



Network vulnerabilities

INGI2347: COMPUTER SYSTEM SECURITY (Spring 2016)

Marco Canini

UCL
Université
catholique
de Louvain

Plan for today

Lecture 6

- Networking basics
- Networking attacks
 - Eavesdropping, sniffing
 - Spoofing
 - Session Hijacking
 - Cache poisoning
- Denial of Service (DoS)
 - SYN flooding



DARPA Internet Design Goals

"to develop an effective technique for multiplexed utilization of existing interconnected network" – D. Clark, The Design Philosophy of the DARPA Internet Protocols

■ Secondary goals:

- Tolerate loss of networks or gateways
- Support multiple types of communications service
- Accommodate a variety of networks
- Permit distributed management of its resources
- Be cost effective
- Permit host attachment with a low level of effort
- Used resources must be accountable

■ Primarily designed for a benign and trustworthy environment



Layering

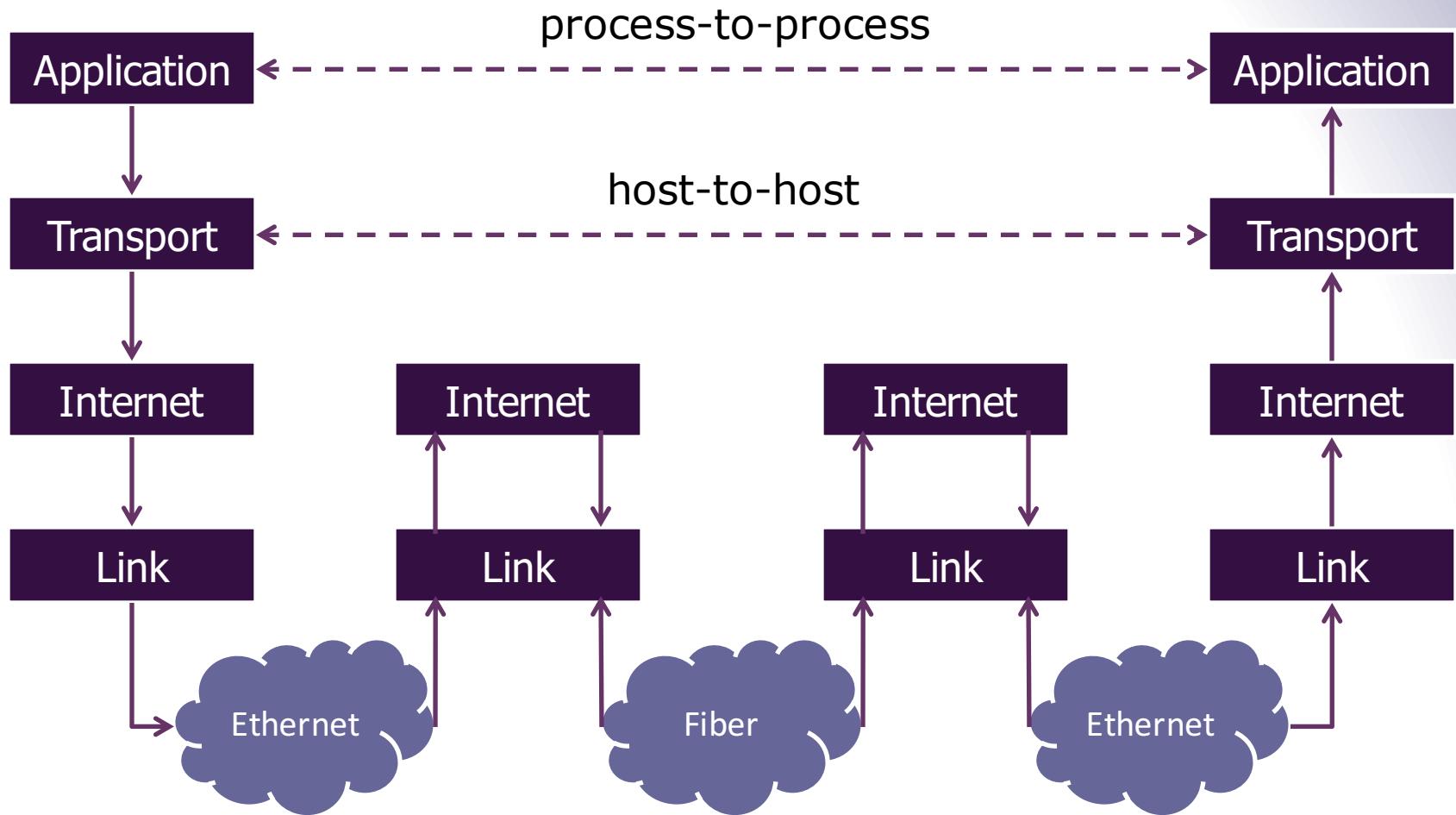
A result of abstraction
in network design

- Higher level services are implemented by using services at lower levels
- Advantages
 - Decompose problems
 - Modular changes

TCP/IP Layered Model	OSI Layered Model	
Application (e.g. SMTP, HTTP, Telnet)	Application	7
	Presentation	6
	Session	5
Transport (e.g. TCP, UDP)	Transport	4
Internet (e.g. IP, ARP, ICMP)	Network	3
Link (e.g. Ethernet, PPP, X25)	Data Link	2
	Physical	1

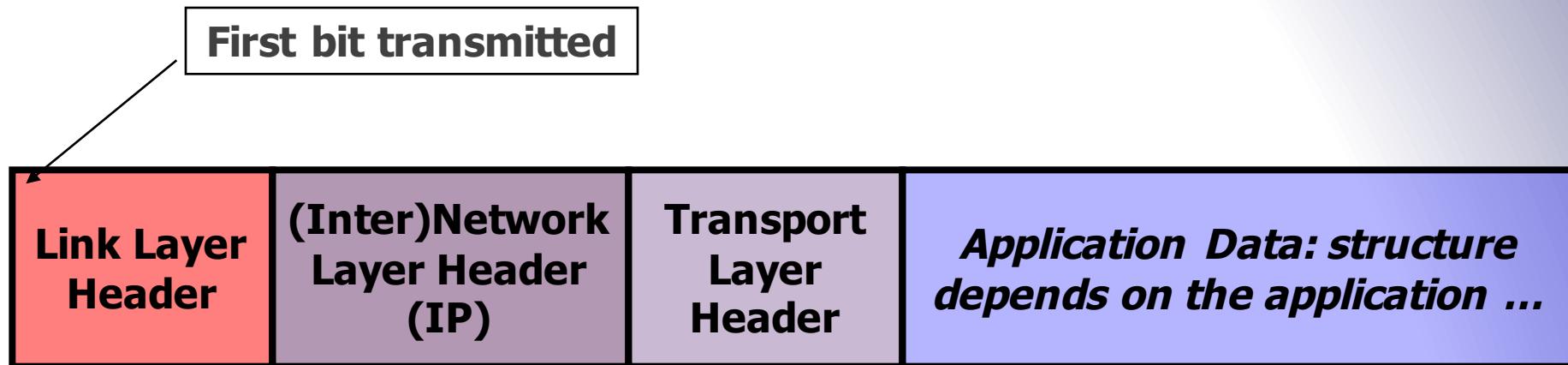


Data Flow





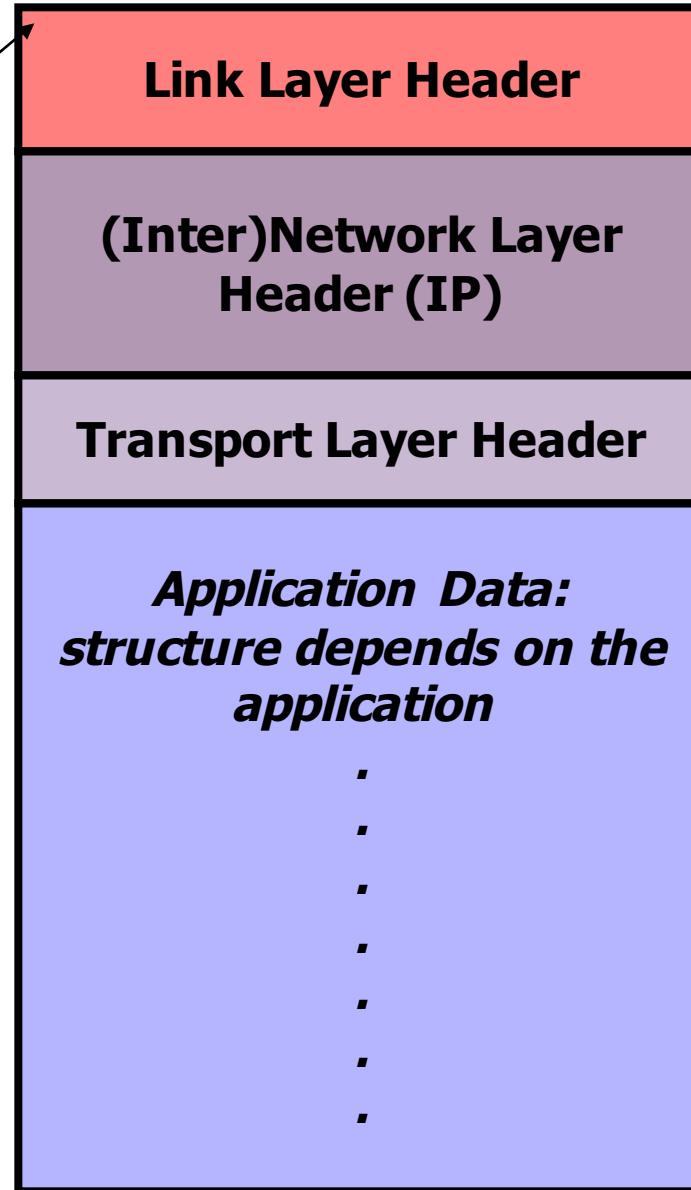
Horizontal View of a Single Packet





Vertical View of a Single Packet

First bit transmitted



Recall IP

How does the router know where to forward a packet?

- IP adopts a datagram approach
 - Every packet (datagram) contains the destination IP address
 - Packets can be sent at anywhere at any time
 - Best-effort delivery (unreliable)



Recall IP

0 4 8 16 19
31

Version	Hlen	TOS	Length	
Identification		Flags	Offset	
Time to Live	Protocol	Checksum		
Source Address				
Destination Address				
Options (variable length)				Pad

- Can you trust the contents?

