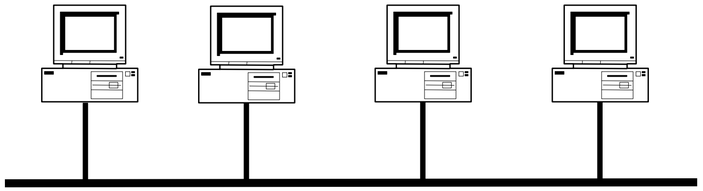
Attacks

Link-layer threats

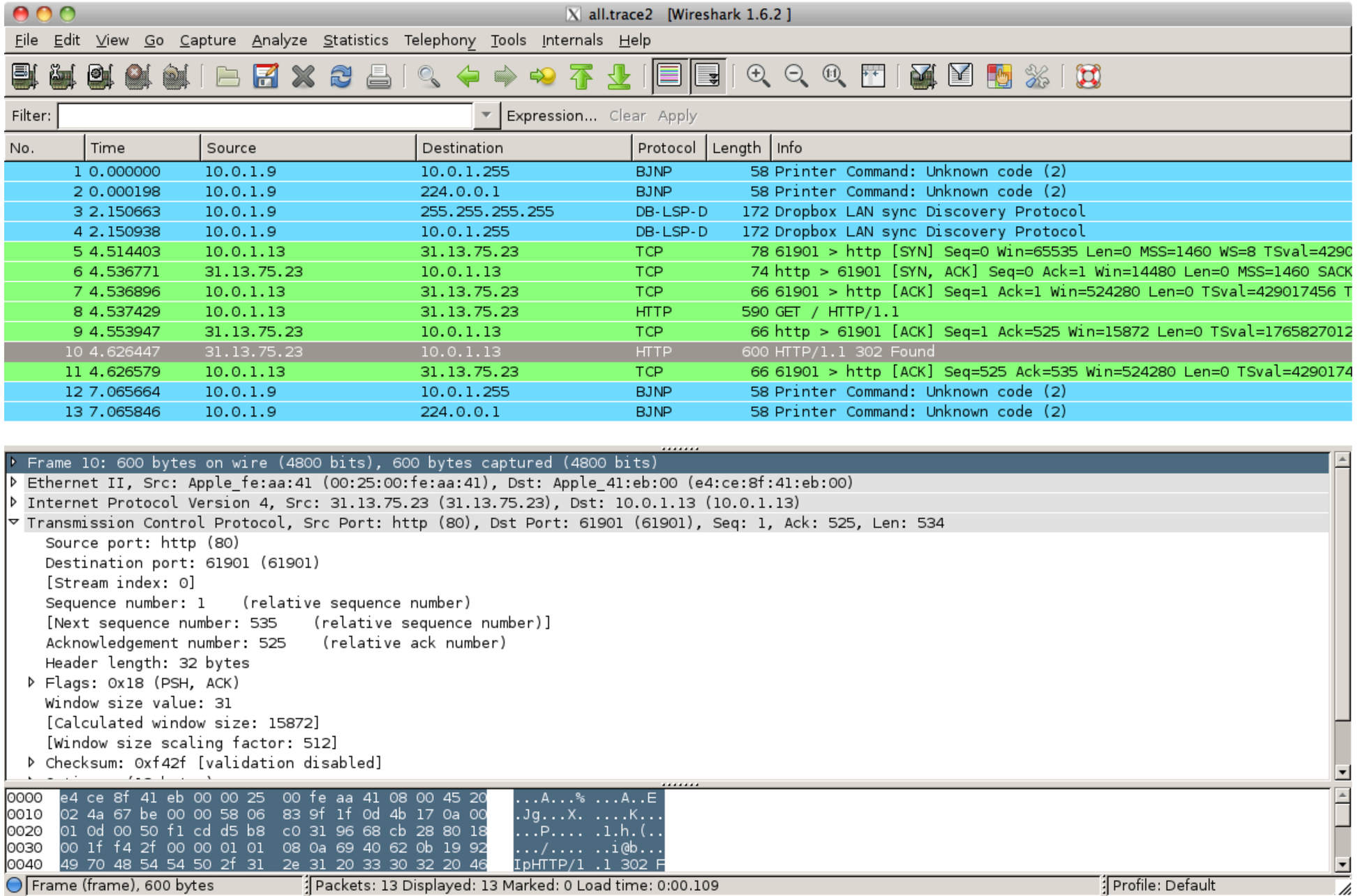


- Confidentiality: eavesdropping (aka sniffing)
- Integrity: injection of spoofed packets
- Availability: delete legit packets (e.g., jamming)

Eavesdropping

- For subnets using broadcast technologies (e.g., WiFi, some types of Ethernet), attacker can eavesdrop
  - Each attached system's NIC (= Network Interface Card) can capture any communication on the subnet
  - Tools: tcpdump / windump (low-level text-based printout), wireshark (GUI)

Wireshark


all.trace2 [Wireshark 1.6.2]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: Expression... Clear Apply
No. Time Source Destination Protocol Length Info
1 0.000000 10.0.1.9 10.0.1.255 BJNP 58 Printer Command: Unknown code (2)
2 0.000198 10.0.1.9 224.0.0.1 BJNP 58 Printer Command: Unknown code (2)
3 2.150663 10.0.1.9 255.255.255.255 DB-LSP-D 172 Dropbox LAN sync Discovery Protocol
4 2.150938 10.0.1.9 10.0.1.255 DB-LSP-D 172 Dropbox LAN sync Discovery Protocol
5 4.514403 10.0.1.13 31.13.75.23 TCP 78 61901 > http [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=8 TSval=4290
6 4.536771 31.13.75.23 10.0.1.13 TCP 74 http > 61901 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK
7 4.536896 10.0.1.13 31.13.75.23 TCP 66 61901 > http [ACK] Seq=1 Ack=1 Win=524280 Len=0 TSval=429017456 T
8 4.537429 10.0.1.13 31.13.75.23 HTTP 590 GET / HTTP/1.1
9 4.553947 31.13.75.23 10.0.1.13 TCP 66 http > 61901 [ACK] Seq=1 Ack=525 Win=15872 Len=0 TSval=1765827012
10 4.626447 31.13.75.23 10.0.1.13 HTTP 600 HTTP/1.1 302 Found
11 4.626579 10.0.1.13 31.13.75.23 TCP 66 61901 > http [ACK] Seq=525 Ack=535 Win=524280 Len=0 TSval=4290174
12 7.065664 10.0.1.9 10.0.1.255 BJNP 58 Printer Command: Unknown code (2)
13 7.065846 10.0.1.9 224.0.0.1 BJNP 58 Printer Command: Unknown code (2)
Frame 10: 600 bytes on wire (4800 bits), 600 bytes captured (4800 bits)
Ethernet II, Src: Apple_fe: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
Source port: http (80)
Destination port: 61901 (61901)
[Stream index: 0]
Sequence number: 1 (relative sequence number)
[Next sequence number: 535 (relative sequence number)]
Acknowledgement number: 525 (relative ack number)
Header length: 32 bytes
Flags: 0x18 (PSH, ACK)
Window size value: 31
[Calculated window size: 15872]
[Window size scaling factor: 512]
Checksum: 0xf42f [validation disabled]
0000 e4 ce 8f 41 eb 00 00 25 00 fe aa 41 08 00 45 20 ...A...% ...A..E
0010 02 4a 67 be 00 00 58 06 83 9f 1f 0d 4b 17 0a 00 .Jg...X. ....K...
0020 01 0d 00 50 f1 cd d5 b8 c0 31 96 68 cb 28 80 18 ...P.... .1.h.(..
0030 00 1f f4 2f 00 00 01 01 08 0a 69 40 62 0b 19 92 .../.... ..i@b...
0040 49 70 48 54 54 50 2f 31 2e 31 20 33 30 32 20 46 IpHTTP/1 .1 302 F
Frame (frame), 600 bytes
Packets: 13 Displayed: 13 Marked: 0 Load time: 0:00.109
Profile: Default

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".

