

# Internet Security [1]

## VU 188.366

### *TCP/IP - Part 2/2*

Paolo Milani Comparetti, Christian Platzer, Gilbert Wondracek,  
Markus Huber and Edgar Weippl

[inetsec@iseclab.org](mailto:inetsec@iseclab.org)

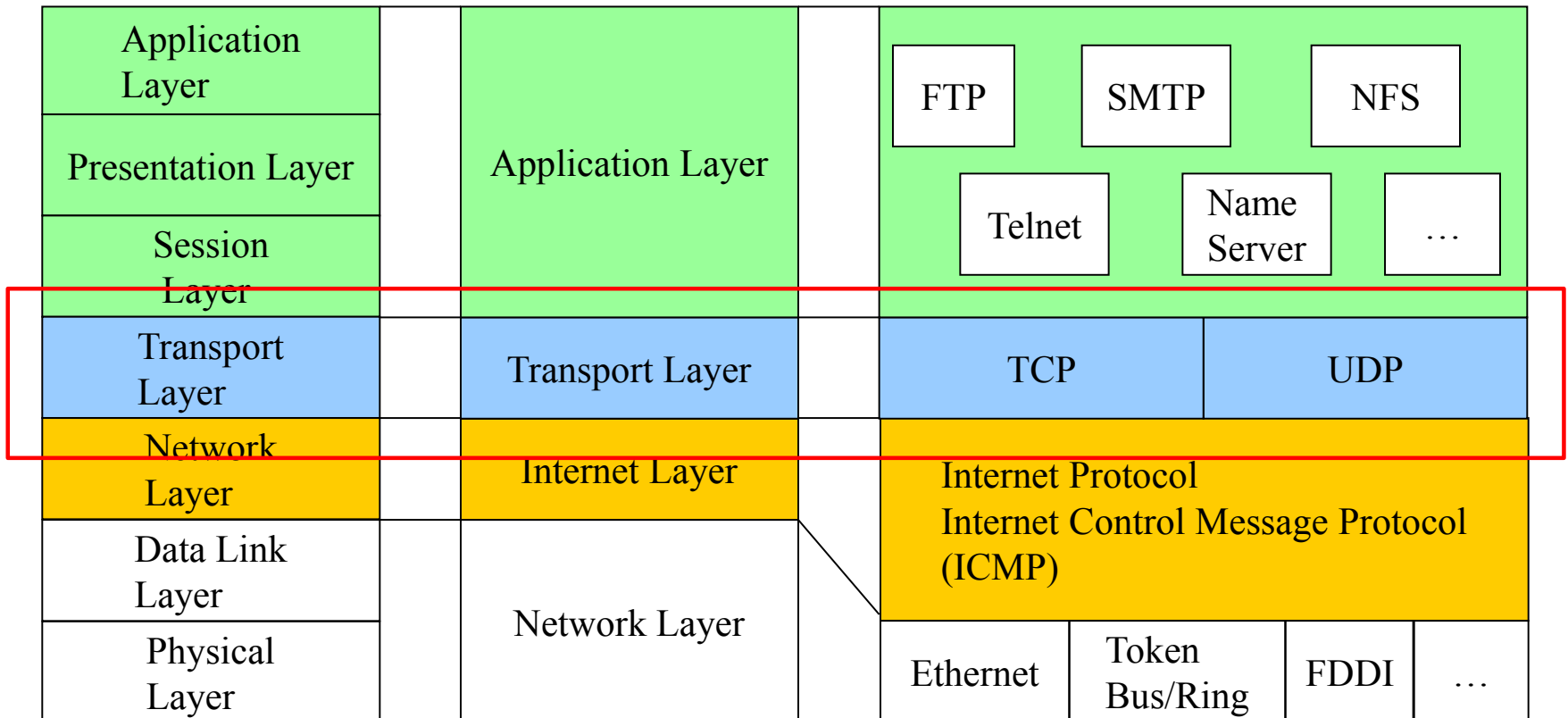
# News from the Lab

*Int. Secure Systems Lab  
Technical University Vienna*

- Registration closes next week
- Challenge 1
  - David "Evil Atom" Madner cracked it first, grats
  - One week to go
  - 43 so far, keep it coming
- 2nd challenge will start next week
  - Only 1 week this time!
  - SQL waits for you!

# Network Protocol Stack

*Int. Secure Systems Lab  
Technical University Vienna*



# Covered in the previous lectures..

*Int. Secure Systems Lab  
Technical University Vienna*

- Sniffing
  - Wireless
  - Ethernet
- Spoofing
  - MAC spoofing (Layer 2)
  - IP spoofing (Layer 3)
- Protocols (+ Attacks)
  - ARP (Layer 2/3, spoofing, flooding, etc.)
  - IP (Layer 3, Fragmentation, etc.)
  - Routing (RIP, BGP, OSPF...)
  - ICMP (Smurf, Redirect, Dest. Unreachable, etc.)