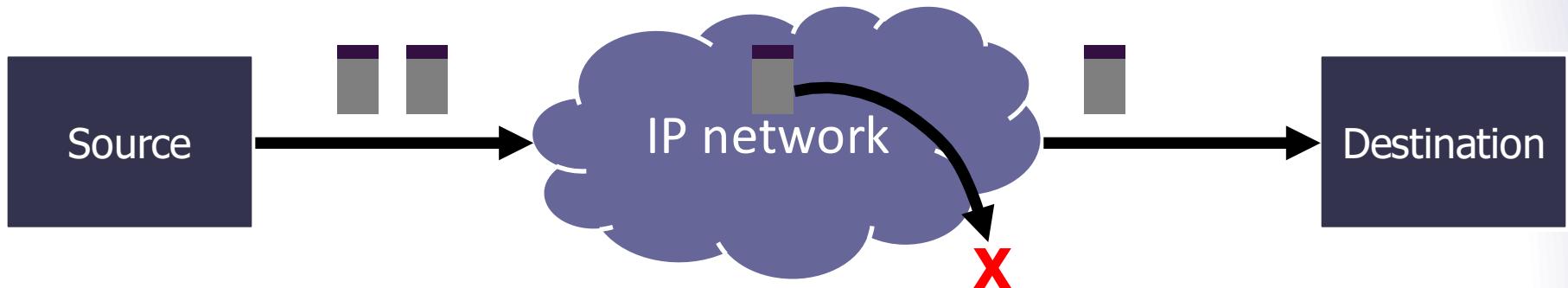
Recall IP

- Two IP addresses
 - Source IP address (32 bits)
 - Destination IP address (32 bits)
- Destination address
 - Unique **identifier/locator** for the receiving host
 - Allows each node to make forwarding decisions
- Source address
 - Unique identifier/locator for the sending host
 - Recipient can decide whether to accept packet
 - Enables recipient to send a reply back to source



IP: “Best Effort” Packet Delivery

- Routers inspect destination address, locate “next hop” in forwarding table
 - Address = ~unique **identifier/locator** for the receiving host
- Only provides a “ *I'll give it a try* ” delivery service:
 - Packets may be lost
 - Packets may be corrupted
 - Packets may be delivered out of order





Recall UDP

0
31

16



- Minimal message-oriented transport protocol
- Provide no guarantees for message delivery
- Port numbers to perform application demultiplexing
 - e.g., port 53 is DNS, port 2049 is NFS, port 138 is Netbios

Recall TCP

0 16 31

Source Port			Destination Port
Sequence Number			
Acknowledgment Number (if ACK set)			
Data Offset	0	Flags	Window Size
Checksum		Urgent Pointer (if URG set)	
Options (variable length)			Pad

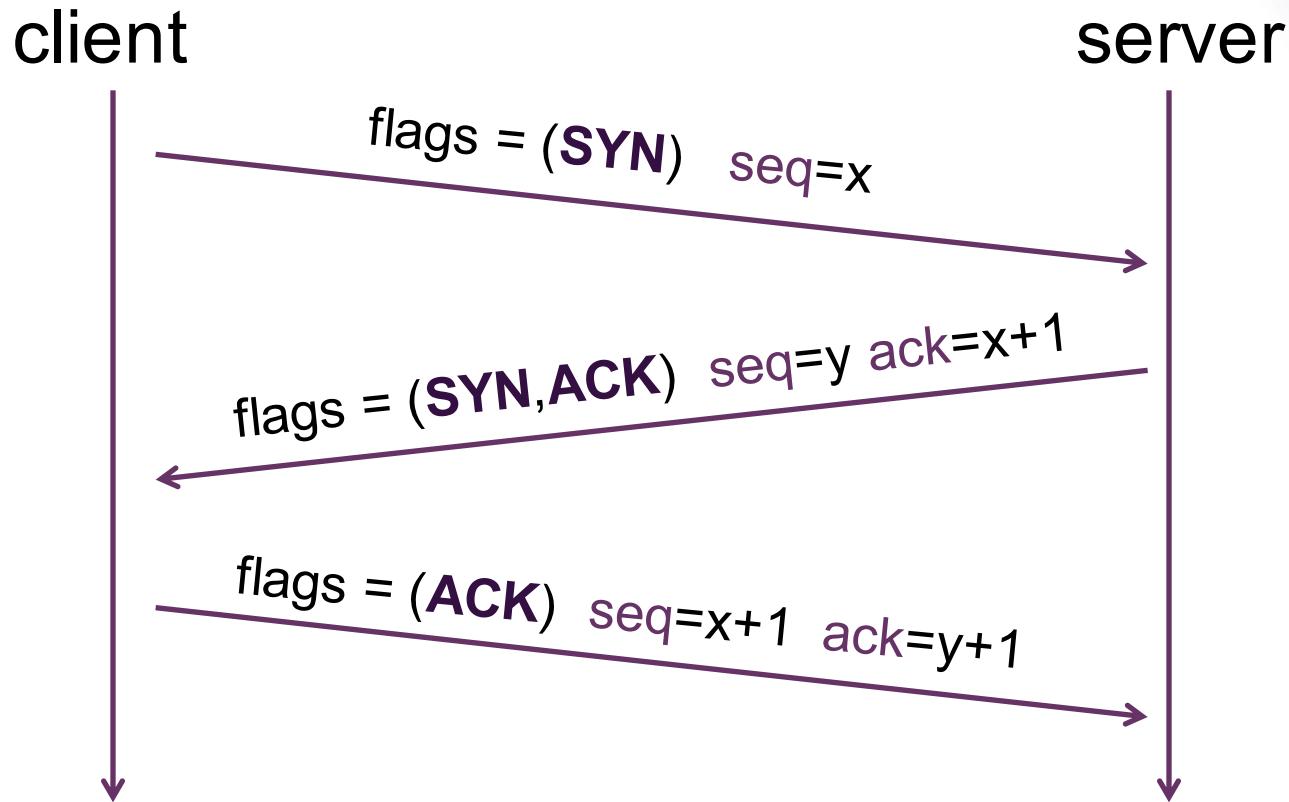
■ Reliable, in-order message delivery

- Sequence numbers identify the order of the bytes sent
- Ack number specify the sequence number of the next expected byte
- Sliding window flow control to not overflow receiver



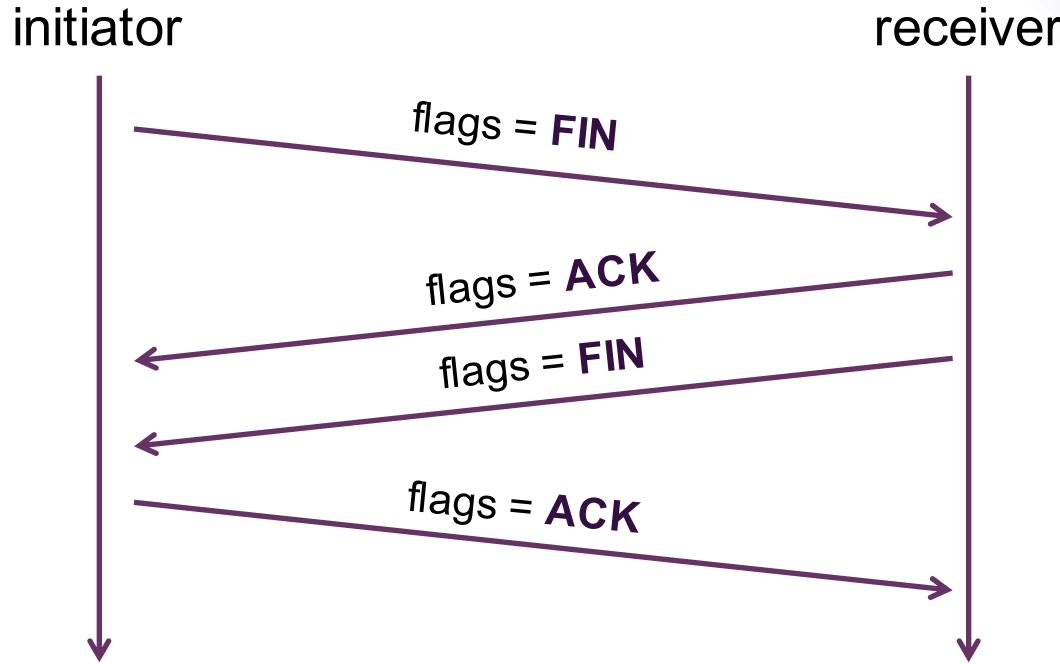
TCP Connection Establishment

- 3-way handshake: SYN ; SYN + ACK ; ACK





TCP Connection Termination



- Each side of the connection terminates independently
 - A connection can be “half-open”, in which case one side has terminated, but the other has not

Recall ARP (Address Resolution Protocol)

Problem:

- Need mapping between IP addresses and hardware addresses

■ Solution: ARP

- Every host maintains IP-HW address mapping table (cache)
- Timeout associate with cached info (15 min)

■ Sender

- Broadcasts “Who is IP address X?”
- Broadcast message includes sender’s IP and HW addresses

■ Receivers

- Any host with sender in cache “refreshes” timeout
- Host with IP address X replies “IP X has HW address Y”
- Target host adds sender (if not already in cache)



Mapping Names to Addresses

- Domain Name System (DNS)
 - Hierarchical name space divided into sub-trees ("zones")
 - Zones distributed over collection of DNS *name servers*
- Hierarchy of DNS servers
 - Root (**hardwired** into other servers)
 - Top-level domain (**TLD**) servers
 - "Authoritative" DNS servers (e.g. for *ucouvain.be*)

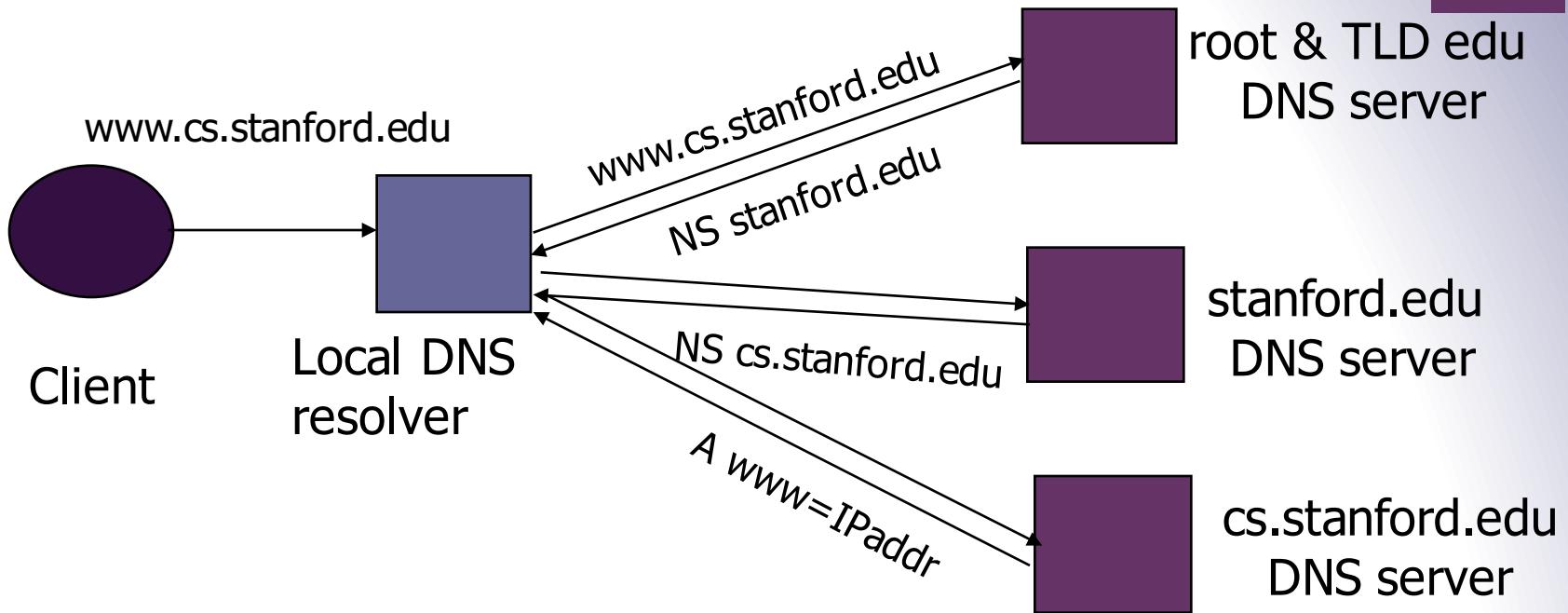


Mapping Names to Addresses

- Domain Name System (DNS)
 - Hierarchical name space divided into *zones*
 - Zones distributed over collection of DNS name servers
- Hierarchy of DNS servers
 - Root (hardwired into other servers)
 - Top-level domain (TLD) servers
 - “Authoritative” DNS servers (e.g. for *uclouvain.be*)
- Performing the translations
 - Each computer configured to contact a *resolver*