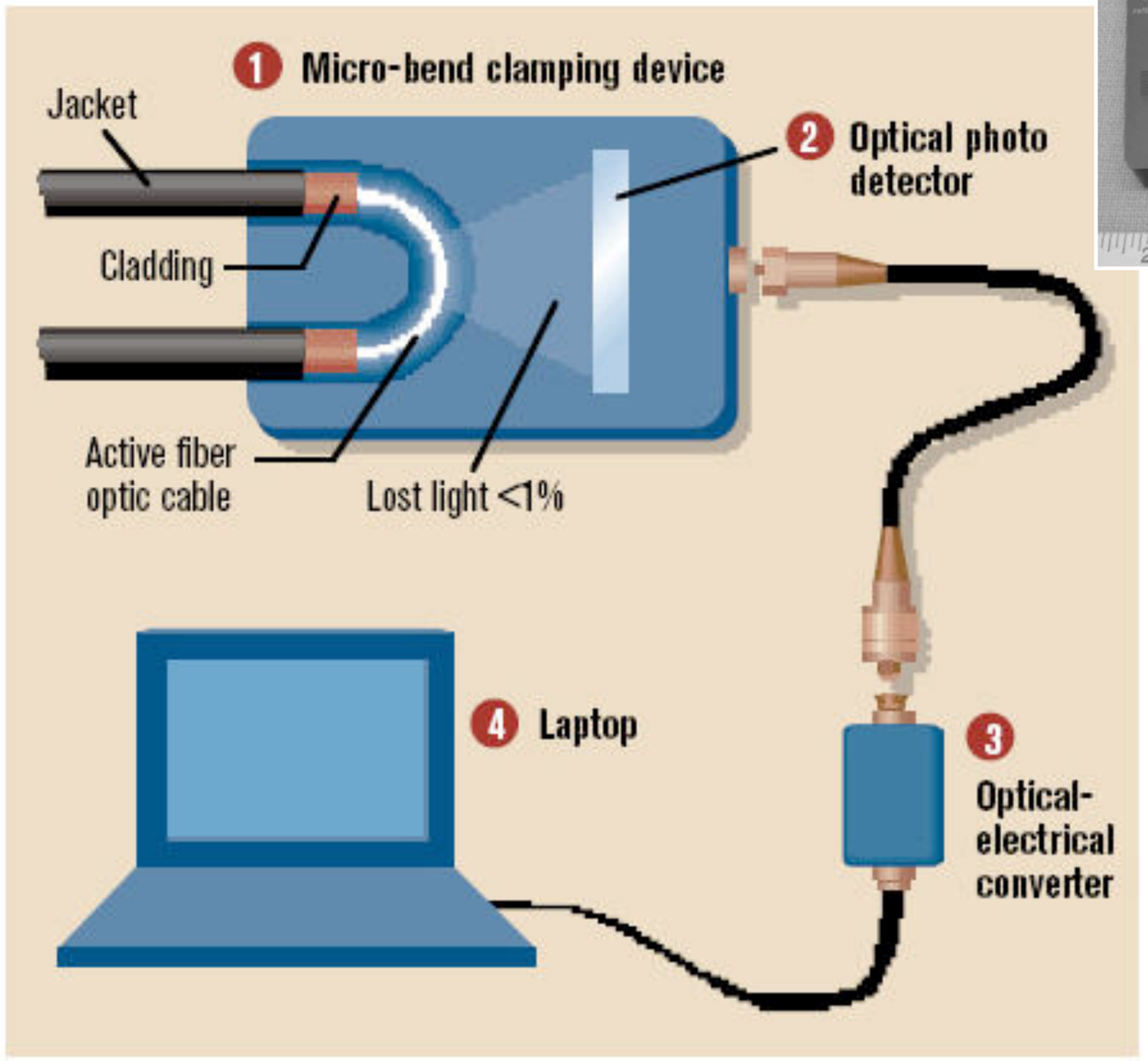
The Regulus guided missile submarine, USS Halibut (SSN 587) which carried out Operation Ivy Bells.

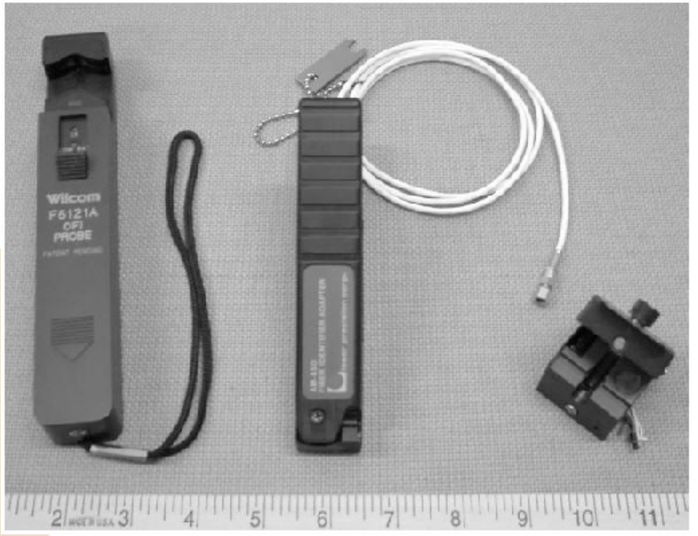
In an effort to alter the balance of Cold War, these men scoured the ocean floor for a five-inch diameter cable carry secret Soviet communications between military bases.

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.


1 Micro-bend clamping device
Jacket
2 Optical photo detector
Cladding
Active fiber optic cable
Lost light <1%
4 Laptop
3 Optical-electrical converter