# In This Lecture: UDP and TCP

*Int. Secure Systems Lab  
Technical University Vienna*

- Many protocols use IP as the underlying network layer
- Important ones are
  - ICMP (Internet Control Message Protocol)
  - **UDP** (User Datagram Protocol)
  - **TCP** (Transmission Control Protocol)

# UDP

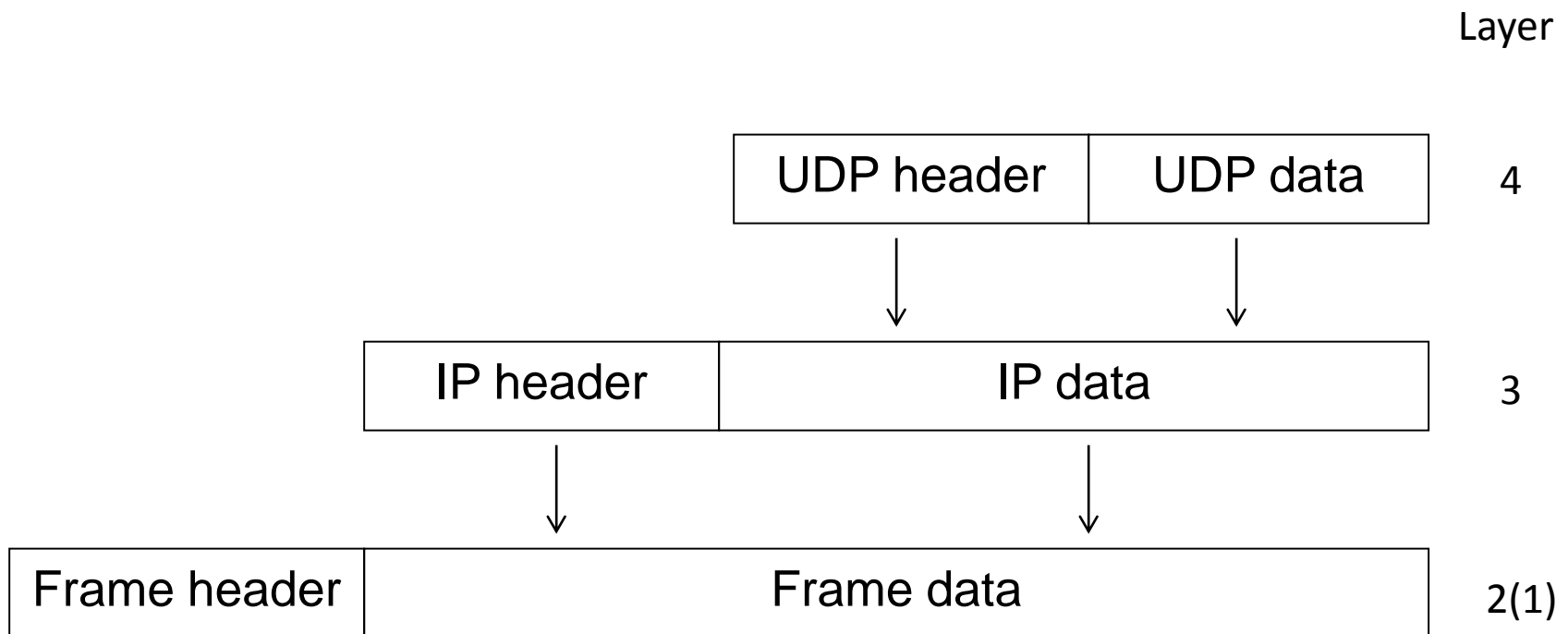
# User Datagram Protocol (UDP)

*Int. Secure Systems Lab  
Technical University Vienna*

- relies on IP
  - connectionless
  - unreliable (checksum optional)
  - best-effort
  - datagram delivery service
- 
- delivery, integrity, non-duplication and ordering are not guaranteed

# UDP Encapsulation

*Int. Secure Systems Lab  
Technical University Vienna*



# UDP is "unreliable"

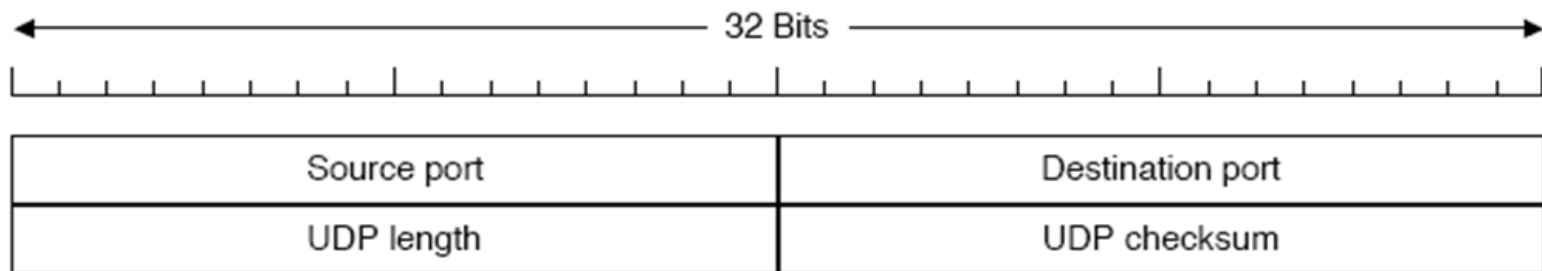
- UDP is based on IP
- IP networks may:
  - drop packets
  - corrupt packets (IP checksum only on headers!)
  - transmit packets out of order
  - duplicate packets
- UDP does not fix these problems
  - datagram may not arrive
  - may arrive corrupted (UDP checksum is optional)
  - may arrive out of order
  - may arrive in multiple copies