DNS Lookup Example



DNS record types (partial list):

- NS: name server (points to other server)
- A: address record (contains IP address)
- MX: address in charge of handling email
- TXT: generic text (e.g. used to distribute site public keys)

DNS Protocol

DNS protocol: *query* and *reply* messages,
both with **same message format**

(Mainly uses UDP transport rather than TCP)

Message header:

- **Identification:** 16 bit # for query, reply to query uses same #
- Replies can include “Authority” (name server responsible for answer) and “Additional” (info client is likely to look up soon anyway)
- Replies have a **Time To Live** (in seconds) for caching

<i>16 bits</i>	<i>16 bits</i>
Identification	Flags
# Questions	# Answer RRs
# Authority RRs	# Additional RRs
Questions (variable # of resource records)	
Answers (variable # of resource records)	
Authority (variable # of resource records)	
Additional information (variable # of resource records)	



Network attacks

General Communication Security

Goals: CIA

■ Confidentiality:

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

■ Integrity

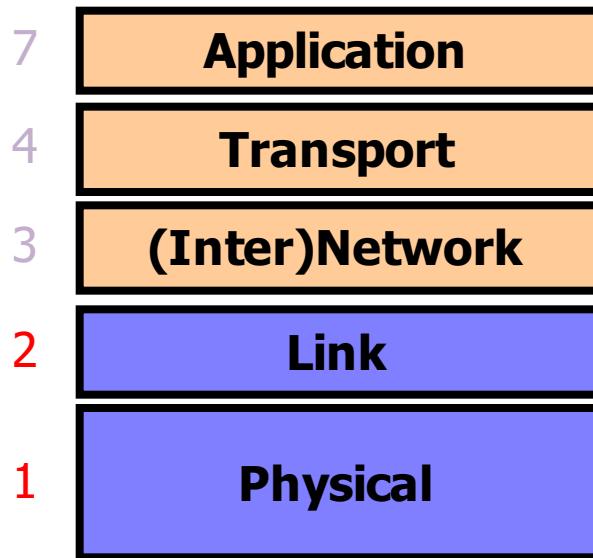
- No one can *manipulate* our data / processing / communication unless we want them to

■ 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)
 - Wireshark (GUI for displaying 800+ protocols)

Sniffing Tool: Wireshark

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.56.1	192.168.56.2	TCP	78	60922 > ftp [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=8 TSval=390581402
2	0.000018	192.168.56.2	192.168.56.1	TCP	74	ftp > 60922 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK=1 TSval=4294960214
3	0.000344	192.168.56.1	192.168.56.2	TCP	66	60922 > ftp [ACK] Seq=1 Ack=1 Win=131768 Len=0 TSval=390581402
4	0.002983	192.168.56.2	192.168.56.1	FTP	86	Response: 220 (vsFTPD 2.3.5)
5	0.003105	192.168.56.1	192.168.56.2	TCP	66	60922 > ftp [ACK] Seq=1 Ack=21 Win=131744 Len=0 TSval=39058140!
6	2.541423	192.168.56.1	192.168.56.2	FTP	80	Request: USER ftpuser
7	2.541444	192.168.56.2	192.168.56.1	TCP	66	ftp > 60922 [ACK] Seq=21 Ack=15 Win=14848 Len=0 TSval=4294960214
8	2.541663	192.168.56.2	192.168.56.1	FTP	100	Response: 331 Please specify the password.
9	2.542065	192.168.56.1	192.168.56.2	TCP	66	60922 > ftp [ACK] Seq=15 Ack=55 Win=131712 Len=0 TSval=3905839:
10	5.230322	192.168.56.1	192.168.56.2	FTP	79	Request: PASS secret

Internet Protocol Version 4, Src: 192.168.56.1 (192.168.56.1), Dst: 192.168.56.2 (192.168.56.2)
Transmission Control Protocol, Src Port: 60922 (60922), Dst Port: ftp (21), Seq: 15, Ack: 55, Len: 13
File Transfer Protocol (FTP)
PASS secret\r\n

Hex	Dec	ASCII
0010	00 41 dc 84 40 00 40 06	.A..@. @. l...8...
0020	38 02 ed fa 00 15 f3 e9	8..... .0.....
0030	08 0a 17 47 e0 f7 ff ff	@P..... .G....
0040	e4 67 50 41 53 53 20 73	.gPASS s ecret..

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)
- 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!

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 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

[PRINT](#) [E-MAIL](#) [SHARE](#) [COMMENTS \(477\)](#)

FONT | SIZE: [-](#) [+](#)

(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.

IMAGES



[+ View More Images](#)

MORE NEWS

- [Toyota seeks damage control, in public and private](#) 02.09.10
- [Snow shuts down federal government, life goes on](#) 02.09.10
- [Iran boosts nuclear enrichment, drawing warnings](#) 02.09.10

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.

Marco Cappiello © 2016



UPDATE



Vancouver's Venues Shine

Update

NEWS | LOCAL BEAT

\$250K Reward Out for Vandals Who Cut AT&T Lines

Local emergency declared during outage

By **LORI PREUITT**

Updated 2:12 PM PST, Fri, Apr 10, 2009

[PRINT](#) [EMAIL](#) [SHARE](#) [BUZZ UP!](#) [TWITTER](#) [FACEBOOK](#)



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.

10 Mar 2016

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**

ARP Spoofing

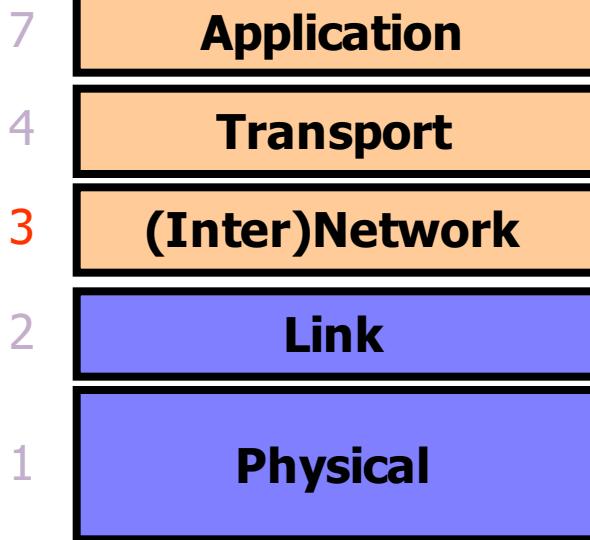
- Recall: ARP maps between IP addresses and hardware addresses
- ARP is very simple and insecure
 - client: who knows the Ethernet address of 1.2.3.4?
 - anybody: 1.2.3.4 has Ethernet address 01:02:03:04:05:06
- It is easy to forge responses (even non-solicited) to redirect traffic!

Spoofing Considerations

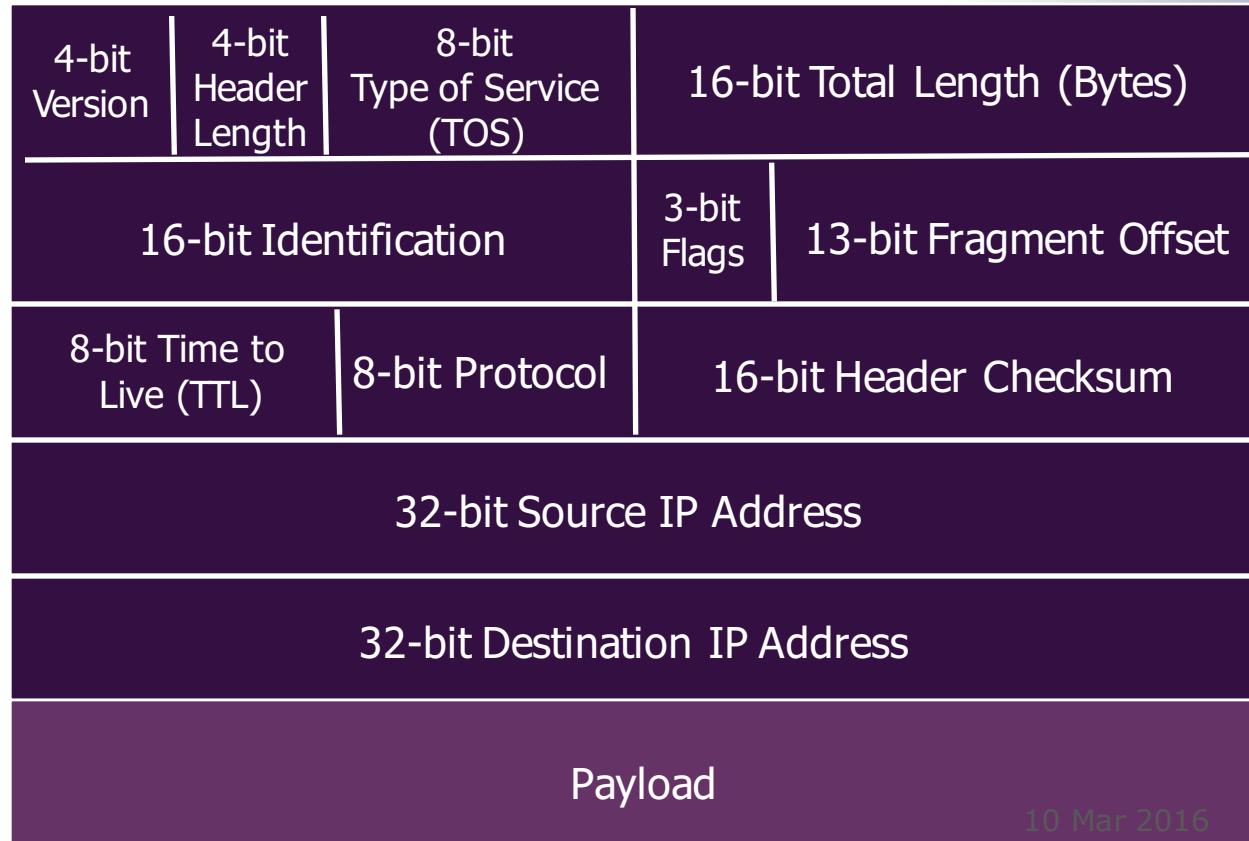
- “On path” attackers can see victim’s traffic ⇒ spoofing is easy
- “Off path” attackers can’t see victim’s traffic
 - They have to resort to blind spoofing
 - Often must guess/infer header values to succeed
 - We then care about work factor: how hard is this
 - But sometimes they can just brute force
 - E.g., 16-bit value: just try all 65,536 possibilities!
- When we say an attacker “can spoof”, we usually mean “with reasonable chance of success”



Layer 3: General Threats?