Wilcom
F6121A
CFI
PROBE

Link-Layer Spoofing

- Attacker can inject spoofed packets, and lie about the source address

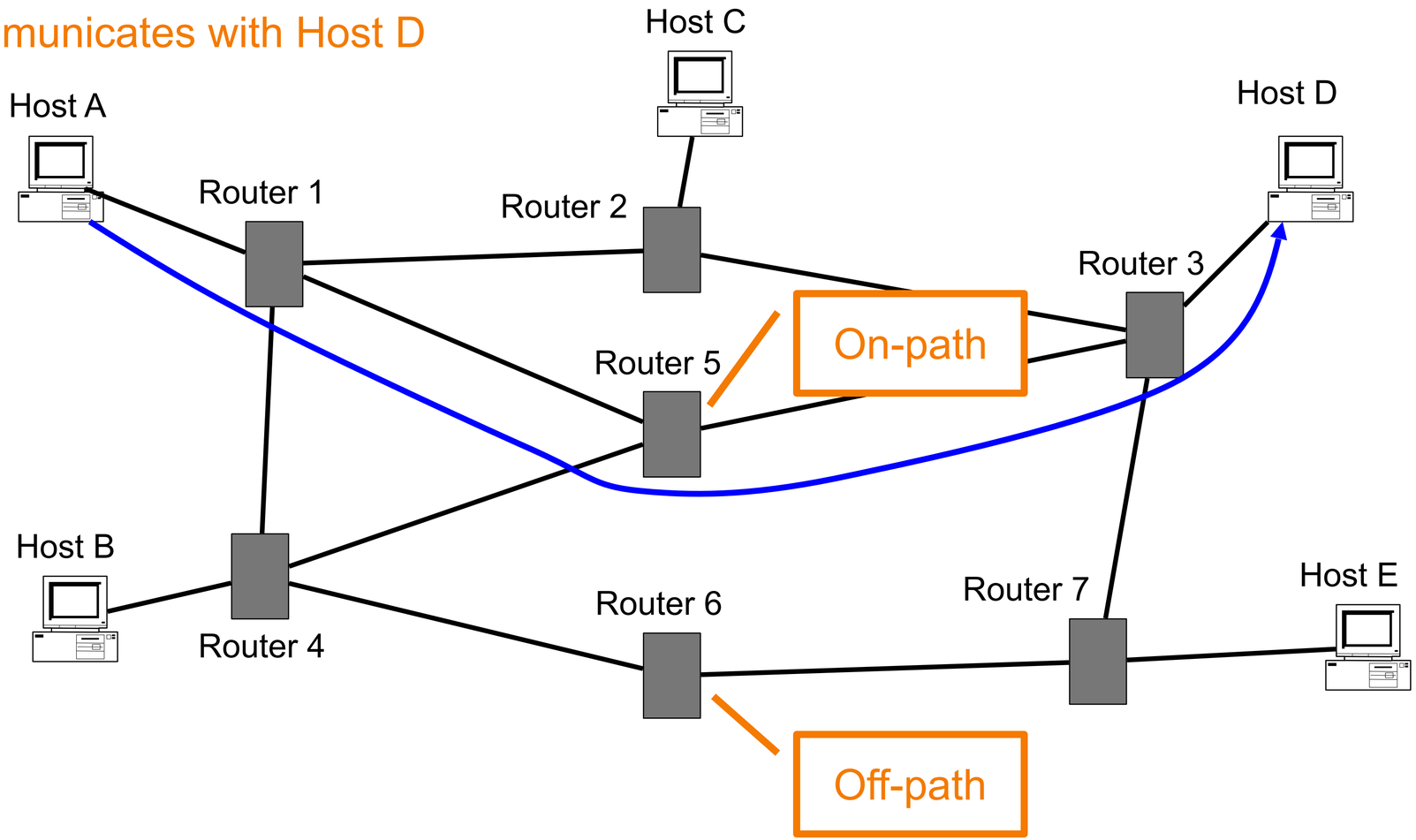
- 

Spoofing

- With physical access to a local network, attacker can create any packet they like
  - Spoofing = lie about source address
- 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

