump / windump (low-level ASCII printout)
    - Wireshark (GUI for displaying 800+ protocols)
Wireshark: GUI for Packet Capture/Exam.
Wireshark: GUI for Packet Capture/Exam.
Wireshark: GUI for Packet Capture/Exam.

<table>
<thead>
<tr>
<th>No</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Length</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0000000</td>
<td>10.0.1.9</td>
<td>10.0.1.255</td>
<td>BJNP</td>
<td>58</td>
<td>Printer Command: Unknown code (2)</td>
</tr>
<tr>
<td>2</td>
<td>0.000198</td>
<td>10.0.1.9</td>
<td>224.0.1.1</td>
<td>BJNP</td>
<td>58</td>
<td>Printer Command: Unknown code (2)</td>
</tr>
<tr>
<td>3</td>
<td>2.150663</td>
<td>10.0.1.9</td>
<td>255.255.255</td>
<td>DB-LSP-D</td>
<td>172</td>
<td>Dropbox LAN sync Discovery Protocol</td>
</tr>
<tr>
<td>4</td>
<td>2.150698</td>
<td>10.0.1.9</td>
<td>10.0.1.255</td>
<td>DB-LSP-D</td>
<td>172</td>
<td>Dropbox LAN sync Discovery Protocol</td>
</tr>
<tr>
<td>5</td>
<td>4.514403</td>
<td>10.0.1.13</td>
<td>31.13.75.23</td>
<td>TCP</td>
<td>78</td>
<td>61901 &gt; http [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=8 TVal=4296</td>
</tr>
<tr>
<td>6</td>
<td>4.536771</td>
<td>31.13.75.23</td>
<td>10.0.1.13</td>
<td>TCP</td>
<td>74</td>
<td>http &gt; 61901 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK Confirm=0</td>
</tr>
<tr>
<td>7</td>
<td>4.536895</td>
<td>31.13.75.23</td>
<td>10.0.1.13</td>
<td>TCP</td>
<td>66</td>
<td>61901 &gt; http [ACK] Seq=1 Ack=1 Win=524280 Len=0 TVal=429017456</td>
</tr>
<tr>
<td>8</td>
<td>4.537429</td>
<td>31.13.75.23</td>
<td>10.0.1.13</td>
<td>HTTP</td>
<td>550</td>
<td>GET / HTTP/1.1</td>
</tr>
<tr>
<td>9</td>
<td>4.558947</td>
<td>31.13.75.23</td>
<td>10.0.1.13</td>
<td>TCP</td>
<td>66</td>
<td>http &gt; 61901 [ACK] Seq=1 Ack=525 Win=15872 Len=0 TVal=1765827015</td>
</tr>
<tr>
<td>10</td>
<td>4.626447</td>
<td>31.13.75.23</td>
<td>10.0.1.13</td>
<td>HTTP</td>
<td>600</td>
<td>HTTP/1.1 302 Found</td>
</tr>
<tr>
<td>11</td>
<td>4.626579</td>
<td>10.0.1.13</td>
<td>31.13.75.23</td>
<td>TCP</td>
<td>66</td>
<td>61901 &gt; http [ACK] Seq=525 Ack=535 Win=524280 Len=0 TVal=429017456</td>
</tr>
<tr>
<td>12</td>
<td>7.665664</td>
<td>10.0.1.19</td>
<td>10.0.1.255</td>
<td>BJNP</td>
<td>58</td>
<td>Printer Command: Unknown code (2)</td>
</tr>
<tr>
<td>13</td>
<td>7.665846</td>
<td>10.0.1.19</td>
<td>224.0.0.1</td>
<td>BJNP</td>
<td>58</td>
<td>Printer Command: Unknown code (2)</td>
</tr>
</tbody>
</table>

Ethernet II, Src: Apple_ea:aa:41 (00:25:00:fe:aa:41), Dst: Apple_41:eb:00 (e4:ce:8f:41:eb:00)
Internet Protocol Version 4, Src: 31.13.75.23 (31.13.75.23), Dst: 10.0.1.13 (10.0.1.13)
Transmission Control Protocol, Src Port: http (80), Dst Port: 61901 (61901), Seq: 1, Ack: 525, Len: 534