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

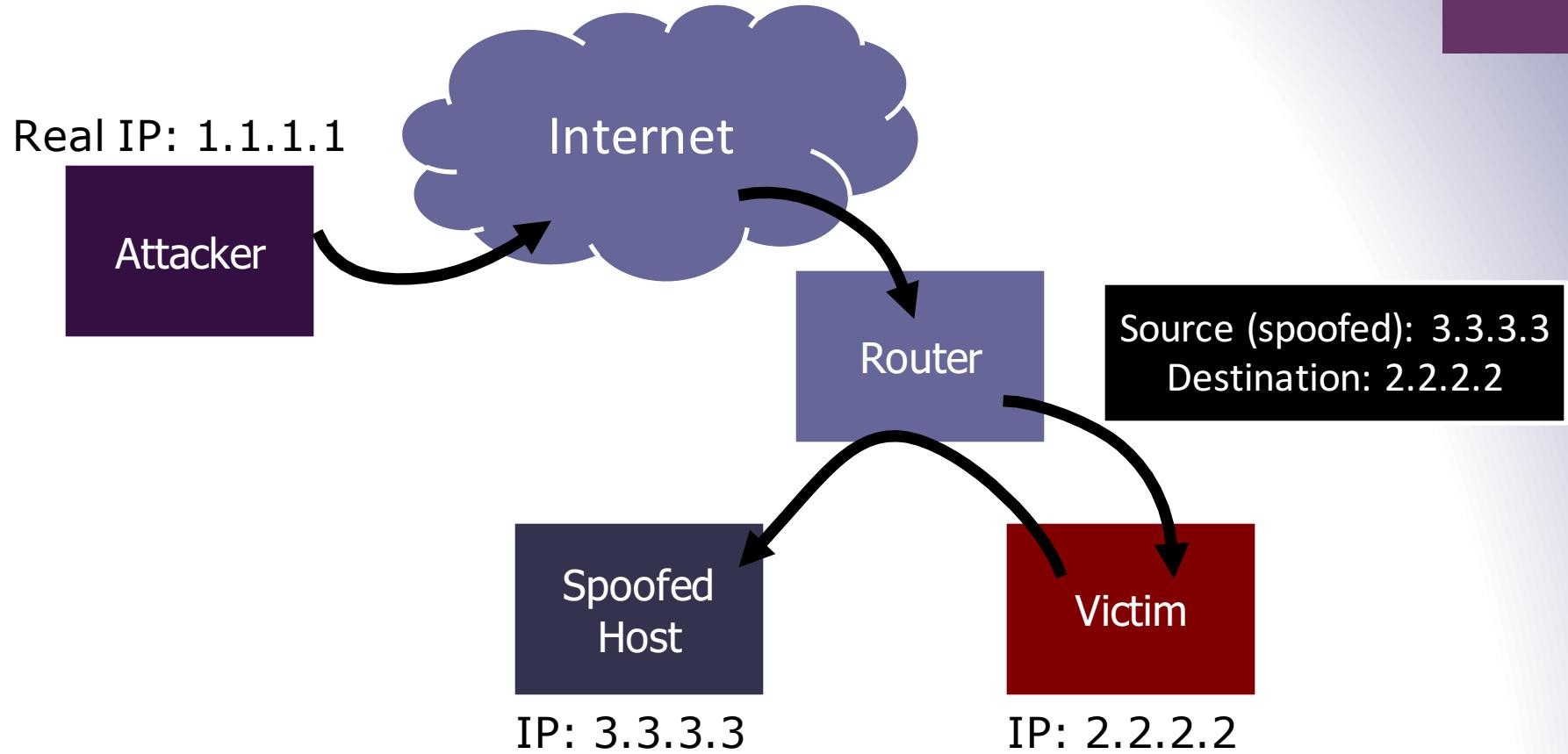


Network-Layer (IP) Threats

- Can set arbitrary source address
 - “**Spoofing**” – receiver has no idea who you are
 - Could be **blind**, or could be coupled with **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)



IP Spoofing Example

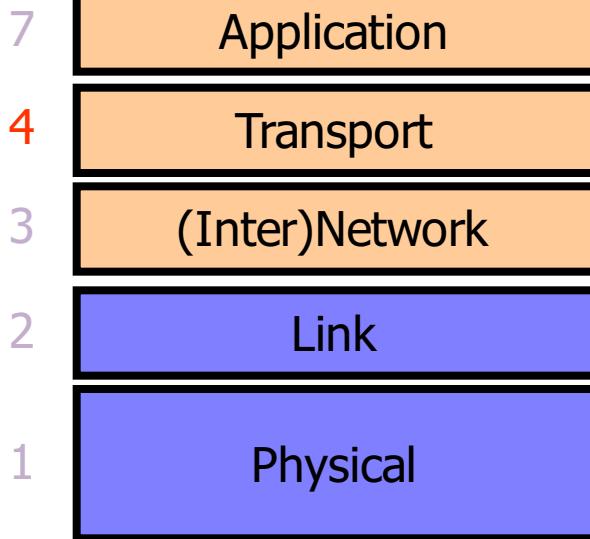


IP Spoofing

- The response to a forged message is sent to the forged source address
 - Spoofing mainly used when an attacker does not care about response
 - The LAN is a special case as the attacker can observe the response
- Easy to use with protocols based on UDP
- Spoofing a TCP connection is more difficult
 - Recall 3-way handshake: SYN ; SYN + ACK ; ACK



Layer 4: General Threats?

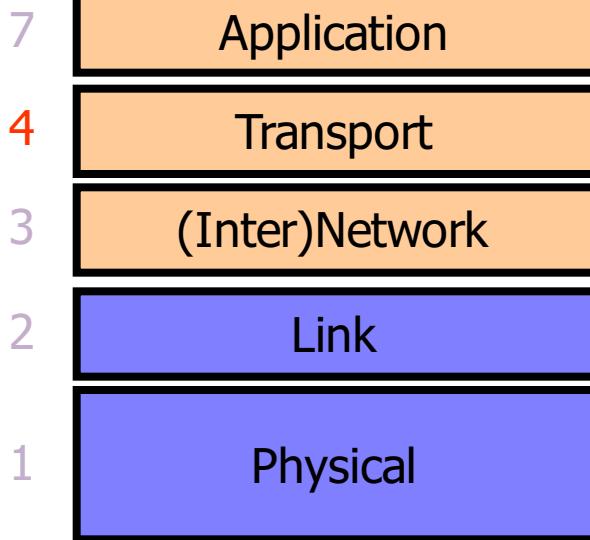


*End-to-end communication
between processes
(TCP, UDP)*

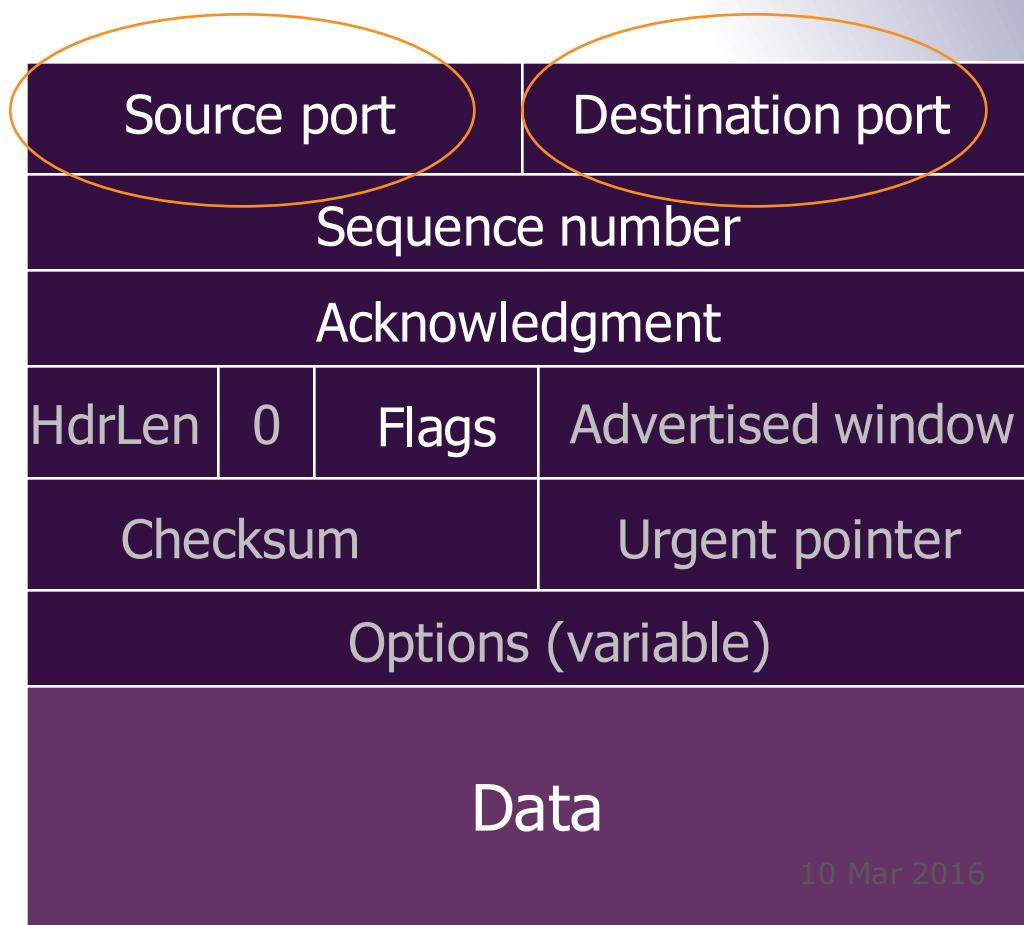
Source port	Destination port
Sequence number	
Acknowledgment	
HdrLen	0
Flags	Advertised window
Checksum	Urgent pointer
Options (variable)	
Data	



Layer 4: General Threats?