On-path vs Off-path Spoofing

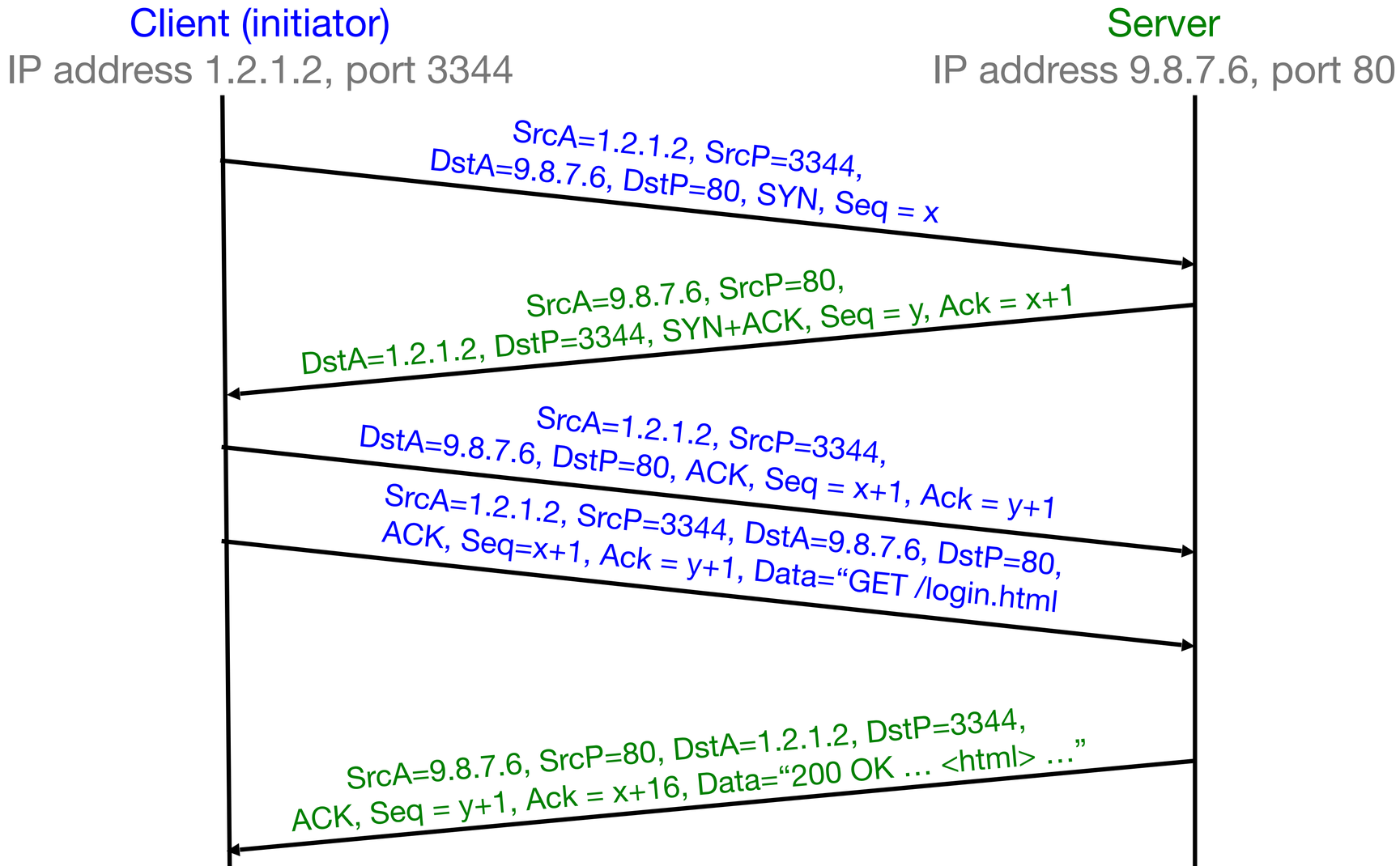
Host A communicates with Host D



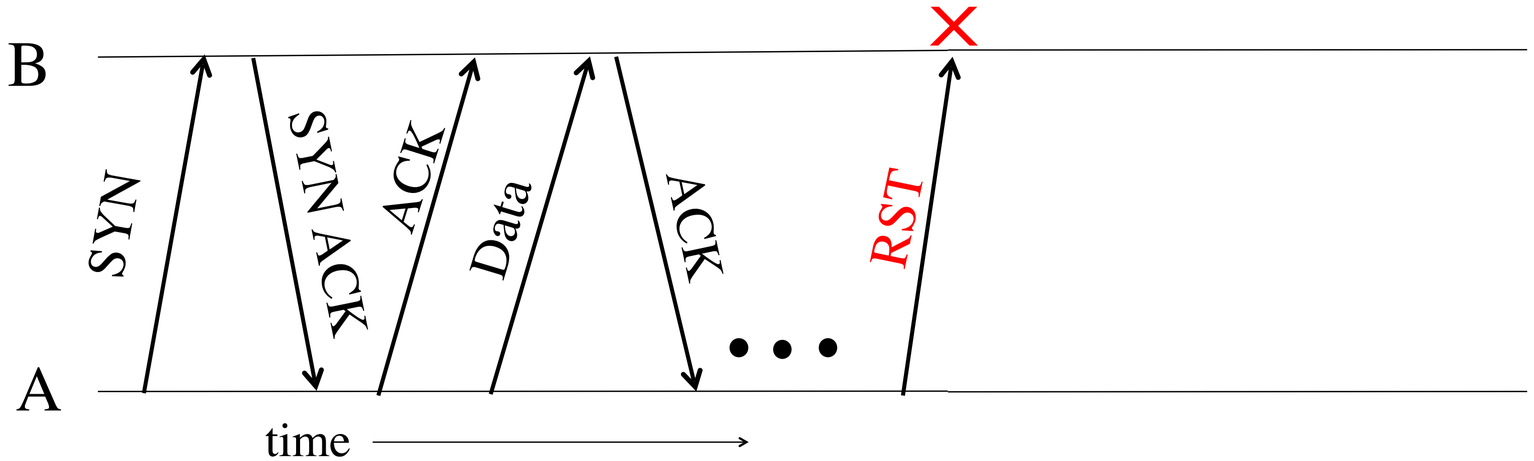
Spoofing on the Internet

- 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 "w/ reasonable chance of success"

TCP Conn. Setup & Data Exchange



Abrupt Termination



- If A sends a TCP packet with RST flag to B and sequence number fits, connection is terminated
  - Unilateral, and takes effect immediately

TCP Threat: Disruption aka RST injection

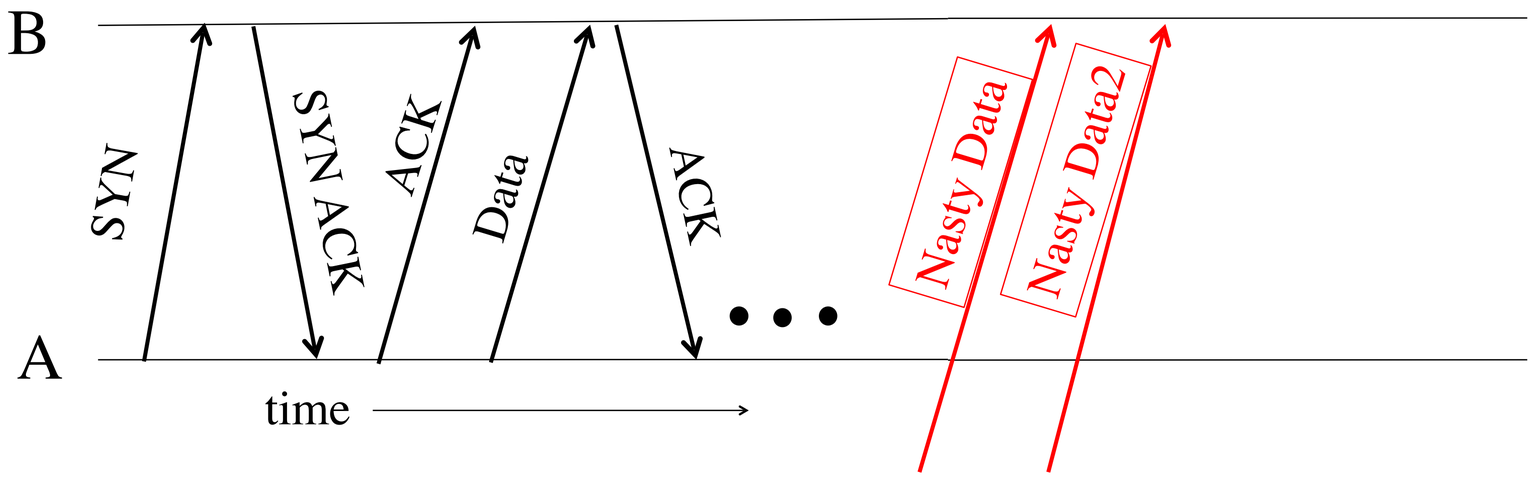
- The attacker can inject RST packets and block connection
  - TCP clients must respect RST packets and stop all communication

- Who uses this?
  - China: The Great Firewall does this to TCP requests
  - A long time ago: Comcast, to block BitTorrent uploads
  - Some intrusion detection systems: To hopefully mitigate an attack in progress

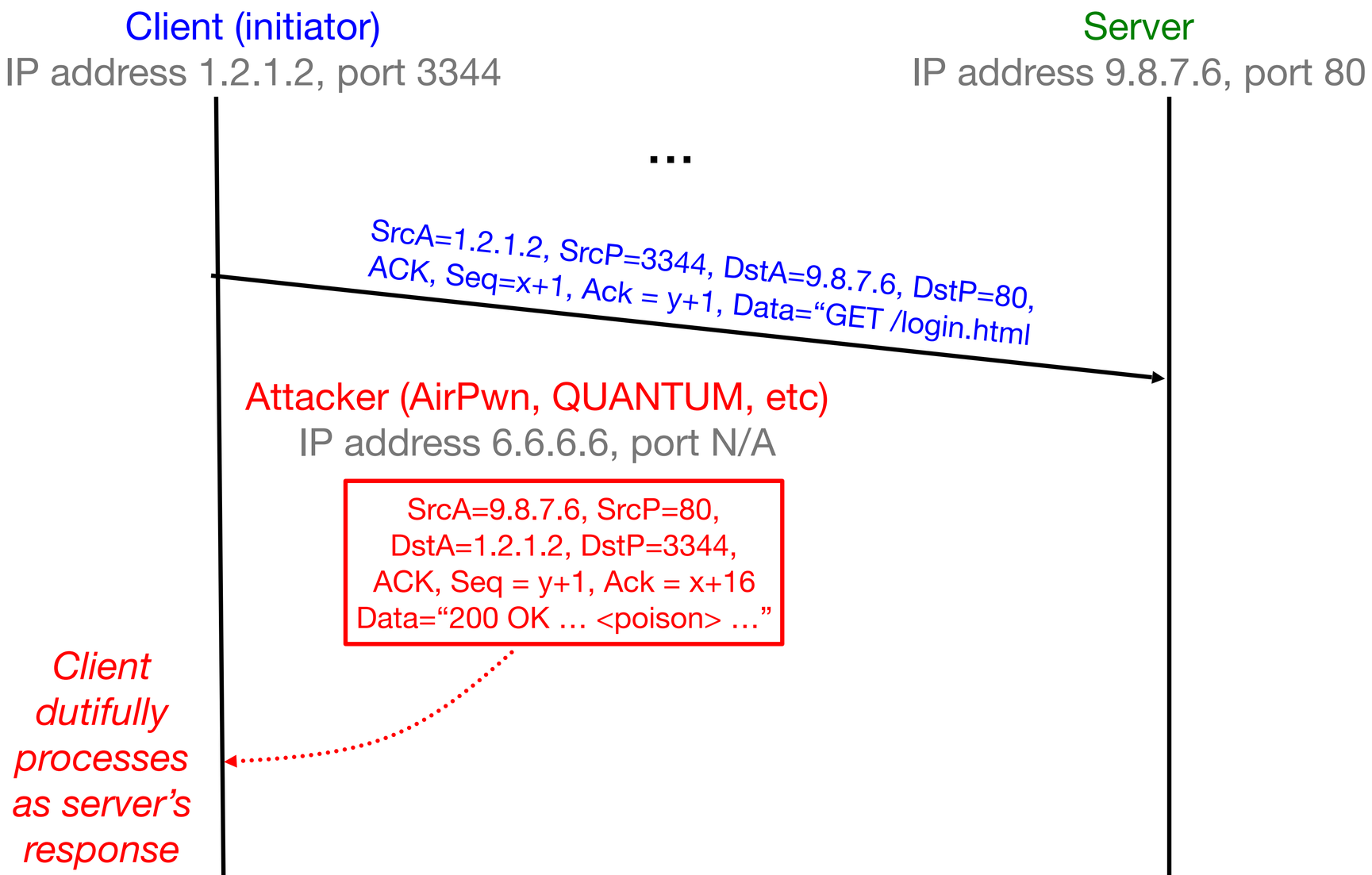
Discuss with a partner: Who can do RST injection? (a) off-path attacker, (b) on-path attacker, (c) man-in-the-middle