# UDP Message

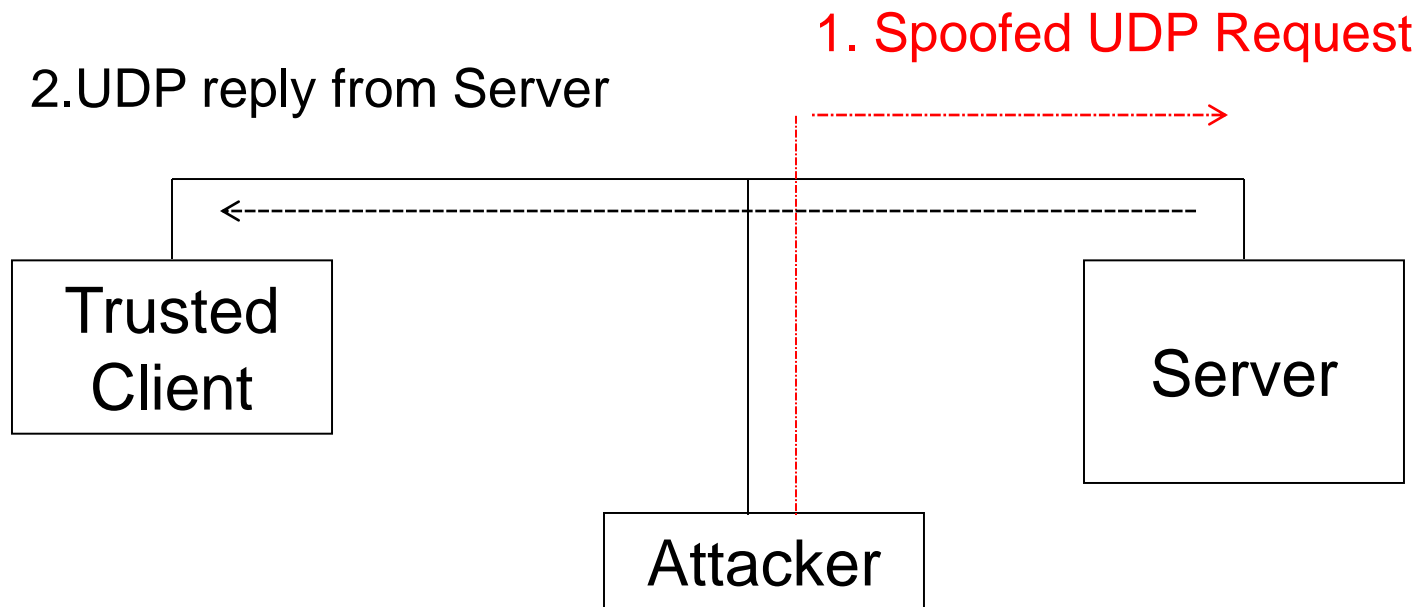
*Int. Secure Systems Lab  
Technical University Vienna*

- Port abstraction
  - allows addressing different destinations for the same IP
- Often used for multimedia
  - and for services based on request/reply schema (DNS, RPC, NFS)
- more efficient than TCP



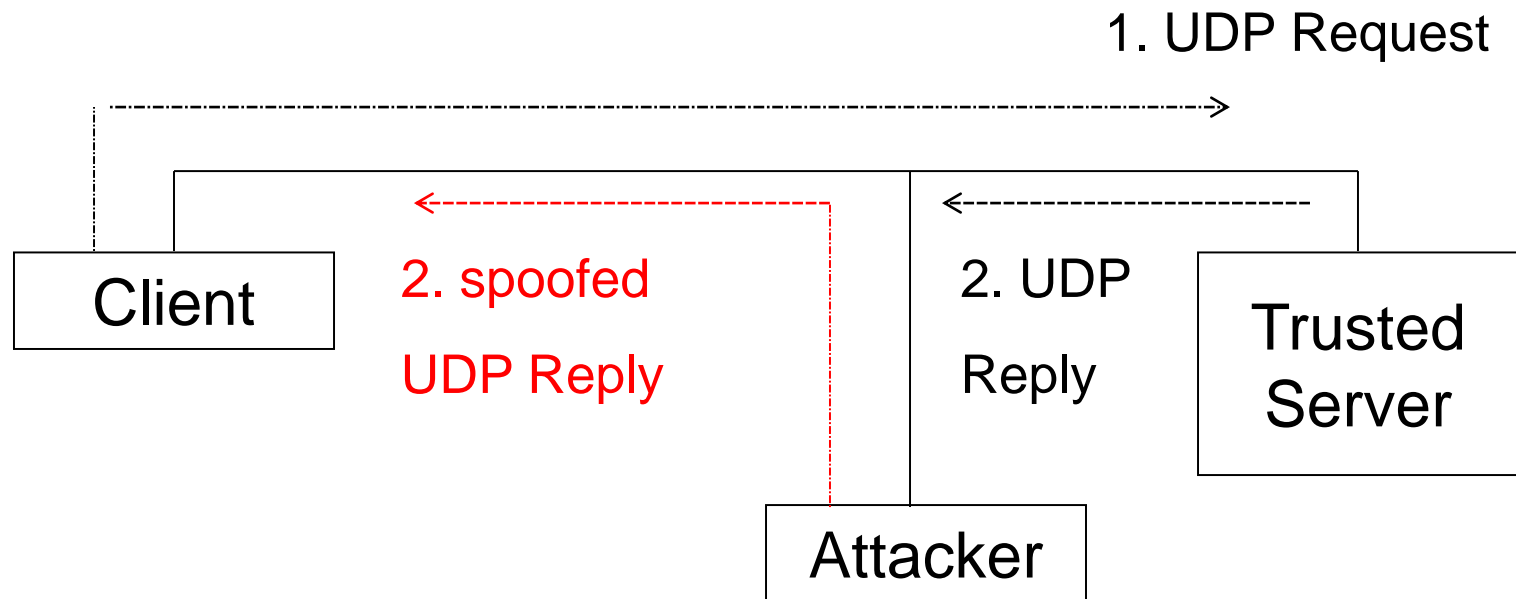
# UDP Spoofing

- Same as IP spoofing
  - just use trusted client's IP in IP source address field