Hypertext Transfer Protocol
HTTP/1.1 302 Found
Location: https://www.facebook.com/
P3P: CP="Facebook does not have a P3P policy. Learn why here: http://fb.me/p3p";
Set-Cookie: highContrast=deleted; expires=Thu, 01-Jan-1970 00:00:00 GMT; path=/; domain=.facebook.com; httponly;
Set-Cookie: wd=deleted; expires=Thu, 01-Jan-1970 00:00:00 GMT; path=/; domain=.facebook.com; httponly;
Content-Type: text/html; charset=utf-8;
X-FB-Debug: 0S41ArTHbMLqsy+lACGmQyqZyr42objFOaJZG0qage;
Date: Thu, 07 Feb 2013 05:39:42 GMT;
Connection: keep-alive;
Content-Length: 0;

...
Physical/Link-Layer Threats: *Eavesdropping*

- Also termed *sniffing*

- For subnets using *broadcast* technologies (e.g., WiFi, some types of Ethernet), get it for “free”
  - Each attached system’s NIC (= Network Interface Card) can capture any communication on the subnet
  - Some handy tools for doing so
    - tcpdump / windump (low-level ASCII printout)
    - Wireshark (GUI for displaying 800+ protocols)
    - Bro (scriptable real-time network analysis)

- For any technology, routers (and internal “switches”) can look at / export traffic they forward

- You can also “tap” a link
  - Insert a device to mirror physical signal
  - Or: just steal it!
Stealing Photons

1. Micro-bend clamping device
2. Optical photo detector
3. Optical-electrical converter
4. Laptop

- Jacket
- Cladding
- Active fiber optic cable
- Lost light <1%
Operation Ivy Bells

By Matthew Carle
Military.com

At the beginning of the 1970's, divers from the specially-equipped submarine, USS Halibut (SSN 587), left their decompression chamber to start a bold and dangerous mission, code named "Ivy Bells".

The Regulus guided missile submarine, USS Halibut (SSN 587) which carried out Operation Ivy Bells.

In an effort to alter the balance of Cold War, these men scoured the ocean floor for a five-inch diameter cable carry secret Soviet communications between military bases.

The divers found the cable and installed a 20-foot long listening device on the cable, designed to attach to the cable without piercing the casing, the device recorded all communications that occurred. If the cable malfunctioned and the Soviets raised it for repair, the bug, by design, would fall to the bottom of the ocean. Each month Navy divers retrieved the recordings and installed a new set of tapes.

Upon their return to the United States, intelligence agents from the NSA analyzed the recordings and tried to decipher any encrypted information. The Soviets apparently were confident in the security of their communications lines, as a surprising amount of sensitive information traveled through the lines without encryption.

prison. The original tap that was discovered by the Soviets is now on exhibit at the KGB museum in Moscow.
Physical/Link-Layer Threats: Disruption

- With physical access to a subnetwork, attacker can
  - Overwhelm its signaling
    - E.g., jam WiFi’s RF
  - Send messages that violate the Layer-2 protocol’s rules
    - E.g., send messages > maximum allowed size, sever timing synchronization, ignore fairness rules

- Routers & switches can simply “drop” traffic

- There’s also the heavy-handed approach …
Sabotage attacks knock out phone service

Nanette Asimov, Ryan Kim, Kevin Fagan, Chronicle Staff Writers
Friday, April 10, 2009

(04-10) 04:00 PDT SAN JOSE --

Police are hunting for vandals who chopped fiber-optic cables and killed landlines, cell phones and Internet service for tens of thousands of people in Santa Clara, Santa Cruz and San Benito counties on Thursday.

The sabotage essentially froze operations in parts of the three counties at hospitals, stores, banks and police and fire departments that rely on 911 calls, computerized medical records, ATMs and credit and debit cards.

The full extent of the havoc might not be known for days, emergency officials said as they finished repairing the damage late Thursday.

Whatever the final toll, one thing is certain: Whoever did this is in a world of trouble if he, she or they get caught.

"I pity the individuals who have done this," said San Jose Police Chief Rob Davis.

Ten fiber-optic cables carrying were cut at four locations in the predawn darkness. Residential and business customers quickly found that telephone service was perhaps more laced into their everyday needs than they thought. Suddenly they couldn't draw out money, send text messages, check e-mail or Web sites, call anyone for help, or even check on friends or relatives down the road.

Several people had to be driven to hospitals because they were unable to summon ambulances. Many businesses lapsed into idleness for hours, without the ability to contact associates or customers.