TCP Threat: Data Injection

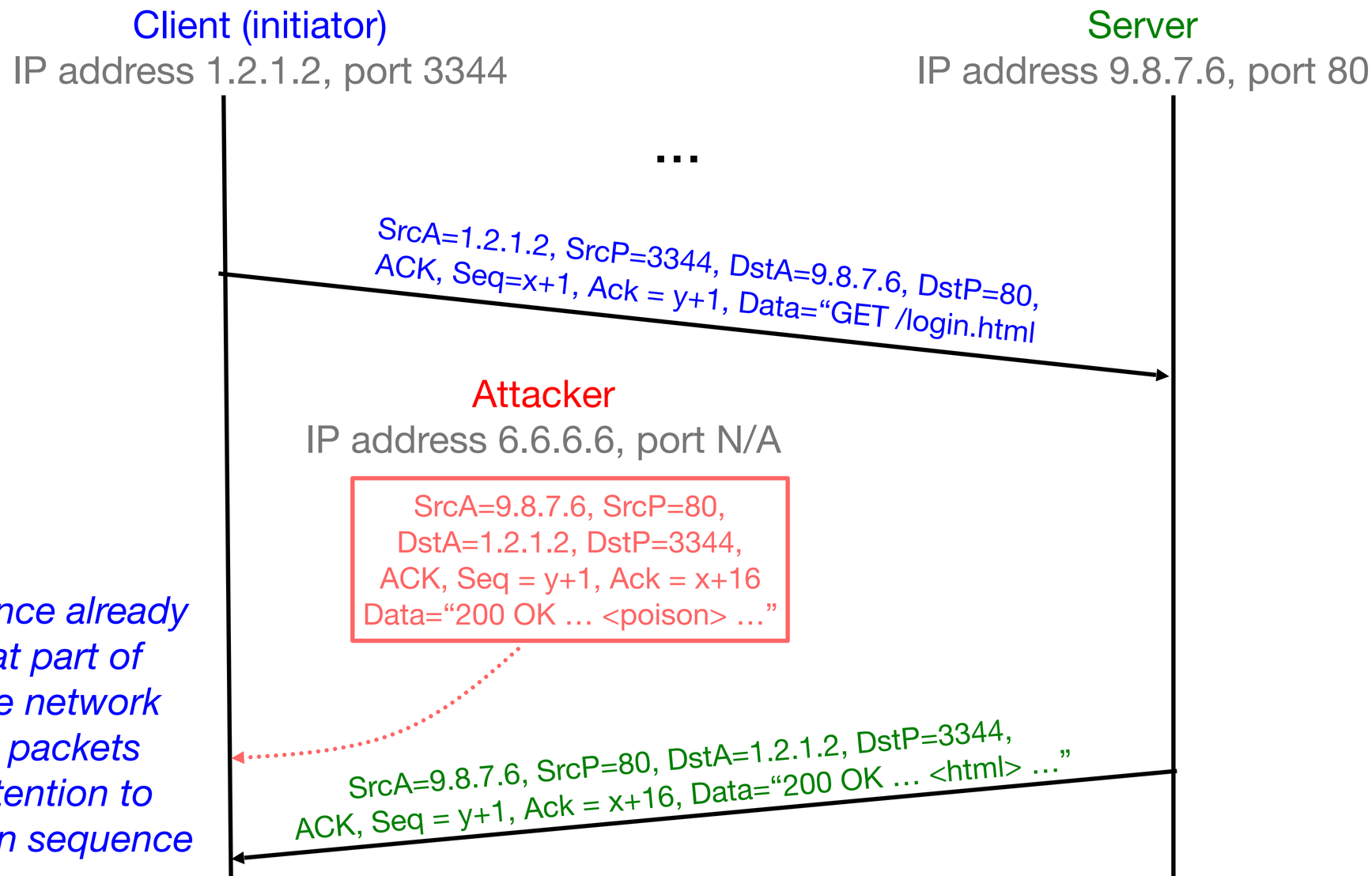
- If attacker knows ports & sequence numbers (e.g., on-path attacker), attacker can inject data into any TCP connection
  - Receiver B is *none the wiser!*
- Termed TCP connection hijacking (or "*session hijacking*")
  - A general means to take over an already-established connection!
- We are toast if an attacker can see our TCP traffic!
  - Because then they immediately know the port & sequence numbers



TCP Data Injection



TCP Data Injection



Client ignores since already processed that part of bytestream: the network can duplicate packets so only pay attention to the first version in sequence