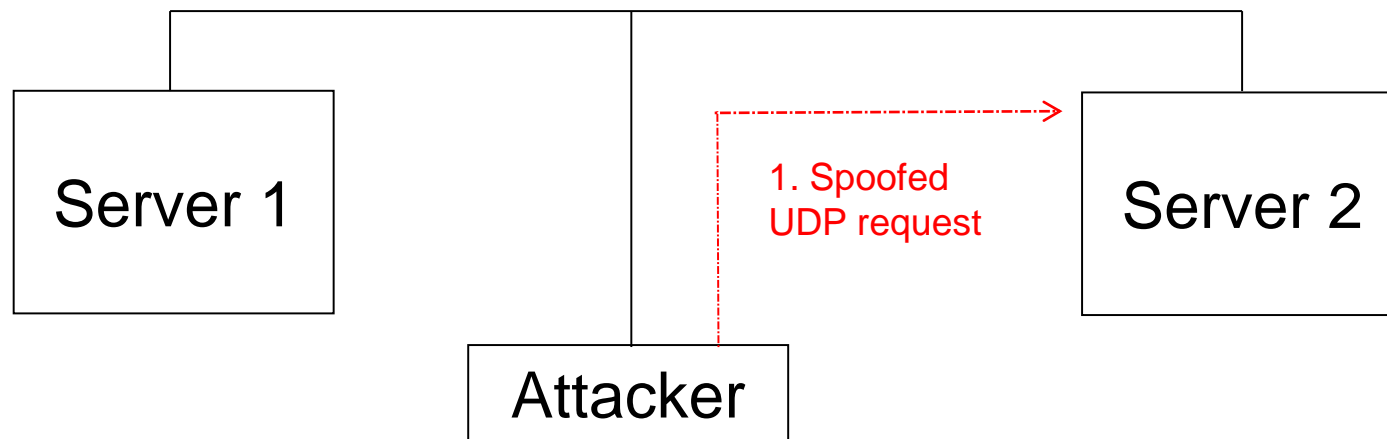
# UDP Hijacking

- Variation of the UDP spoofing attack
- Race against the legitimate server

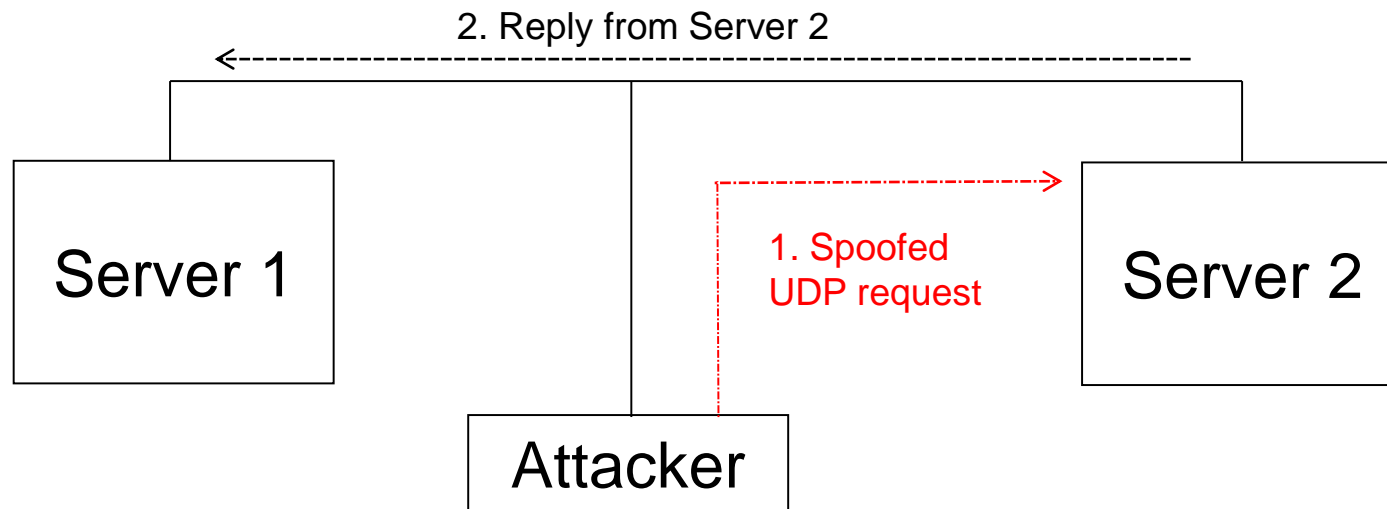


# UDP Storm

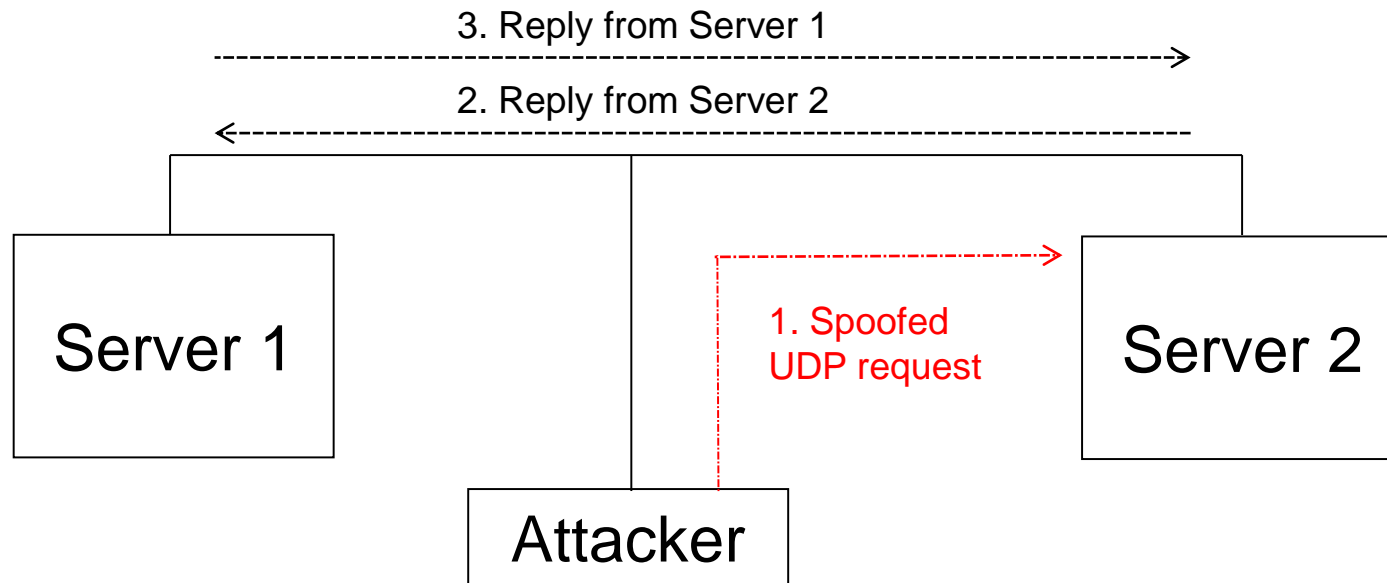
*Int. Secure Systems Lab  
Technical University Vienna*