More than 50,000 landline customers lost service - some were residential, others were business lines that needed the connections for ATMs, Internet and bank card transactions. One line alone could affect hundreds of users.

AT&T is now offering a $250,000 reward for information leading to the arrest of whoever is responsible for severing lines fiber optic cables in San Jose that left much of the area without phone or cell service Thursday.

John Britton of AT&T said the reward is the largest ever offered by the company.
Physical/Link-Layer Threats: Spoofing

• With physical access to a subnetwork, attacker can create any message they like
  – When with a bogus source address: spoofing

• When using a typical computer, may require root/administrator to have full freedom

• Particularly powerful when combined with eavesdropping
  – Because attacker can understand exact state of victim’s communication and craft their spoofed traffic to match it
  – Spoofing w/o eavesdropping = blind spoofing
Spoofing Considerations

• “On path” attackers can see victim’s traffic ⇒ spoofing is easy

• “Off path” attackers can’t see victim’s traffic
  – They have to resort to blind spoofing
  – Often must guess/infer header values to succeed
    o We then care about work factor: how hard is this
  – But sometimes they can just brute force
    o E.g., 16-bit value: just try all 65,536 possibilities!

• When we say an attacker “can spoof”, we usually mean “w/ reasonable chance of success”
Layer 3: General Threats?

IP = Internet Protocol

Bridges multiple “subnets” to provide end-to-end internet connectivity between nodes.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Application</td>
</tr>
<tr>
<td>4</td>
<td>Transport</td>
</tr>
<tr>
<td>3</td>
<td>(Inter)Network</td>
</tr>
<tr>
<td>2</td>
<td>Link</td>
</tr>
<tr>
<td>1</td>
<td>Physical</td>
</tr>
</tbody>
</table>

**IP Header Fields:**

- **Version:** 4-bit
- **Header Length:** 4-bit
- **Type of Service (TOS):** 8-bit
- **Total Length (Bytes):** 16-bit
- **Identification:** 16-bit
- **Flags:** 3-bit
- **Fragment Offset:** 13-bit
- **Time to Live (TTL):** 8-bit
- **Protocol:** 8-bit
- **Header Checksum:** 16-bit
- **Source IP Address:** 32-bit
- **Destination IP Address:** 32-bit

**Payload:**

**IP = Internet Protocol**
Network-Layer (IP) Threats

- Can set arbitrary source address
  - “Spoofing” - receiver has no idea who you are
  - Could be blind, or could be coupled w/ sniffing
  - Note: many attacks require two-way communication
    • So successful off-path/blind spoofing might not suffice

- Can set arbitrary destination address
  - Enables “scanning” - brute force searching for hosts

- Can send like crazy (flooding)
  - IP has no general mechanism for tracking overuse
  - IP has no general mechanism for tracking consent
  - Very hard to tell where a spoofed flood comes from!

- If attacker can manipulate routing, can bring traffic to themselves for eavesdropping (viewed as hard)
5 Minute Break

Questions Before We Proceed?
Layer 4: General Threats?

End-to-end communication between processes (TCP, UDP)

<table>
<thead>
<tr>
<th>Layer 4: General Threats?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
</tr>
<tr>
<td>Transport</td>
</tr>
<tr>
<td>(Inter)Network</td>
</tr>
<tr>
<td>Link</td>
</tr>
<tr>
<td>Physical</td>
</tr>
</tbody>
</table>

End-to-end communication between processes (TCP, UDP)

Table:

- Source port
- Destination port
- Sequence number
- Acknowledgment
- HdrLen
- Flags
- Advertised window
- Checksum
- Urgent pointer
- Options (variable)
- Data
### Layer 4: General Threats?

<table>
<thead>
<tr>
<th>Layer</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Application</td>
</tr>
<tr>
<td>4</td>
<td>Transport</td>
</tr>
<tr>
<td>3</td>
<td>(Inter)Network</td>
</tr>
<tr>
<td>2</td>
<td>Link</td>
</tr>
<tr>
<td>1</td>
<td>Physical</td>
</tr>
</tbody>
</table>

These plus IP addresses define a given connection.