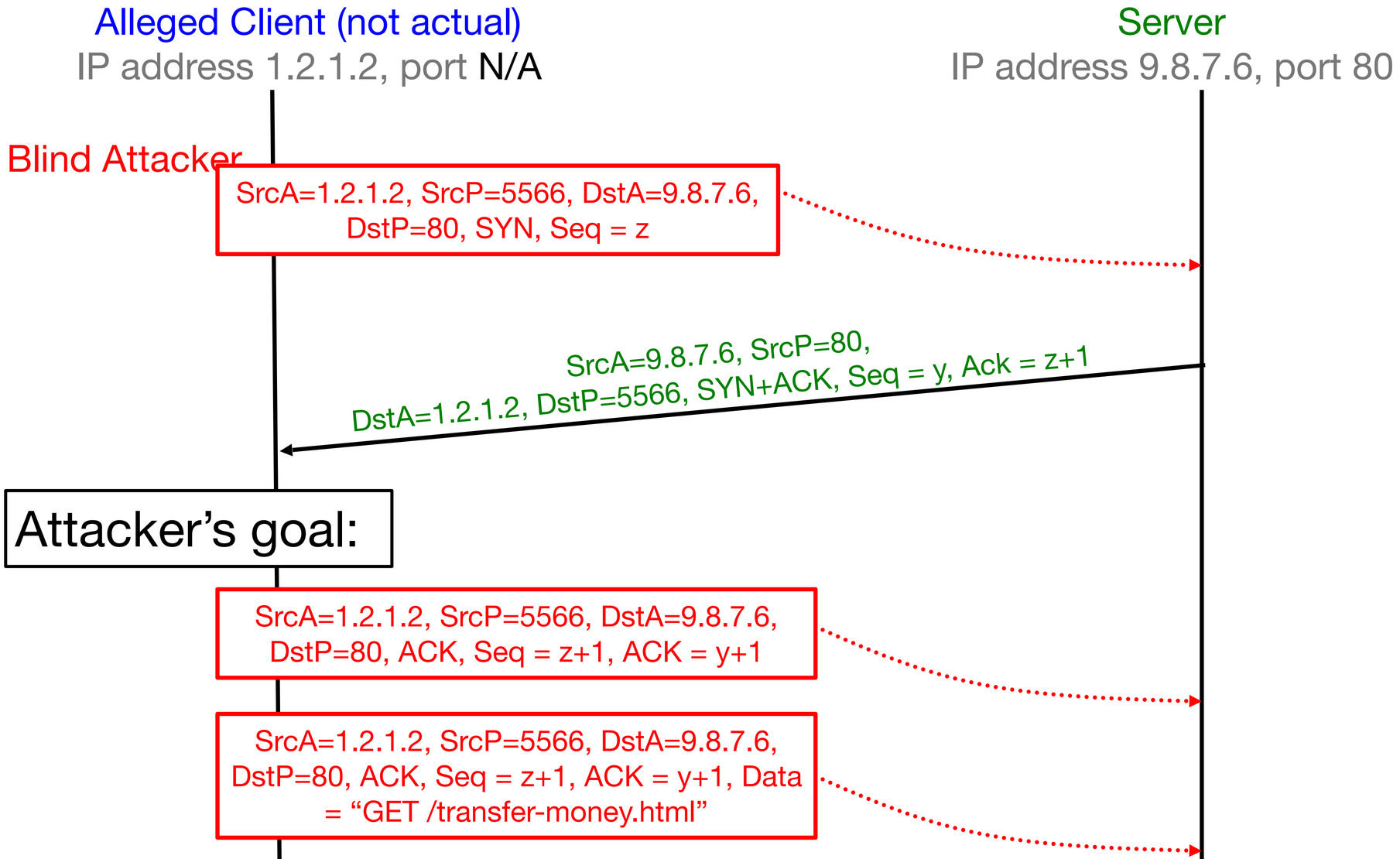
TCP Threat: Blind Hijacking

- Is it possible for an off-path 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

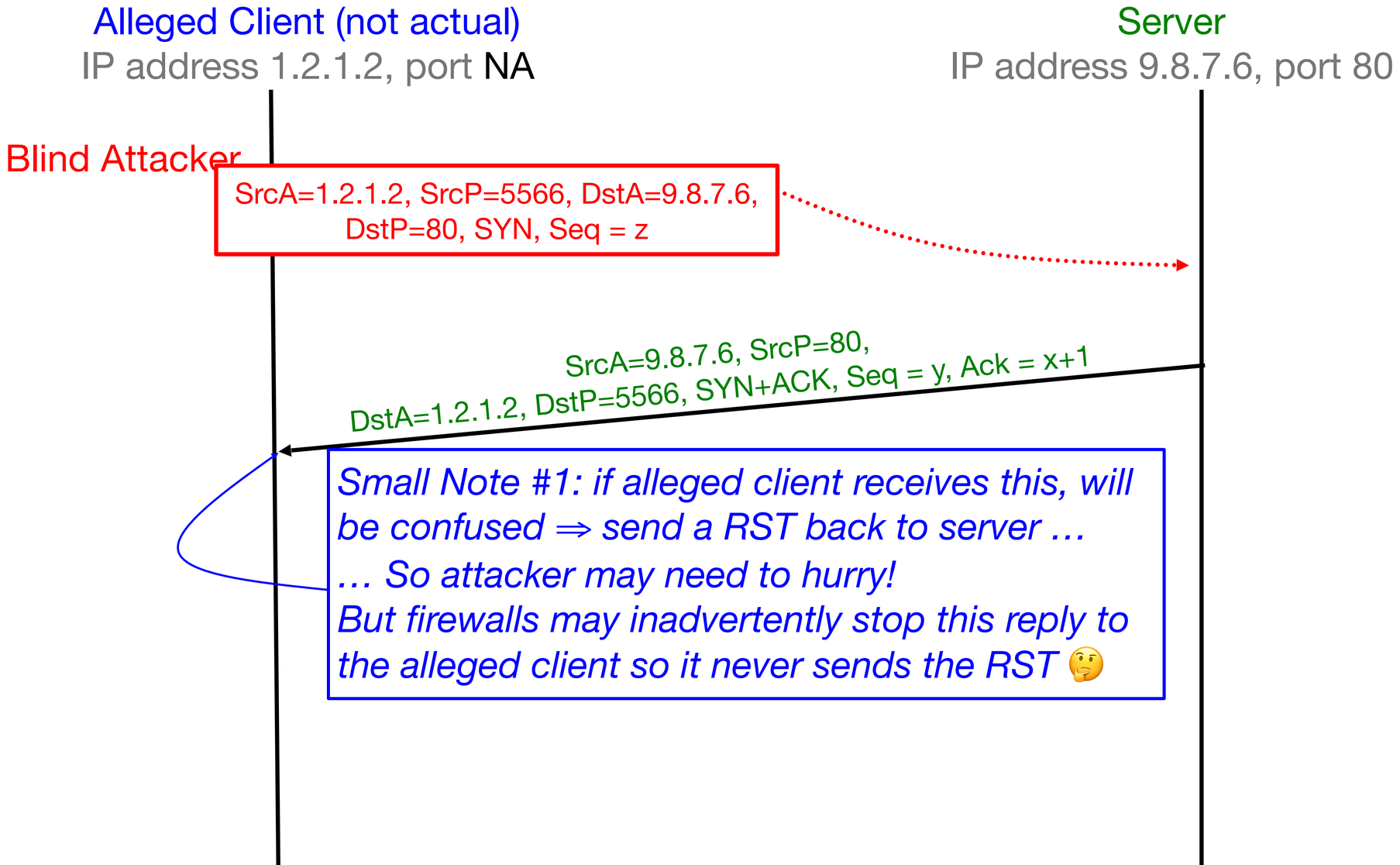
TCP Threat: Blind Spoofing

- Is it possible for an off-path attacker to create a fake TCP connection, even if they can't see responses?
- Yes if somehow they can infer or guess the TCP initial sequence numbers
- Why would an attacker want to do this?
  - 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