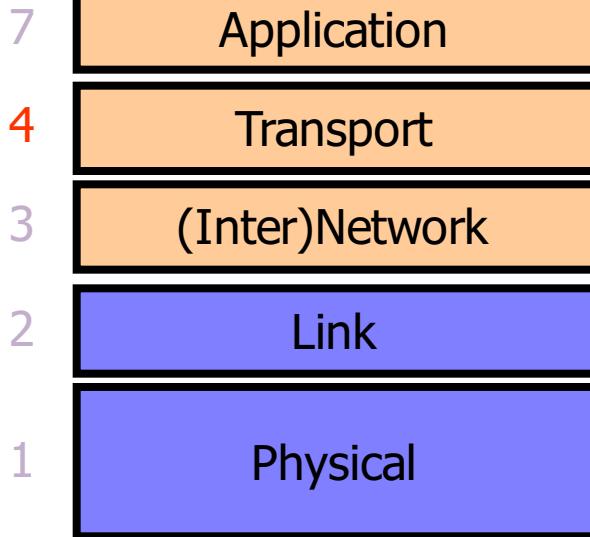


These plus IP addresses define a given connection

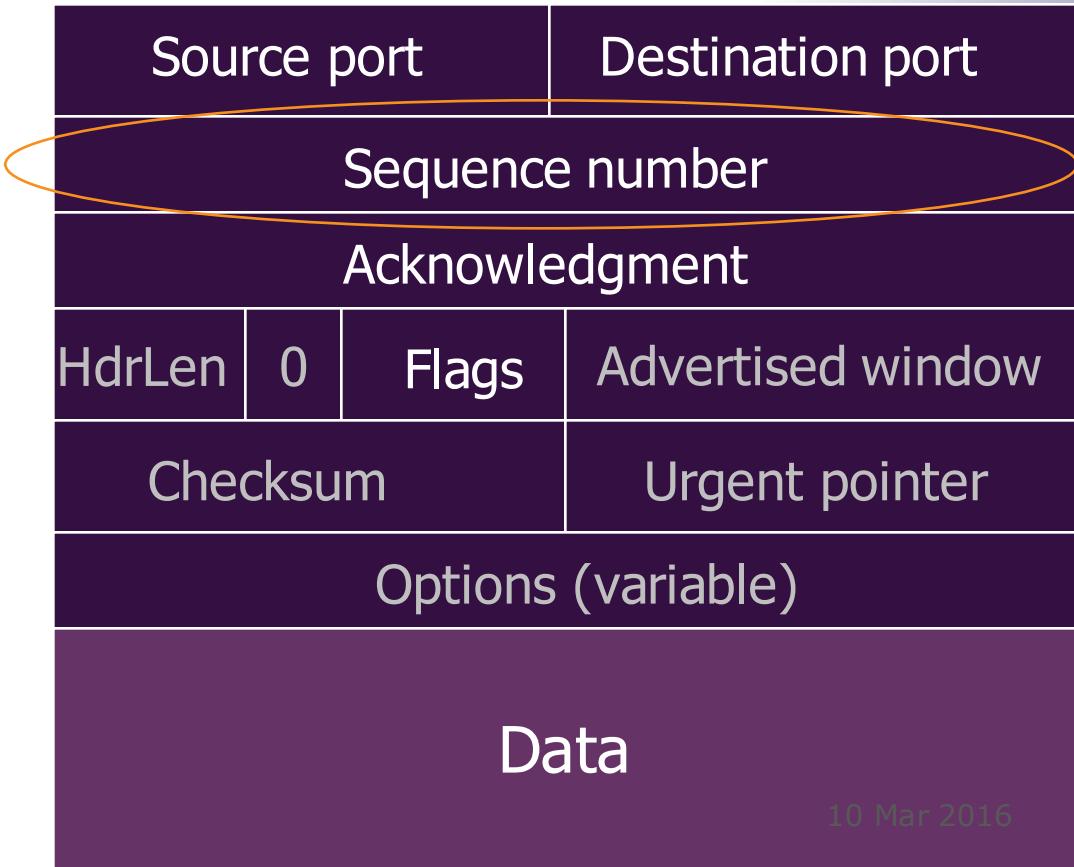




Layer 4: General Threats?



Defines where this packet fits within the sender's bytestream





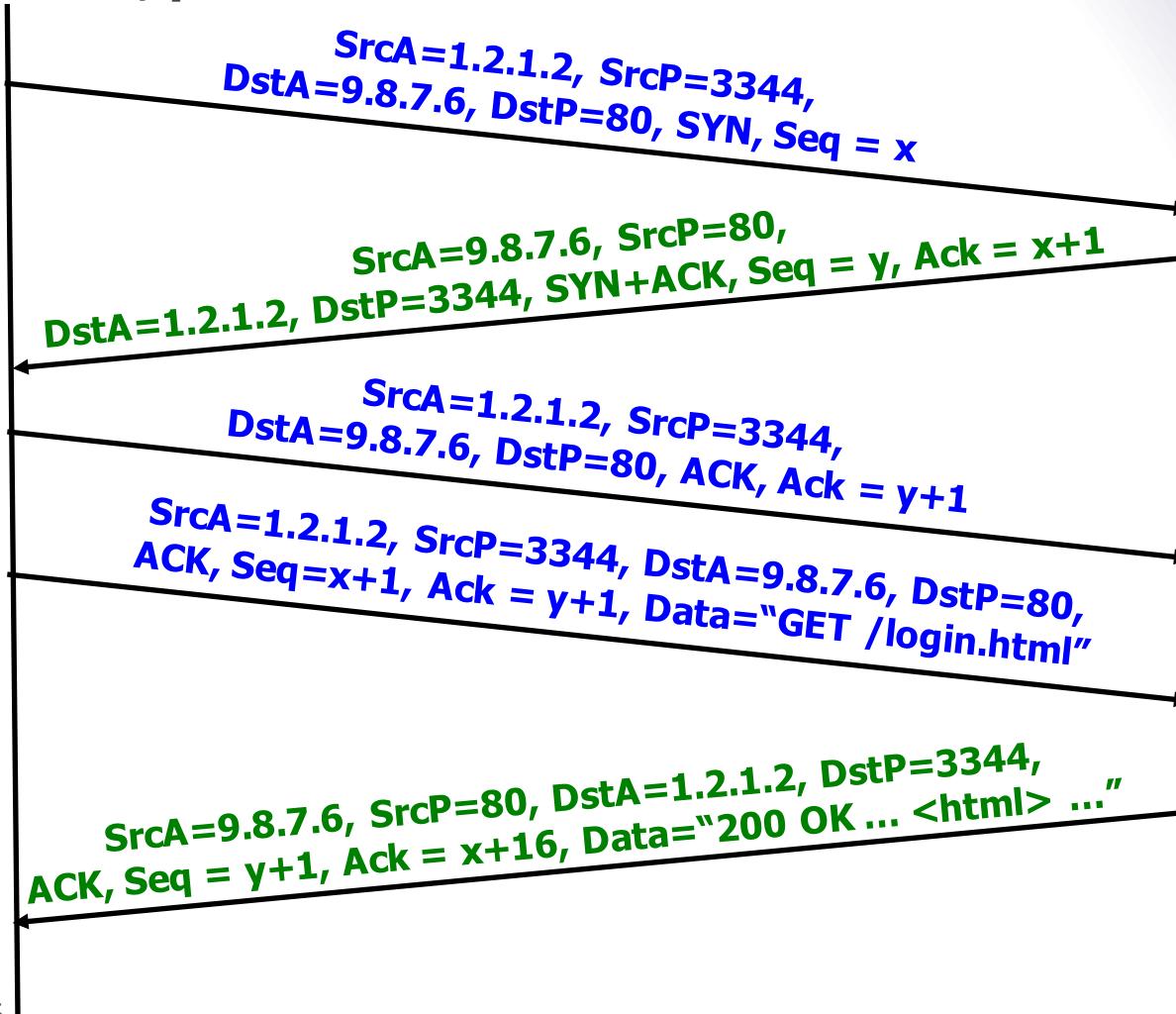
TCP Conn. Setup & Data Exchange

Client (initiator)

IP address 1.2.1.2, port 3344

Server

IP address 9.8.7.6, port 80



TCP Threat: Disruption

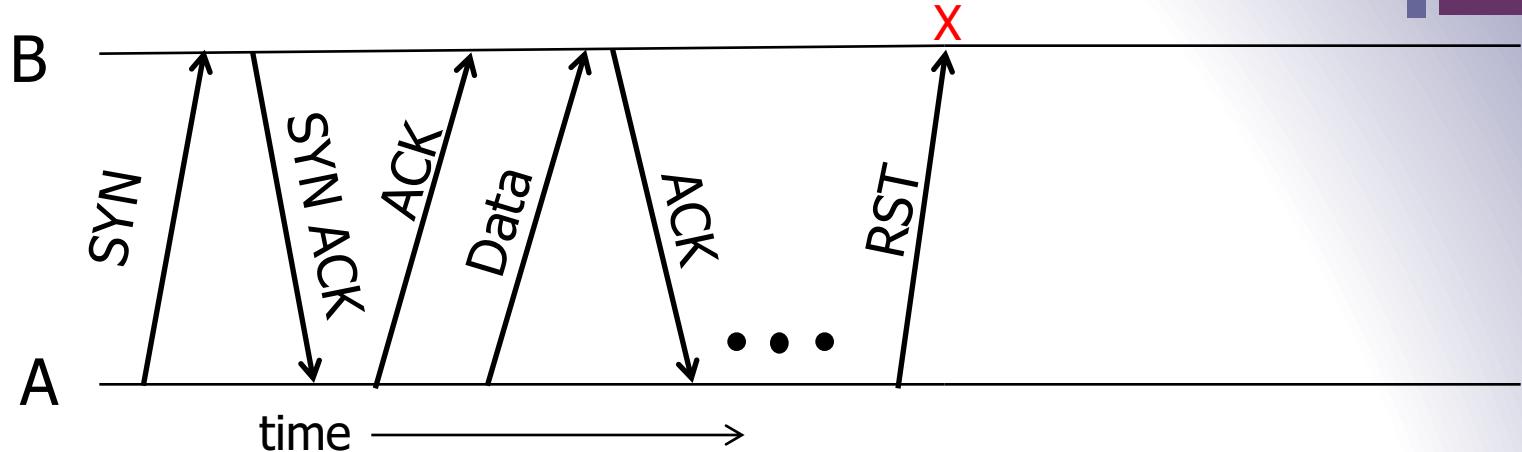
- Normally, TCP finishes (“closes”) a connection by each side sending a FIN control message
 - Reliably delivered, since other side must ack
- But: if a TCP endpoint finds unable to continue (process dies; info from other “peer” is inconsistent), it abruptly **terminates** by sending a RST control message
 - Unilateral
 - Takes effect immediately (no ack needed)
 - Only accepted by peer if has correct sequence number

Source port	Destination port
Sequence number	
Acknowledgment	
HdrLen	0
Flags	Advertised window
Checksum	Urgent pointer
Options (variable)	
Data	

Source port	Destination port
Sequence number	
Acknowledgment	
HdrLen	0
RST	Advertised window
Checksum	Urgent pointer
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

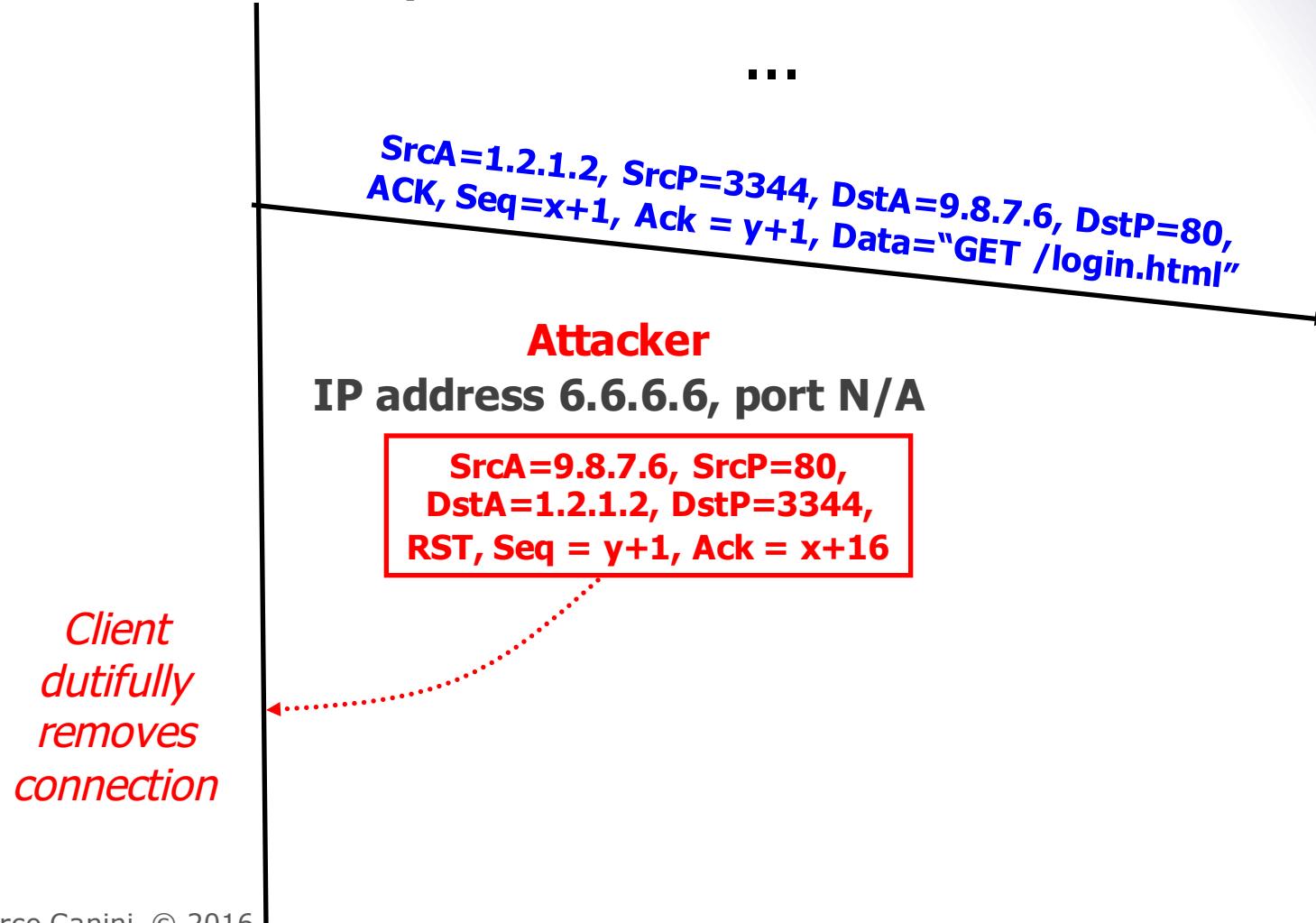
- Normally, TCP finishes (“closes”) a connection by each side sending a FIN control message
 - Reliably delivered, since other side must ack
- But: if a TCP endpoint finds unable to continue (process dies; info from other “peer” is inconsistent), it abruptly terminates by sending a RST control message
 - 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





TCP RST Injection

Client (initiator)
IP address 1.2.1.2, port 3344

Server
IP address 9.8.7.6, port 80

...