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source port</td>
<td></td>
</tr>
<tr>
<td>Destination port</td>
<td></td>
</tr>
<tr>
<td>Sequence number</td>
<td></td>
</tr>
<tr>
<td>Acknowledgment</td>
<td></td>
</tr>
<tr>
<td>HdrLen</td>
<td>0</td>
</tr>
<tr>
<td>Flags</td>
<td></td>
</tr>
<tr>
<td>Advertised window</td>
<td></td>
</tr>
<tr>
<td>Checksum</td>
<td></td>
</tr>
<tr>
<td>Urgent pointer</td>
<td></td>
</tr>
<tr>
<td>Options (variable)</td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td></td>
</tr>
</tbody>
</table>
Layer 4: General Threats?

- Application
- Transport
- (Inter)Network
- Link
- Physical

Defines where this packet fits within the sender’s bytestream

<table>
<thead>
<tr>
<th>Source port</th>
<th>Destination port</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence number</td>
<td></td>
</tr>
<tr>
<td>Acknowledgment</td>
<td></td>
</tr>
<tr>
<td>HdrLen</td>
<td>0</td>
</tr>
<tr>
<td>Checksum</td>
<td>Urgent pointer</td>
</tr>
<tr>
<td>Options (variable)</td>
<td></td>
</tr>
</tbody>
</table>

Data
TCP Conn. Setup & Data Exchange

Client (initiator)
IP address 1.2.1.2, port 3344

Server
IP address 9.8.7.6, port 80

TCP Connection Setup & Data Exchange

<table>
<thead>
<tr>
<th>Client</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP address 1.2.1.2, port 3344</td>
<td>IP address 9.8.7.6, port 80</td>
</tr>
</tbody>
</table>

**First Packet (SYN)**
- **Client**: SrcA=1.2.1.2, SrcP=3344, DstA=9.8.7.6, DstP=80, SYN, Seq = x
- **Server**: SrcA=9.8.7.6, SrcP=80, DstA=1.2.1.2, DstP=3344, SYN+ACK, Seq = y, Ack = x+1

**Second Packet (SYN+ACK)**
- **Server**: SrcA=1.2.1.2, SrcP=3344, DstA=9.8.7.6, DstP=80, ACK, Ack = y+1
- **Client**: SrcA=1.2.1.2, SrcP=3344, DstA=9.8.7.6, DstP=80, ACK, Seq=x+1, Ack = y+1, Data="GET /login.html"

**Third Packet (ACK)**
- **Server**: SrcA=9.8.7.6, SrcP=80, DstA=1.2.1.2, DstP=3344, ACK, Seq = y+1, Ack = x+16, Data="200 OK ... <html> ..."
TCP Threat: Disruption

• Normally, TCP finishes ("closes") a connection by each side sending a FIN control message
  – Reliably delivered, since other side must ack

• But: if a TCP endpoint finds unable to continue (process dies; info from other "peer" is inconsistent), it abruptly terminates by sending a RST control message
  – Unilateral
  – Takes effect immediately (no ack needed)
  – Only accepted by peer if has correct* sequence number
<table>
<thead>
<tr>
<th>Source port</th>
<th>Destination port</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sequence number</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Acknowledgment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HdrLen</th>
<th>Flags</th>
<th>Advertised window</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Checksum</th>
<th>Urgent pointer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Options (variable)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>Source port</td>
</tr>
<tr>
<td>-------------</td>
</tr>
<tr>
<td>Sequence number</td>
</tr>
</tbody>
</table>

**Acknowledgment**

<table>
<thead>
<tr>
<th>HdrLen</th>
<th>RST</th>
<th>Advertised window</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Checksum</th>
<th>Urgent pointer</th>
</tr>
</thead>
</table>

**Options (variable)**

**Data**
Abrupt Termination

• A sends a TCP packet with RESET (RST) flag to B
  – E.g., because app. process on A crashed
  – (Could instead be that B sends a RST to A)

• Assuming that the sequence numbers in the RST fit with what B expects, That’s It:
  – B’s user-level process receives: ECONNRESET
  – No further communication on connection is possible
TCP Threat: Disruption

- Normally, TCP finishes ("closes") a connection by each side sending a FIN control message
  - Reliably delivered, since other side must ack

- But: if a TCP endpoint finds unable to continue (process dies; info from other "peer" is inconsistent), it abruptly terminates by sending a RST control message
  - Unilateral
  - Takes effect immediately (no ack needed)
  - Only accepted by peer if has correct* sequence number