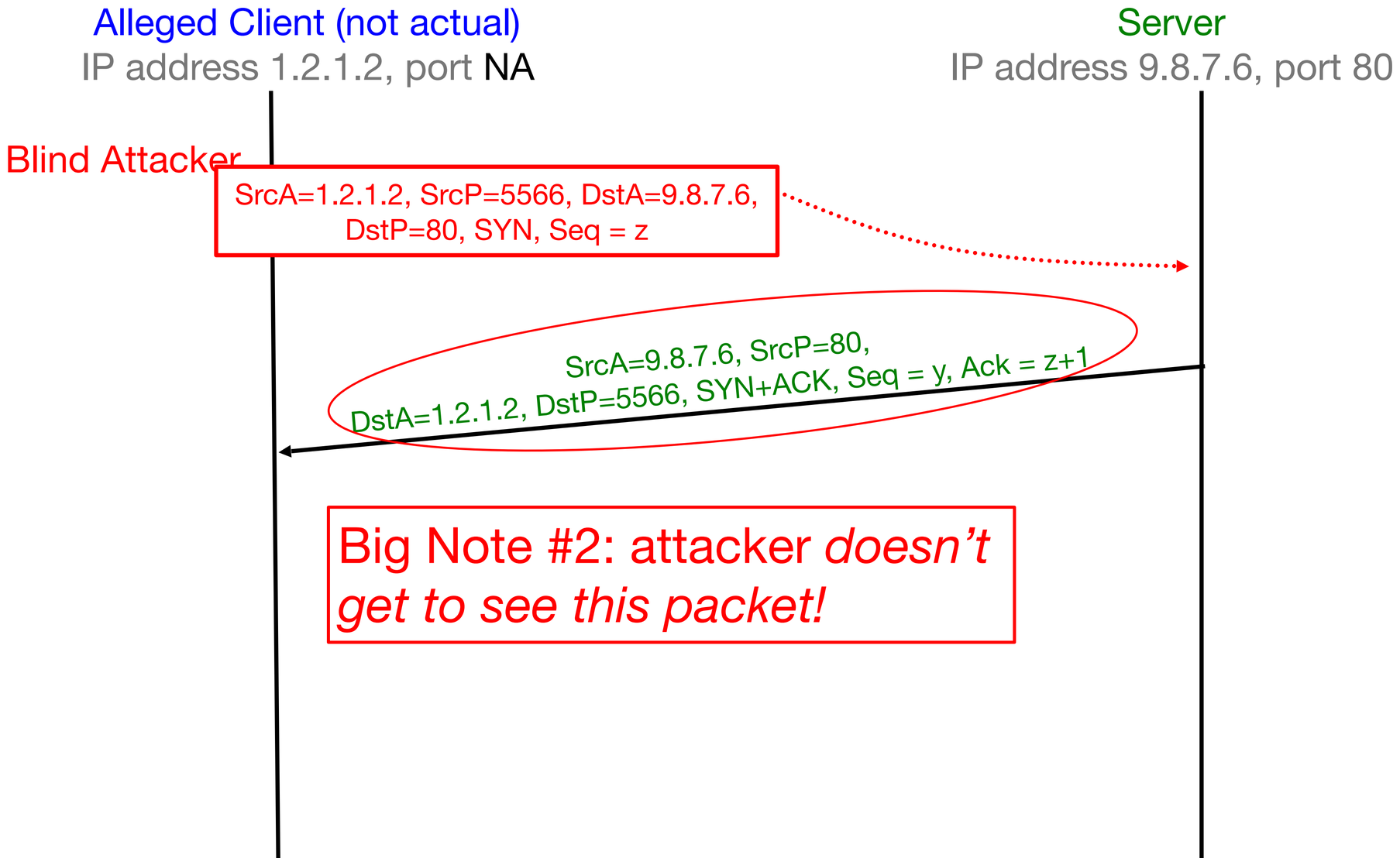
Blind Spoofing on TCP Handshake



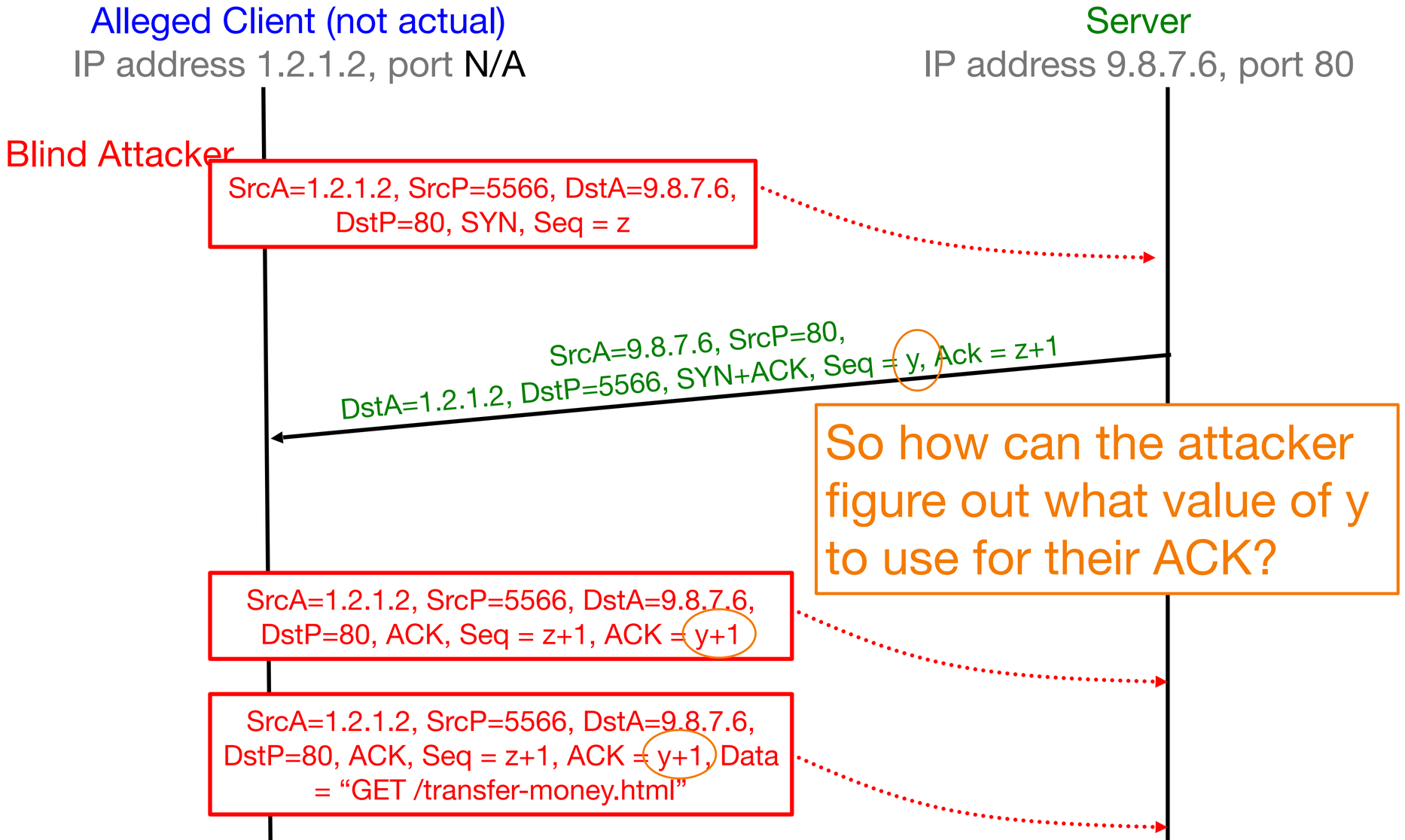
Blind Spoofing on TCP Handshake



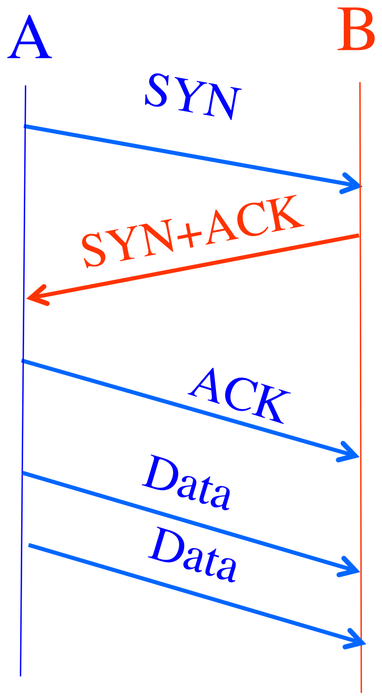
Blind Spoofing on TCP Handshake



Blind Spoofing on TCP Handshake



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
- Blind spoofing no longer a threat
  - Due to randomization of TCP initial sequence numbers

Ghost of blind spoofing...