# UDP Storm

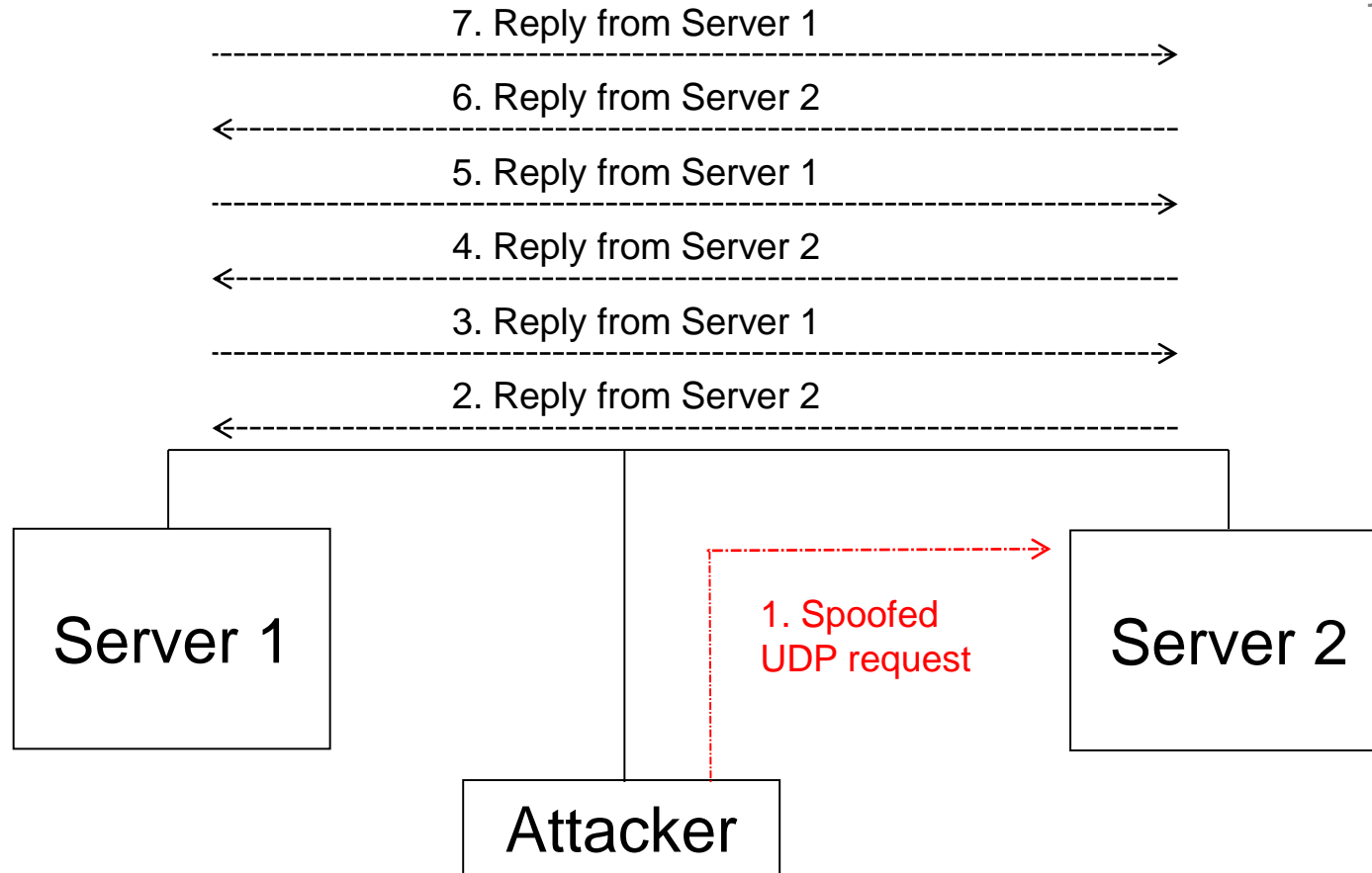


# UDP Storm



# UDP Storm

*Int. Secure Systems Lab  
Technical University Vienna*



# UDP Storm

- Need 2 hosts with UDP services that will reply to each other endlessly
- echo service (TCP/UDP port 7)
  - will reply to any message by echoing back the data it sent
  - useful for network troubleshooting
  - should be disabled in production networks
- chargen service (UDP/TCP port 19)
  - will reply to any message by sending back a random UDP packet
- In the old days (NT 4) → (telnet ntbox 19 | telnet ntbox 53)