- So: if attacker knows ports & sequence numbers, can disrupt any TCP connection
TCP RST Injection

Client (initiator)
IP address 1.2.1.2, port 3344

Server
IP address 9.8.7.6, port 80

Attacker
IP address 6.6.6.6, port N/A

Client dutifully removes connection

0

\[\text{SrcA}=1.2.1.2, \text{SrcP}=3344, \text{DstA}=9.8.7.6, \text{DstP}=80, \text{ACK}, \text{Seq}=x+1, \text{Ack} = y+1, \text{Data} = \text{GET /login.html}\]

\[\text{SrcA}=9.8.7.6, \text{SrcP}=80, \text{DstA}=1.2.1.2, \text{DstP}=3344, \text{RST}, \text{Seq} = y+1, \text{Ack} = x+16\]
TCP RST Injection

Client (initiator)
IP address 1.2.1.2, port 3344

Server
IP address 9.8.7.6, port 80

Client rejects since no active connection

Attacker
IP address 6.6.6.6, port N/A

SrcA=9.8.7.6, SrcP=80, DstA=1.2.1.2, DstP=3344, RST, Seq = y+1, Ack = x+16

SrcA=1.2.1.2, SrcP=3344, DstA=9.8.7.6, DstP=80, ACK, Seq=x+1, Ack = y+1, Data="GET /login.html"

SrcA=9.8.7.6, SrcP=80, DstA=1.2.1.2, DstP=3344, ACK, Seq = y+1, Ack = x+16, Data="200 OK ... <html> ..."
TCP Threat: Data Injection

- What about inserting data rather than disrupting a connection?
  - Again, all that’s required is attacker knows correct ports, seq. numbers
  - Receiver B is none the wiser!

- Termed TCP connection hijacking (or “session hijacking”)
  - A general means to take over an already-established connection!

- We are toast if an attacker can see our TCP traffic!
  - Because then they immediately know the port & sequence numbers
TCP Data Injection

**Client (initiator)**
IP address 1.2.1.2, port 3344

**Server**
IP address 9.8.7.6, port 80

---

**Attacker**
IP address 6.6.6.6, port N/A

---

Client dutifully processes as server’s response

---

**TCP Data Injection Diagram**

- **Client (initiator)**: IP address 1.2.1.2, port 3344
- **Server**: IP address 9.8.7.6, port 80
- **Attacker**: IP address 6.6.6.6, port N/A

TCP packets:

- **SrcA=1.2.1.2, SrcP=3344, DstA=9.8.7.6, DstP=80, ACK, Seq=x+1, Ack = y+1, Data="GET /login.html"**
- **SrcA=9.8.7.6, SrcP=80, DstA=1.2.1.2, DstP=3344, ACK, Seq = y+1, Ack = x+16, Data="200 OK … <poison> …"**

Client processes the server's response as intended.
**TCP Data Injection**

**Client (initiator)**
- IP address 1.2.1.2, port 3344

**Server**
- IP address 9.8.7.6, port 80

**Attacker**
- IP address 6.6.6.6, port N/A

**Packet Details**
- **Client (initiator)**:
  - Source IP: 1.2.1.2, Source Port: 3344,
  - Destination IP: 9.8.7.6, Destination Port: 80,
  - ACK: True, Sequence: x+1, Acknowledgment: y+1,
  - Data: "GET /login.html"

- **Server**:
  - Source IP: 9.8.7.6, Source Port: 80,
  - Destination IP: 1.2.1.2, Destination Port: 3344,
  - ACK: True, Sequence: y+1, Acknowledgment: x+16,
  - Data: "200 OK ...

- **Attacker**:
  - Source IP: 9.8.7.6, Source Port: 80,
  - Destination IP: 1.2.1.2, Destination Port: 3344,
  - ACK: True, Sequence: y+1, Acknowledgment: x+16,
  - Data: "200 OK ...

- **Client** ignores since already processed that part of bytestream

---

34
TCP Threat: Blind Spoofing

• Is it possible for an attacker to inject into a TCP connection even if they can’t see our traffic?