- CVE-2016-5696
  - "Off-Path TCP Exploits: Global Rate Limit Considered Dangerous" Usenix Security 2016 https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/cao
- Key idea:
  - RFC 5961 added some global rate limits that acted as an ***information leak***:
    - Could determine if two hosts were communicating on a given port
    - Could determine if your guess at the sequence number is “in window”
  - Once you get the sequence #s, you can then inject arbitrary content into the TCP stream
- Fixed today

Host Names vs. IP addresses

- Host names
  - Examples: `www.cnn.com` and `bbc.co.uk`
  - Mnemonic name appreciated by humans
  - Variable length, full alphabet of characters
  - Provide little (if any) information about location
- IP addresses
  - Examples: `64.236.16.20` and `212.58.224.131`
  - Numerical address appreciated by routers
  - Fixed length, binary number
  - Hierarchical, related to host location

Networking Roadmap

Layer	Protocols
7. Application	Web security
4.5. Secure transport	TLS
4. Transport	TCP, UDP
3. Internet	IP
2. Link	
1. Physical	

You are here

We're done with these

Extra protocols

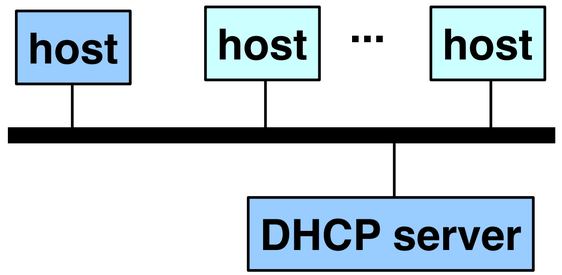
	Protocols
Connect for the first time	DHCP
Convert hostname to IP address	DNS, DNSSEC

DNS Service

- Runs Domain Name Servers
- Translates domain names google.com to IP addresses
- When user browser wants to contact google.com, it first contacts a DNS to find out the IP address for google.com and then sends a packet to that IP address
- More soon..

LAN Bootstrapping: DHCP

- New host doesn't have an IP address yet
  - So, host doesn't know what source address to use
- Host doesn't know *who to ask* for an IP address
  - So, host doesn't know what destination address to use
- Solution: shout to "**discovery**" server that can help
  - Broadcast a server-discovery message (layer 2)
  - Server(s) sends a reply offering an address

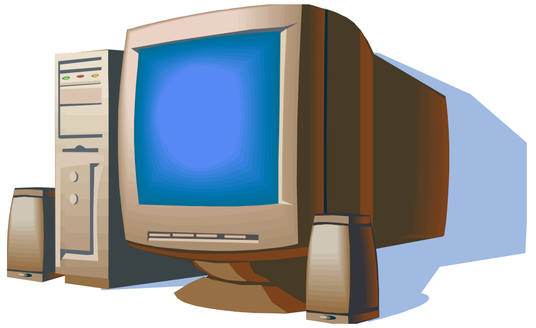


Dynamic Host Configuration Protocol

new client

DHCP discover (broadcast)

DHCP offer



DHCP server

"**offer**" message includes IP address, DNS server, "gateway router", and how long client can have these ("lease" time)

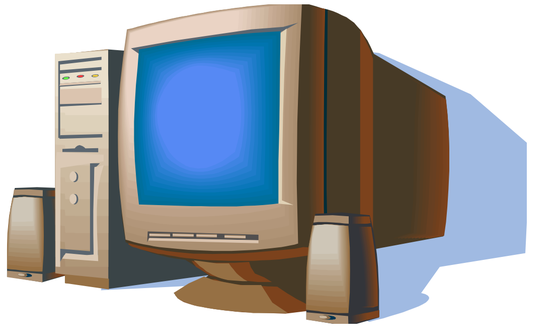
DNS server = system used by client to map hostnames like `gmail.com` to IP addresses like `74.125.224.149`

Gateway router = router that client uses as the first hop for all of its Internet traffic to remote hosts

Dynamic Host Configuration Protocol

new client

DHCP discover (broadcast)



DHCP server

DHCP offer

DHCP request (broadcast)

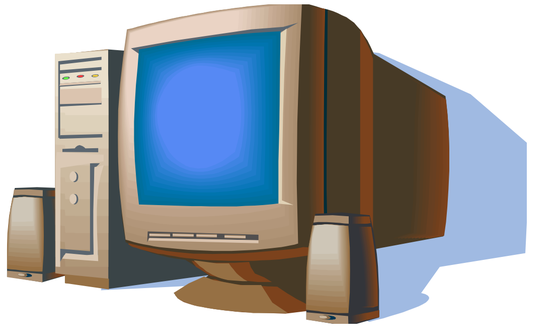
DHCP ACK

"**offer**" message includes IP address, DNS server, "gateway router", and how long client can have these ("lease" time)

Dynamic Host Configuration Protocol

new client

DHCP discover (broadcast)



DHCP server

DHCP offer

"**offer**" message includes IP address, DNS server, "gateway router", and how long client can have these ("lease" time)

DHCP request (broadcast)

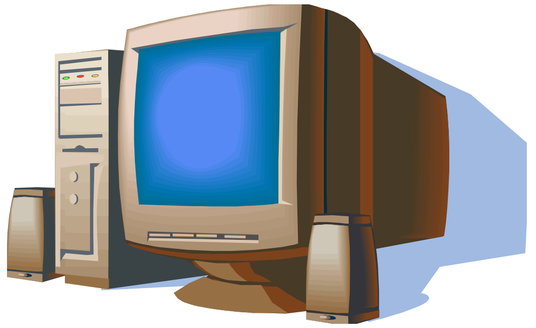
Threats?

DHCP ACK

Dynamic Host Configuration Protocol

new client

DHCP discover (broadcast)



DHCP server

DHCP offer

DHCP request (broadcast)

DHCP ACK

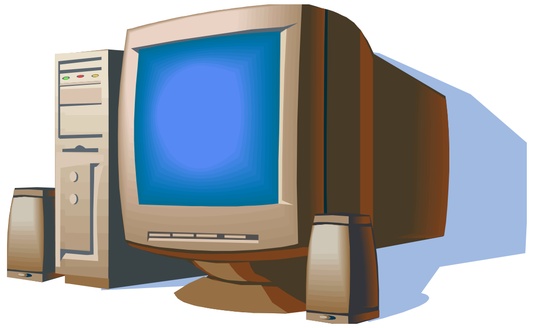
Local attacker on same subnet can **hear** new host's DHCP request

"**offer**" message includes IP address, DNS server, "gateway router", and how long client can have these ("lease" time)

Dynamic Host Configuration Protocol

new client

DHCP discover (broadcast)



DHCP server

DHCP offer

DHCP request (broadcast)

DHCP ACK

"**offer**" message includes IP address, DNS server, "gateway router", and how long client can have these ("lease" time)

Attacker can **race** the actual server; if attacker wins, replaces DNS server and/or gateway router

DHCP Threats

- Substitute a fake DNS server
  - Redirect any of a host's lookups to a machine of attacker's choice
- Substitute a fake gateway router
  - Intercept all of a host's off-subnet traffic (even if not preceded by a DNS lookup)
  - Relay contents back and forth between host and remote server and modify however attacker chooses
- An invisible Man In The Middle (MITM)
  - Victim host has no way of knowing it's happening
- How can we fix this?

Hard, because we lack a ***trust anchor***

Takeaways

- Broadcast protocols inherently at risk of local attacker spoofing
- When initializing, systems are particularly vulnerable because they can lack a trusted foundation to build upon
  - Tension between ***wiring in trust*** vs. ***flexibility and convenience***
- MITM attacks insidious because no indicators they're occurring