- Classic UDP Storm attack: echo-chargen loop

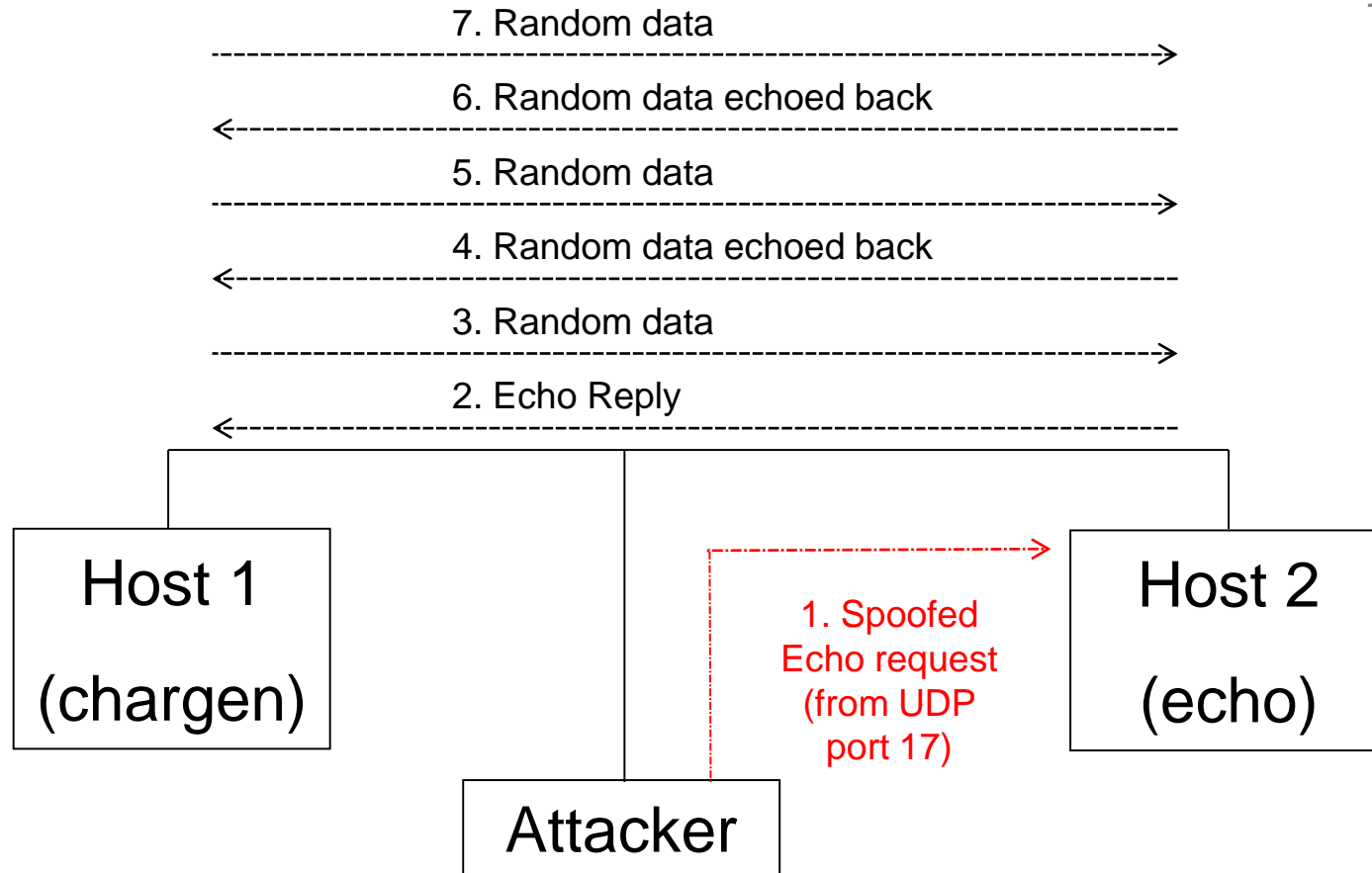
# Echo-Chargen loop

*Int. Secure Systems Lab  
Technical University Vienna*

- Spoofed UDP datagram is sent to the echo service (port 7)
- Source port is set to the chargen device (port 19)
- Reply of the echo service is interpreted as a request by the chargen service
- Reply of the chargen service is interpreted as a request by the echo service
  - DoS both hosts
  - Same attack can be carried out using two echo services
- If echo and chargen on same host, DoS single host