• **YES**: if somehow they can infer or guess the port and sequence numbers

• Let’s look at a simpler related attack where the goal of the attacker is to create a fake connection, rather than inject into a real one
  – Why?
  – Perhaps to leverage a server’s trust of a given client as identified by its IP address
  – Perhaps to frame a given client so the attacker’s actions during the connections can’t be traced back to the attacker
Spoofing an Entire TCP Connection

**Alleged Client (not actual)**
- IP address 1.2.1.2, port N/A

**Server**
- IP address 9.8.7.6, port 80

**Blind Attacker**
- SrcA=1.2.1.2, SrcP=5566,
  DstA=9.8.7.6, DstP=80, SYN, Seq = z

**Attacker’s goal:**
- SrcA=9.8.7.6, SrcP=80,
  DstA=1.2.1.2, DstP=5566, SYN+ACK, Seq = y, Ack = z+1

- SrcA=1.2.1.2, SrcP=5566, DstA=9.8.7.6,
  DstP=80, ACK, Seq = z+1, ACK = y+1

- SrcA=1.2.1.2, SrcP=5566, DstA=9.8.7.6,
  DstP=80, ACK, Seq = z+1, ACK = y+1,
  Data = “GET /transfer-money.html”
Spoofing an Entire TCP Connection

Alleged Client (not actual)
IP address 1.2.1.2, port NA

Server
IP address 9.8.7.6, port 80

Blind Attacker

SrcA=1.2.1.2, SrcP=5566,
DstA=9.8.7.6, DstP=80, SYN, Seq = z

SrcA=9.8.7.6, SrcP=80,
DstA=1.2.1.2, DstP=5566, SYN+ACK, Seq = y, Ack = x+1

Small Note #1: if client receives this, will be confused ⇒ send a RST back to server …
… So attacker may need to hurry!
Spoofing an Entire TCP Connection

Alleged Client (not actual)
IP address 1.2.1.2, port NA

Server
IP address 9.8.7.6, port 80

Blind Attacker
SrcA=1.2.1.2, SrcP=5566,
DstA=9.8.7.6, DstP=80, SYN, Seq = z

Big Note #2: attacker doesn’t get to see this packet!
Spoofing an Entire TCP Connection

**Alleged Client (not actual)**
IP address 1.2.1.2, port **N/A**

**Server**
IP address 9.8.7.6, port **80**

Blind Attacker

- **SrcA**=1.2.1.2, **SrcP**=5566, **DstA**=9.8.7.6, **DstP**=80, **SYN**, **Seq** = **z**

So how can the attacker figure out what value of **y** to use for their **ACK**?

- **SrcA**=9.8.7.6, **SrcP**=80, **DstA**=1.2.1.2, **DstP**=5566, **SYN+ACK**, **Seq** = **y**, **Ack** = **z+1**

- **SrcA**=1.2.1.2, **SrcP**=5566, **DstA**=9.8.7.6, **DstP**=80, **ACK**, **Seq** = **z+1**, **ACK** = **y+1**

- **SrcA**=1.2.1.2, **SrcP**=5566, **DstA**=9.8.7.6, **DstP**=80, **ACK**, **Seq** = **z+1**, **ACK** = **y+1**, **Data** = “GET /transfer-money.html”
Reminder: Establishing a TCP Connection

Each host tells its *Initial Sequence Number* (ISN) to the other host.

(Spec says to pick based on local clock)

Hmm, any way for the attacker to know *this*?

How Do We Fix This?

Use a (Pseudo)-Random ISN

Sure - make a non-spoofed connection *first*, and see what server used for ISN *y* then!
## Summary of TCP Security Issues

- An attacker who can observe your TCP connection can manipulate it:
  - Forcefully **terminate** by forging a RST packet
  - **Inject** (spoof) data into either direction by forging data packets
  - Works because they can include in their spoofed traffic the correct sequence numbers (both directions) and TCP ports
  - *Remains a major threat today*
Summary of TCP Security Issues

• An attacker who can observe your TCP connection can manipulate it:
  – Forcefully **terminate** by forging a RST packet
  – **Inject** (spoof) data into either direction by forging data packets
  – Works because they can include in their spoofed traffic the correct sequence numbers (both directions) and TCP ports
  – *Remains a major threat today*

• An attacker who can **predict** the ISN chosen by a server can “blind spoof” a connection to the server
  – Makes it appear that host ABC has connected, and has sent data of the attacker’s choosing, when in fact it hasn’t
  – *Undermines any security based on trusting ABC’s IP address*
  – Allows attacker to “frame” ABC or otherwise **avoid detection**
  – **Fixed** (mostly) today by choosing **random** ISNs