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"

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

*Client
rejects
since no
active
connection X*

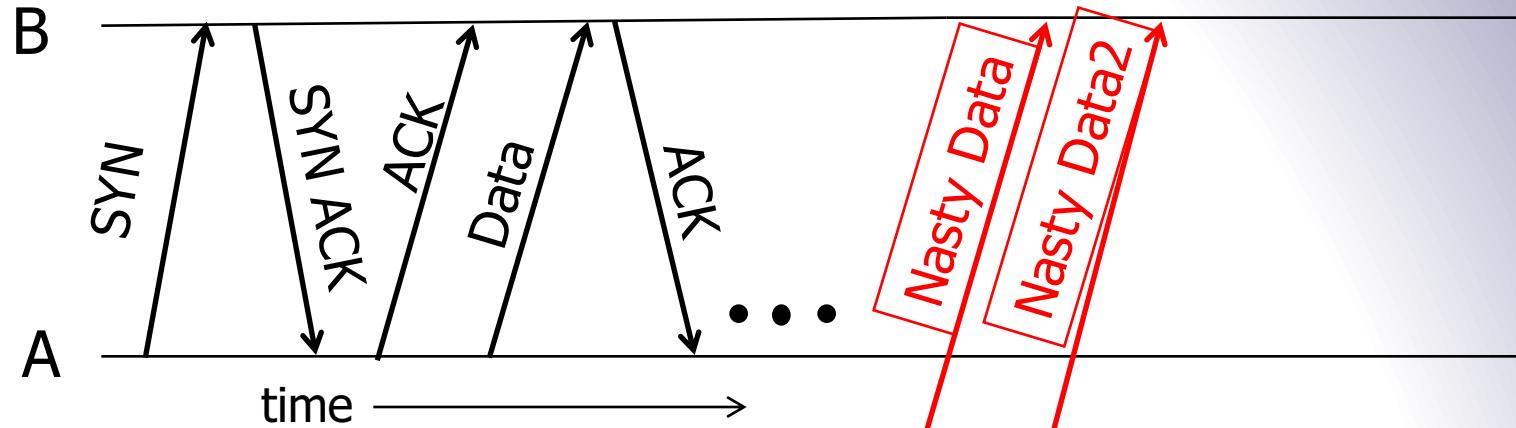
SrcA=9.8.7.6, SrcP=80, DstA=1.2.1.2, DstP=3344,
ACK, Seq = y+1, Ack = x+16, Data="200 OK ... <html> ..."

Example DoS: RST Injection

- Attacker sends a RST packet to an open socket at s
 - If correct SN_S then connection will close \Rightarrow DoS
 - Naively, success prob. is $1/2^{32}$ (32-bit seq. #'s).
 - ... but, many systems allow for a large window of acceptable seq. #. Much higher success probability.
 - Attacker can flood with RST packets until one works
- Most effective against long lived connections
 - E.g. BGP



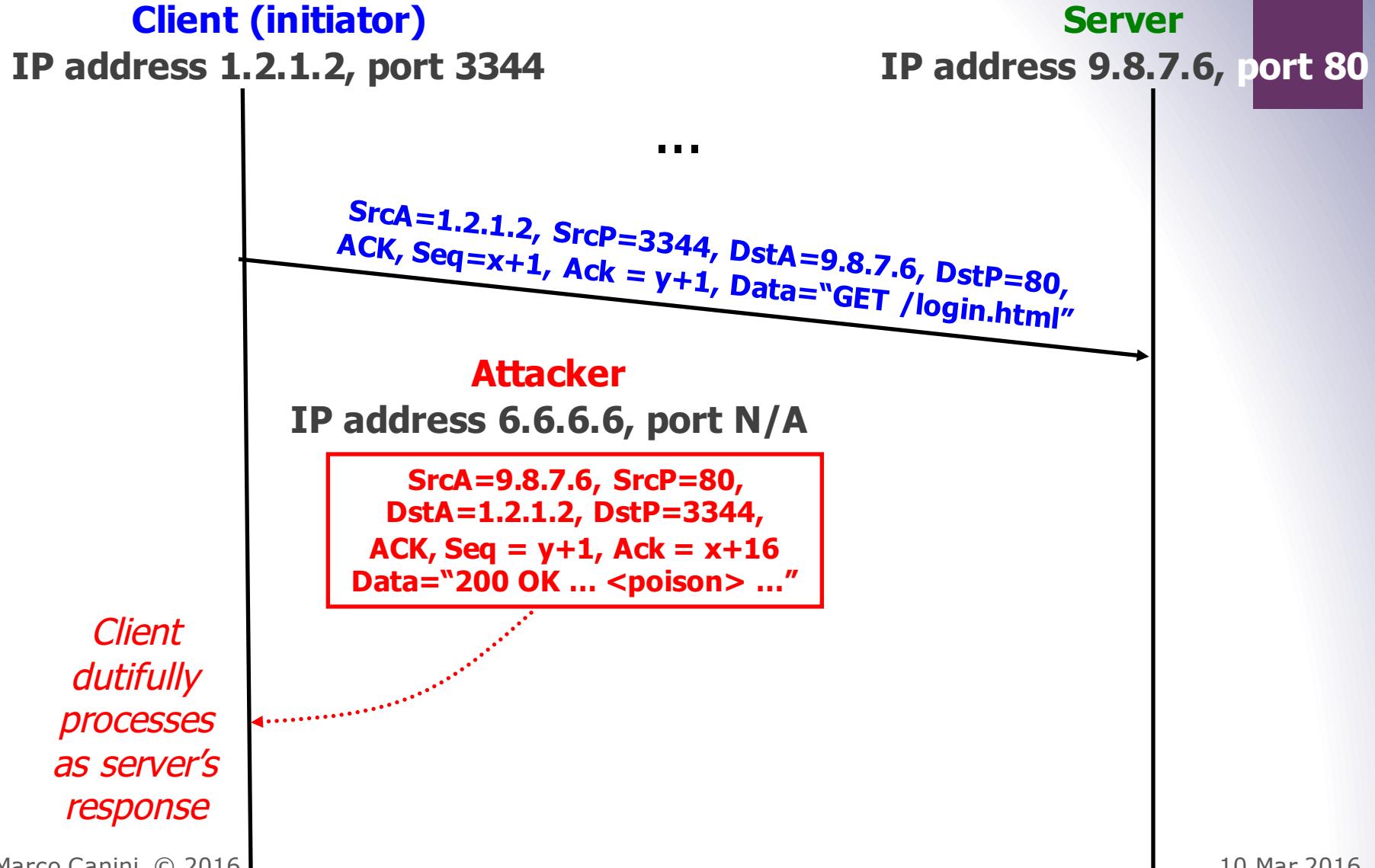
TCP Threat: Data Injection



- What about **inserting data** rather than disrupting a connection?
 - Again, all that's required is attacker knows correct ports, seq. numbers
 - Receiver B is *not any 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