# Echo-Chargen Loop

Int. Secure Systems Lab  
Technical University Vienna



# UDP Portscan

- Which UDP ports are available on a certain host?
  - provide some network service
  - may be vulnerable to attack
- A **Portscan** is part of the information gathering phase of a network attack
- (Zero-length) UDP packet is sent to each port
  - If an ICMP error message „port unreachable“ is received, the service is assumed to be unavailable
  - if no reply, assume it is available
- Many TCP/IP stack implementations implement a limit on the error message rate, therefore this type of scan can be slow (e.g. Linux limit is 80 messages every 4 seconds)

# UDP Portscan

*Int. Secure Systems Lab  
Technical University Vienna*

## How to perform a UDP Portscan?

- by hand (with packet filter and RAW-socket)
- use netcat (<http://netcat.sourceforge.net/>) and tcpdump
- or use e.g. **nmap** -sU <address>  
(<http://www.insecure.org/nmap/>)

# TCP

# Transmission Control Protocol (TCP)

*Int. Secure Systems Lab  
Technical University Vienna*

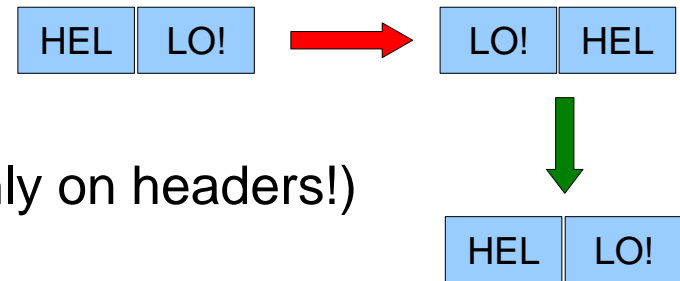
- relies on IP to provide:
  - connection-oriented
  - reliable
  - stream delivery service
  
- no loss
- no duplication
- no transmission errors
- correct data ordering

# TCP

- Provides (like UDP) the port abstraction
- Allows two nodes to establish a virtual circuit
- identified by a 4-tuple:
  - $\langle \text{src\_ip}, \text{src\_port}, \text{dst\_ip}, \text{dst\_port} \rangle$
- virtual circuit is composed of two streams (full duplex)
- The pair  $\langle \text{IP address}, \text{port} \rangle$  is called a socket

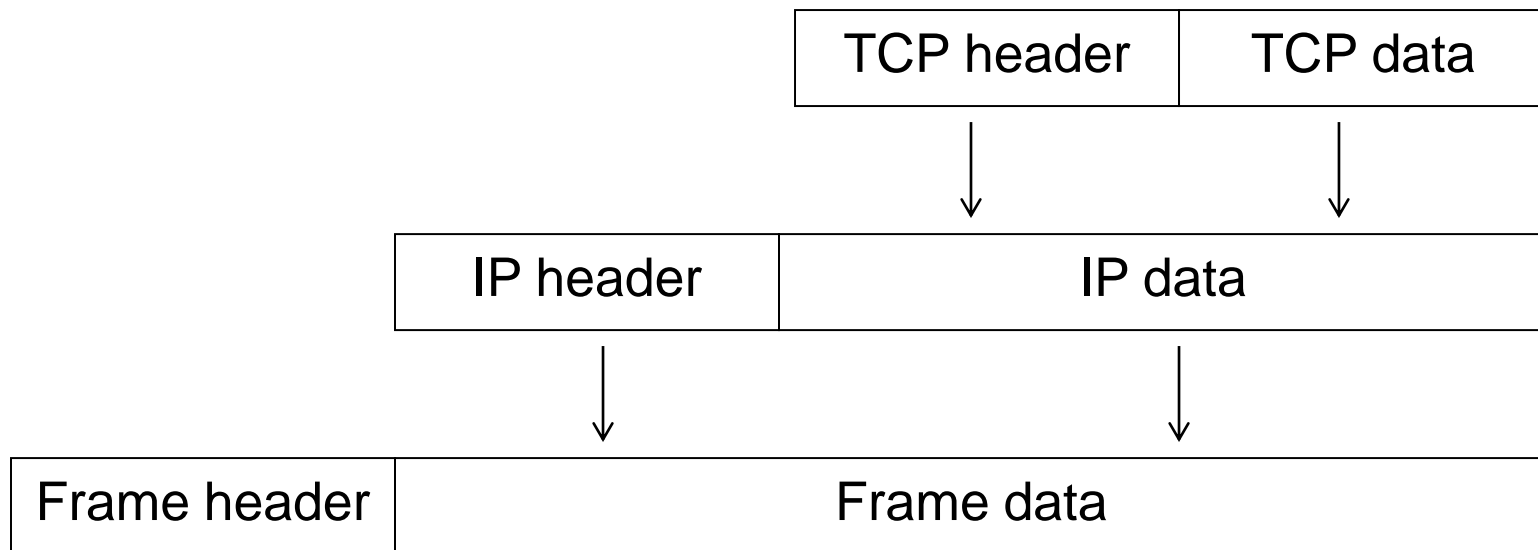
# TCP is reliable

- TCP is based on IP
- IP networks may:
  - drop packets
  - corrupt packets (IP checksum only on headers!)
  - transmit packets out of order
  - duplicate packets
- TCP fixes these problems
  - send acknowledgements to confirm that packets were received (otherwise, retransmit)
  - TCP has checksum to avoid data corruption
  - TCP segments carry enough information to reassemble them into a "stream" in the correct order