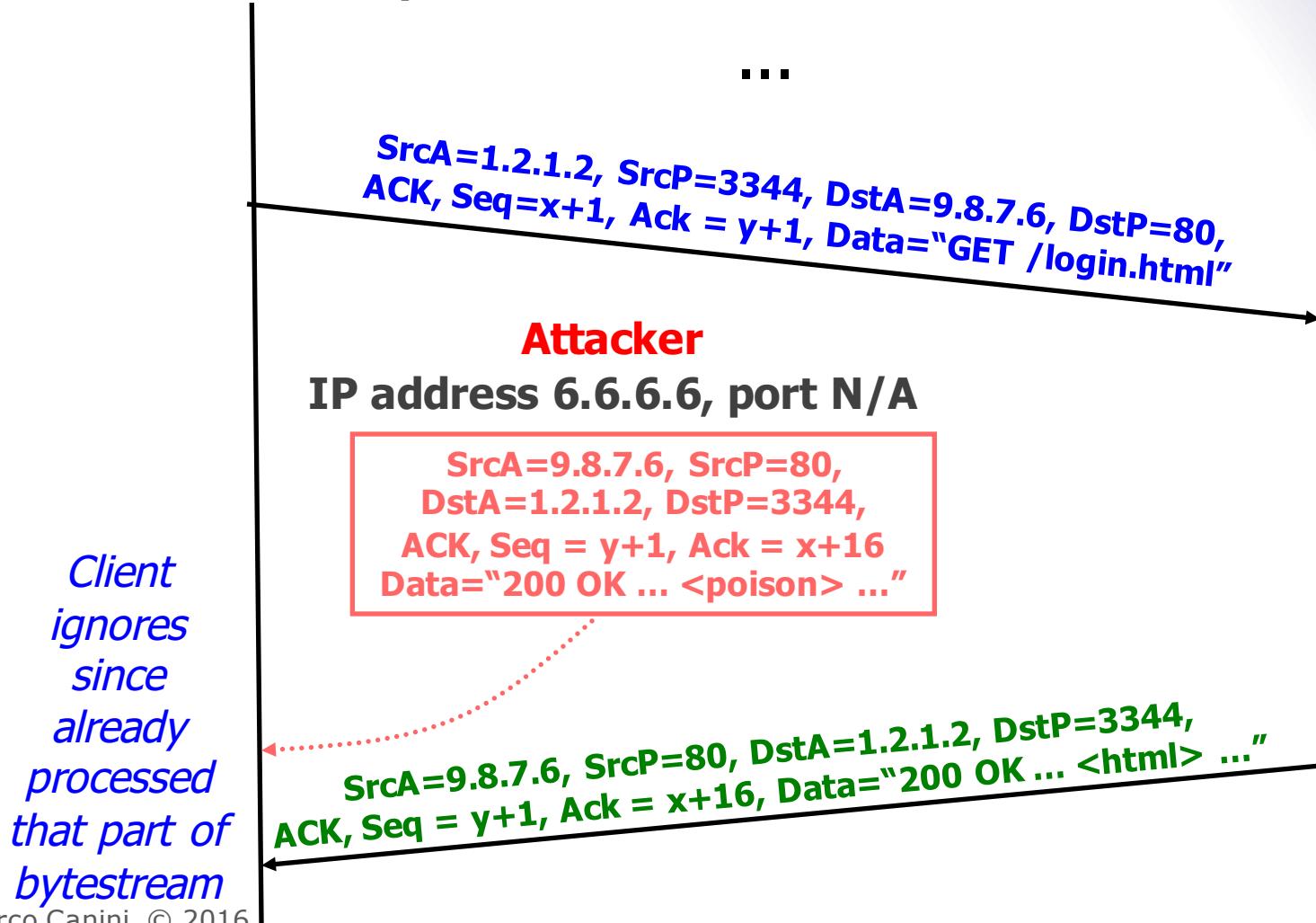




TCP Data Injection

Client (initiator)
IP address 1.2.1.2, port 3344

Server
IP address 9.8.7.6, port 80



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

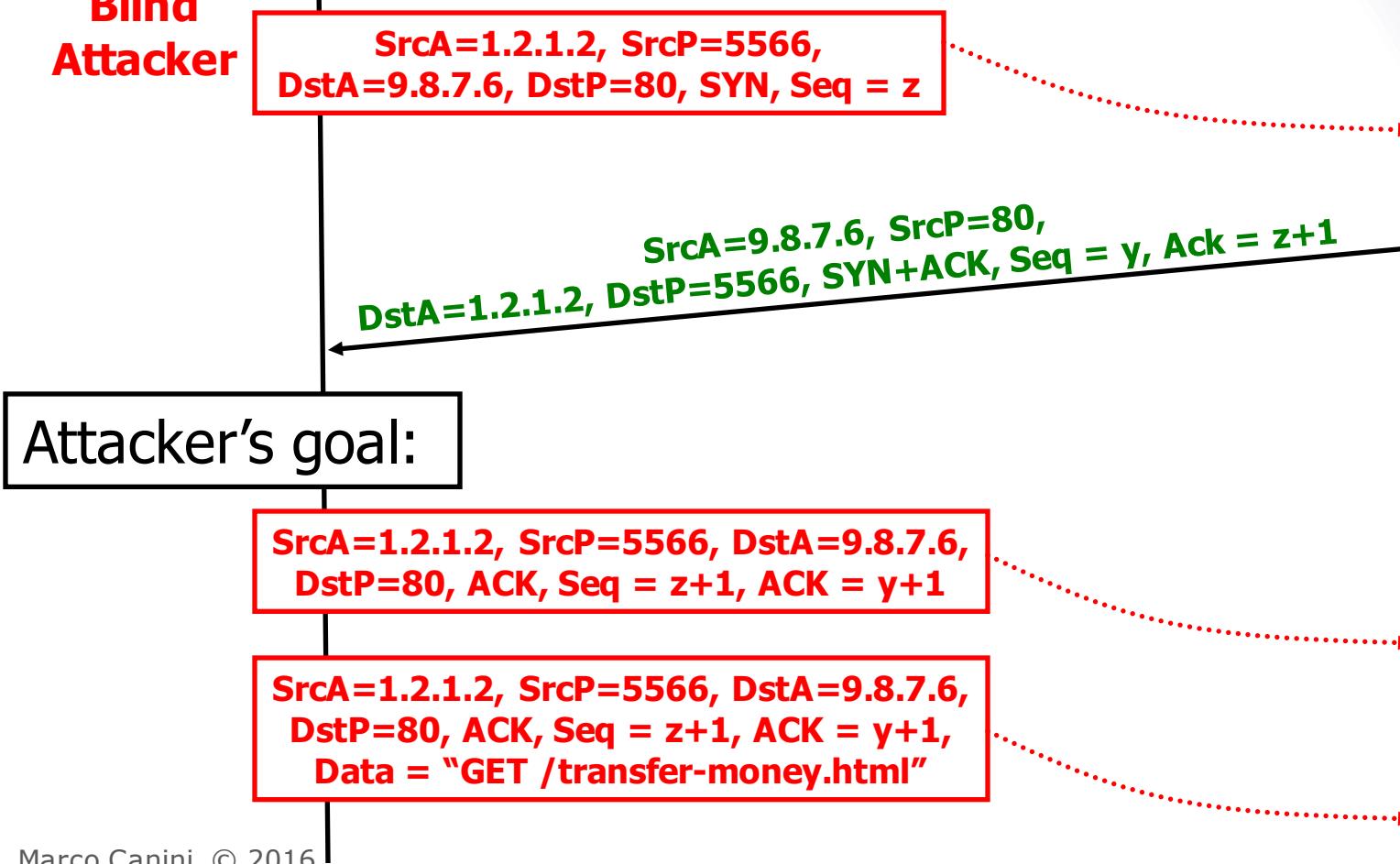
Alleged Client (not actual)

IP address 1.2.1.2, port N/A

**Blind
Attacker**

Server

IP address 9.8.7.6, port 80





Spoofing an Entire TCP Connection

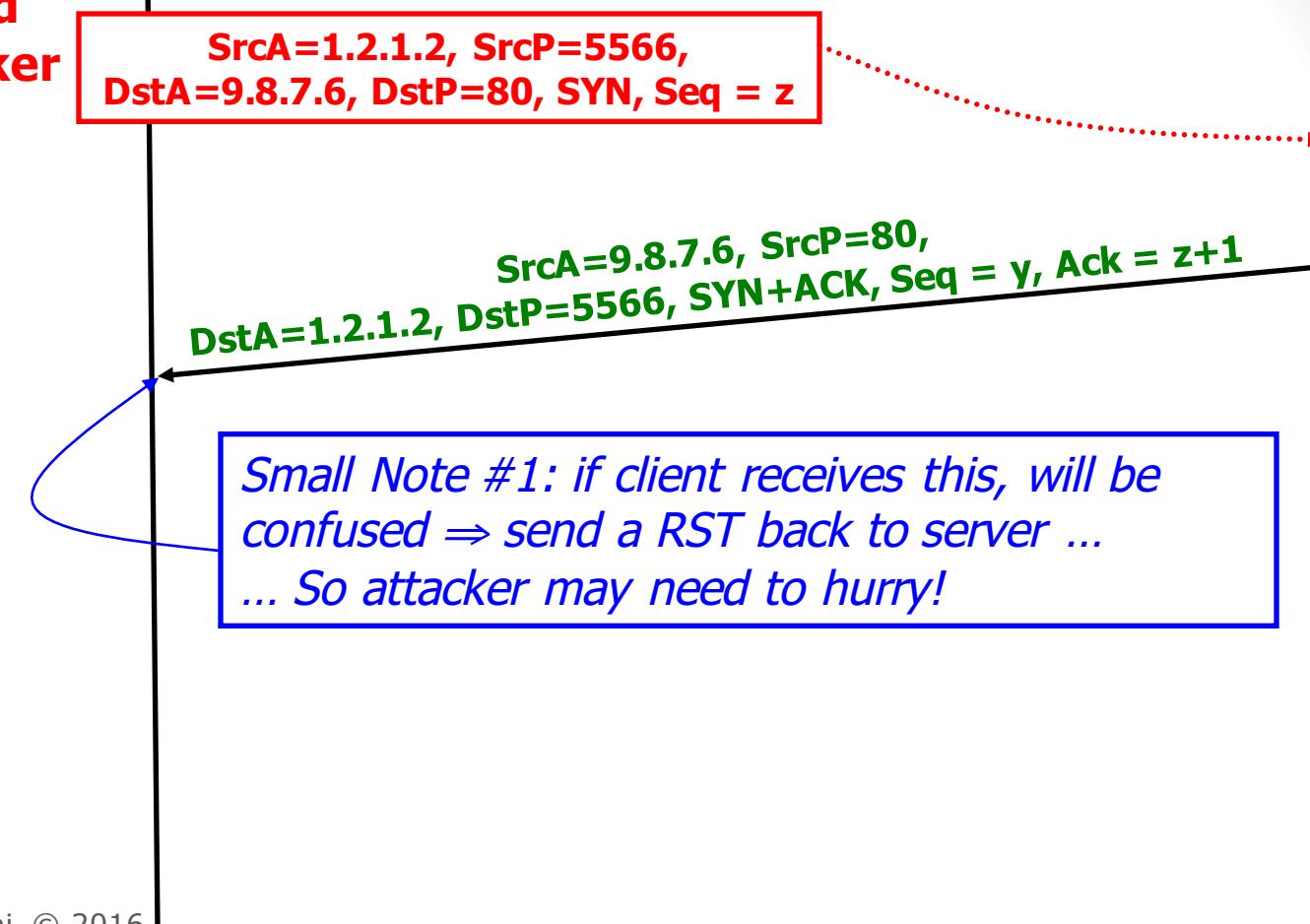
Alleged Client (not actual)

IP address 1.2.1.2, port N/A

**Blind
Attacker**

Server

IP address 9.8.7.6, port 80





Spoofing an Entire TCP Connection

Alleged Client (not actual)

IP address 1.2.1.2, port N/A

**Blind
Attacker**

Server

IP address 9.8.7.6, port 80

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 = z+1

**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

**Blind
Attacker**

Server

IP address 9.8.7.6, port 80

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 = z+1

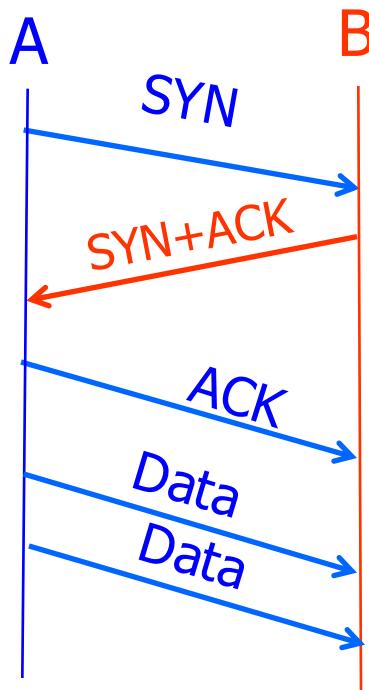
SrcA=1.2.1.2, SrcP=5566, DstA=9.8.7.6,
DstP=80, ACK, Seq = z+1, ACK = y+1

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"

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



Recall: Establishing TCP Connection



How Do We Fix This?

**Use a (Pseudo)-
Random ISN**

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**?

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 (will see later how SSL helps)*

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 (will see later how SSL helps)*
- 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

Threats to Comm. Security Goals

- Attacks can **subvert** each type of goal
 - Confidentiality: **eavesdropping** / theft of information
 - Especially easy when attacker controls a machine close to victim (e.g. WiFi routers)
 - Integrity: **altering** data, **manipulating** execution (e.g., code injection)
 - TCP state easily obtained by eavesdropping
 - Availability: **denial-of-service**
- Attackers can also **combine** different types of attacks towards an overarching goal
 - E.g. use eavesdropping (confidentiality) to construct a spoofing attack (integrity) that tells a server to drop an important connection (denial-of-service)

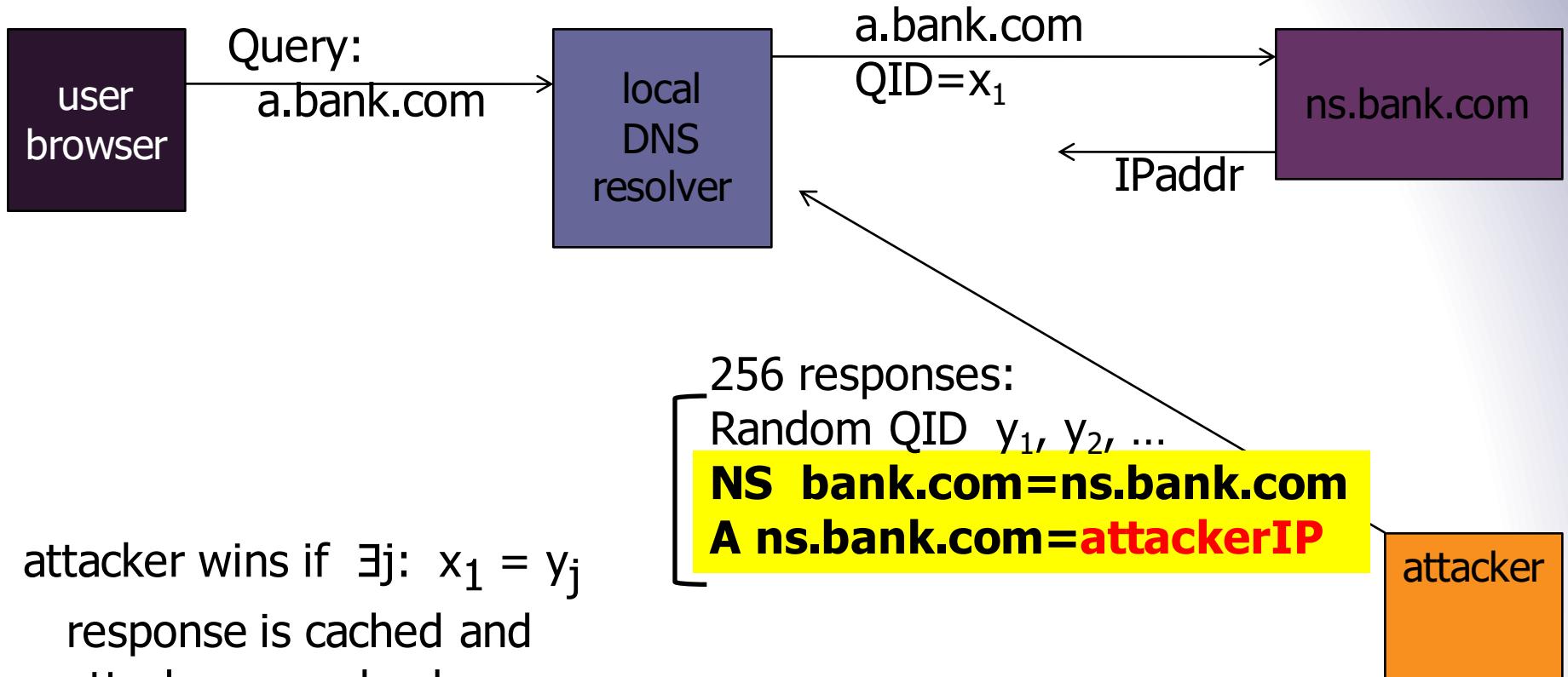
Basic DNS Vulnerabilities

- Users/hosts trust the host-address mapping provided by DNS:
 - Used as basis for many security policies:
Browser same origin policy, URL address bar
- Obvious problems
 - Interception of requests or compromise of DNS servers can result in incorrect or malicious responses
 - e.g.: malicious access point in a cafe
 - Solution – authenticated requests/responses
 - Provided by DNSsec ... but few use DNSsec



DNS cache poisoning (a la Kaminsky'08)

- Victim machine visits attacker's web site, downloads Javascript



Defenses

- Increase Query ID size. How?
 - Changing DNS would be hard, fields are what they are
- Randomize client source port, instead of 1 UDP port
 - Now attack takes several hours
 - Microsoft's updated DNS server is said to preallocate 2,500 UDP ports
 - $2^{16} + 2^{11} = 2^{27} = \text{134 million}$
- Ask every DNS query twice:
 - Attacker has to guess Query ID correctly twice (32 bits)
 - ... but can DNS system handle the additional load?

Extra

↗ www.ioactive.com



Summary

- DNS servers had a core bug, that allows arbitrary cache poisoning
 - The bug works even when the host is behind a firewall
 - There are enough variants of the bug that we needed a stopgap before working on something more complete
- Industry rallied pretty ridiculously to do something about this, with hundreds of millions protected
- DNS clients are at risk, in certain circumstances
- We are entering (or, perhaps, holding back a little longer) a third age of security research, where all networked apps are "fair game"
 - Autoupdate in particular is a mess, broken by design (except for Microsoft)
- SSL is not the panacea it would seem to be
 - In fact, SSL certs are themselves dependent on DNS
- DNS bugs ended up creating something of a "skeleton key" across almost all major websites, despite independent implementations
- Internal networks are not at all safe, both from the effects of Java, and from the fact that internal routing could be influenced by external activity
 - The whole concept of the fully internal network may be broken – there are just so many business relationships – and, between IPsec not triggering and SSL not being cert-validated, these relationships may not be secure
 - We're not even populating CDN's securely!

- Watch Dan Kaminsky's 2008 talk on this serious vulnerability in DNS <https://vimeo.com/17247507>



2	56578021657	78760546412	87546200012	56578021657	78760546412	875
2	89535670000	56701352679	56489854222	89535670000	56701352679	564
3	01444587901	886524.2134	30215021564	01444587901	886524.2134	302
4	89564875564	54654240404	87459823654	89564875564	54654240404	874
5	02654895465	23421404359	86123030213	02654895465	23421404359	851
0	13025165465	78553402213	13311000011	13025165465	78553402213	133
4	76540215497	49758672464	25468952654	76540215497	49758672464	254
3	87654860216	97968652031	78021328503	87654860216	97968652031	786
5	54897564202	25679561203	57920045685	54897564202	25679561203	576
3	15465465460	26456530979	48314904153	15465465460	26456530979	483
5	21654					1246 185
5	40216					2123 515
1	56102					4545 231
1	62165					5425 625
2	13245450154	34659782135	35656497652	13245450154	34659782135	356
5	84987984301	54023100002	31200124556	84987984301	54023100002	312
0	24568765435	13656462857	87976423120	24568765435	13656462857	875
1	01235435435	55645622256	31655976421	01235435435	55645622256	311
2	43021648576	79866566433	05234605242	43021648576	79866566433	052
1	53441100000	59823101346	58257561221	53441100000	59823101346	592
7	000000001243	56457242104	56024565237	000000001243	56457242104	560
4	53727672034	23168976543	85421245454	53727672034	23168976543	854
4	25375763520	24212124567	45456402124	25375763520	24212124567	454
2	43597572672	54212054276	24575454012	43597572672	54212054976	245
0	40133727967	85323051564	42245454440	40133727967	85323051564	422
3	97801322479	65246791530	55546520303	97801322479	65246791630	558

Denial of Service

Denial of Service (DoS)

- An attempt to make a machine or network resource unavailable to its intended users, either temporarily or indefinitely
- Many methods of attack exist, classifiable as:
 - Consumption of computation resources (e.g. CPU, memory, bandwidth)
 - Disruption of configuration (e.g. routing information)
 - Disruption of state information (e.g. sending TCP RST packets)
 - Disruption of physical components
 - Obstructing communication media between the victim and its users
- DoS attacks typically involve forged IP addresses

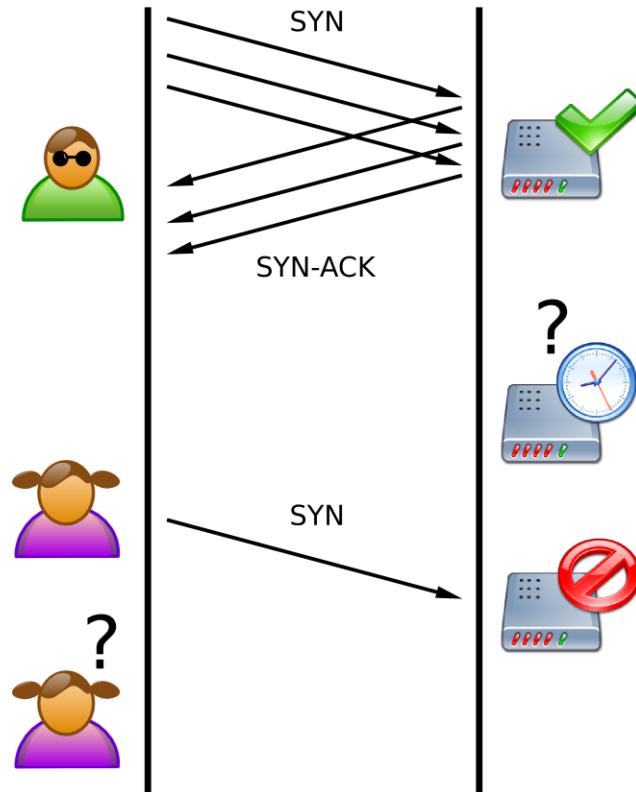
Historical example

Ping of death (PoD)

- One of the earliest denial of service attack
 - Up to 1997, affected Unix, Linux, Mac, Windows, printers, and routers
- Attack based on sending a malformed ICMP packet
 - Send a ping larger than the maximum IPv4 packet size (65,535 bytes) could crash a target
 - Exploit fragmentation (problem has nothing to do with ICMP)
 - When packet is reassembled, a buffer overflow can occur, which often causes a system crash

SYN Flooding

- Send a large number TCP/SYN packets
 - Recall 3-way handshake: SYN ; SYN + ACK ; ACK



Source: Wikipedia

SYN Flooding

- On receiving the SYN packet, the server allocates necessary memory for the connection and enters it in a **queue** of half open connections
- Once the queue overflows, the server **cannot** accept any new connection
- The attacker can **forge** the source address of his SYN packets to remain anonymous
- Recent versions of operating systems (Windows, Unix, Linux) are **protected** against such attacks

Protections Against SYN Flooding

- Increase the size of the queue
- Reduce timeout while server is waiting for an ACK
- Drop the oldest SYN in the queue
- Filtering e.g. on IP addresses
- ...

Protections Against SYN Flooding

SYN Cache

- Allocate minimal state on the server, reply with SYN + ACK and complete the connection creation once the ACK is received
- Host compute a hash based on some secret bits, IP addresses and transport ports that determines the location in a global hash table where the incomplete connection information is stored
- Still must be prepared to overflow

Protections Against SYN Flooding

SYN cookies

- Once the connection queue is almost filled up, the server uses SYN cookies
- Upon reception of a SYN:
 - The server sends a SYN + ACK containing a SYN cookie
 - The server erases the SYN entry
- Upon reception of an ACK:
 - The server checks whether it contains a valid cookie
 - If so this highly likely means that the client has already sent a SYN and so it is an honest client



SYN Cookie Content

SYN cookies are specific **Initial Sequence Numbers**:

- t is a counter incremented every 64 s modulo 32
- m is the Maximum Segment Size encoded on 3 bits
- s is the result of a cryptographic hash function computed on t and the IP address and port number of the server and client

ISN =

$t \text{ mod } 32$	m	s
5 bits	3 bits	24 bits
32 bits		

SYN Cookie Check

Upon reception of an ACK, the server carries out the following operations:

- Check that the received value t is valid with respect to the current time
 - Otherwise, this means the connection is expired
- Recompute s to check its validity
- Decode the value m , which allows the server to reconstruct the SYN queue entry

Any questions?



Stay tuned



Next time you will learn about

Network defense tools