# TCP Encapsulation

*Int. Secure Systems Lab  
Technical University Vienna*



# TCP Segment

*Int. Secure Systems Lab  
Technical University Vienna*

source port (2 bytes)		destination port (2)	
sequence number (4 bytes)			
acknowledgement number (4 bytes)			
hlen	reserved	flags	window (2 bytes)
checksum (2 bytes)		urgent pointer (2 bytes)	
options			padding
data			

# TCP Seq/Ack Numbers

*Int. Secure Systems Lab  
Technical University Vienna*

- Sequence number (seq)
  - specifies the position of the segment data in the communication stream
  - seq = 1234 means:
    - The payload of this segment contains data starting from 1234
- Acknowledgement number (ack)
  - specifies the position of the next expected byte from the communication partner
  - ack = 12345 means:
    - I have received the bytes correctly to 12344, I expect the next byte to be 12345
- Both are used to manage error control
  - retransmission, duplicate filtering
  - also for flow control

# TCP Window

- Used to perform flow control
- Segment will be accepted only if the sequence number has a value between
  - last ack number sent and
  - last ack number sent + window size
- The window size changes dynamically to adjust the amount of information that can be sent by the sender
  - set by the receiver to announce how much it can take
  - window size = amount of data the client can handle now

# TCP Flags

- Flags are used to manage the establishment and shutdown of a virtual circuit
  - SYN: request for synchronization of seq/ack numbers (used during connection setup)
  - ACK: the acknowledgement number is valid (all segments in a virtual circuit have this flag set, except the first)
  - FIN: request to shutdown a virtual circuit (used during connection tear-down)
  - RST: request to immediately reset the virtual circuit
  - URG: states that the urgent pointer is valid
  - PSH request a „push“ operation on the stream (pass the data to the application (interactive) as soon as possible)

# TCP Options

*Int. Secure Systems Lab  
Technical University Vienna*

- Additional optional fields (some defined in later standards)
- Maximum Segment Size (MSS)
  - the size of the largest packet a partner is able to receive
  - set during setup phase
- Window scale factor
  - allows to specify a larger window of TCP data to accept (flow control)
  - otherwise at most 64k bytes "in flight" (un-ACKed)
- Timestamp
  - for TCP-Echo requests + responses (similar to ICMP)

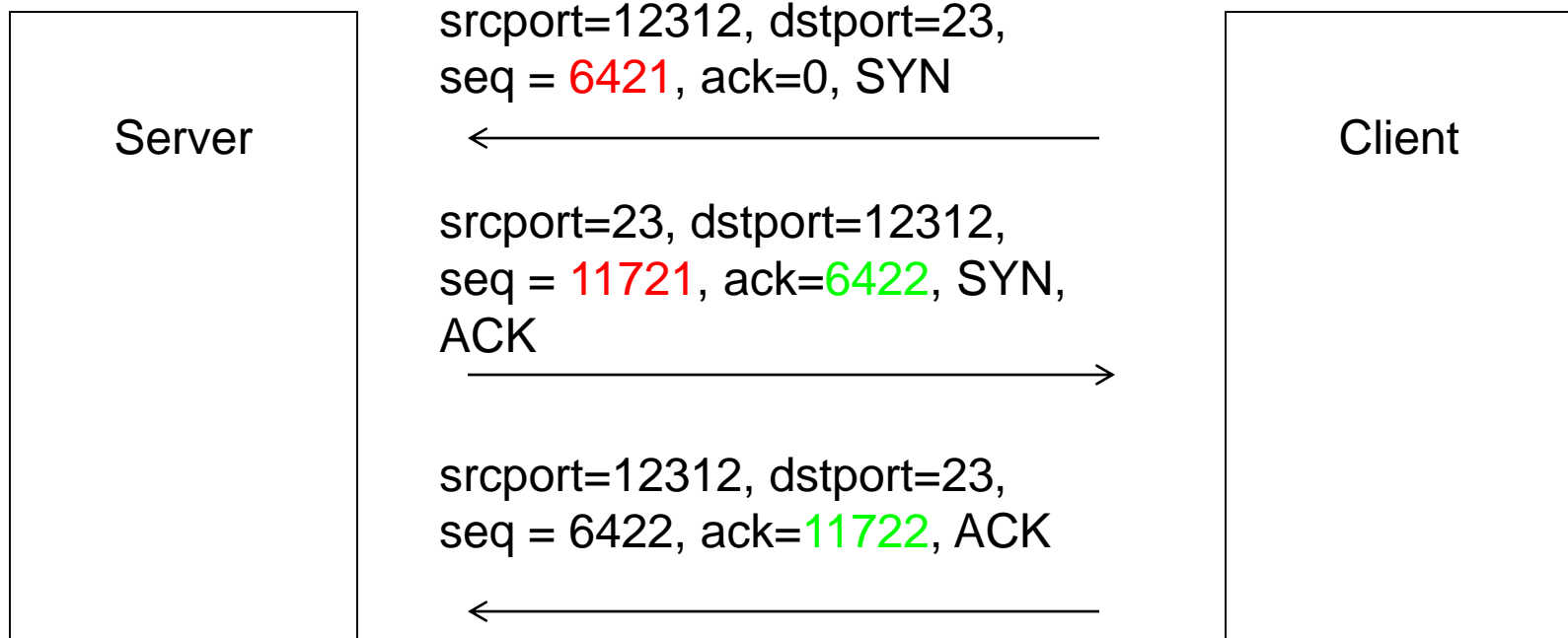
# TCP Virtual Circuit :: Setup

*Int. Secure Systems Lab  
Technical University Vienna*

- A server listens to a specific port
- Client sends a connection request to the server, with SYN flag set and a **random** initial sequence number  $c$
- The server answers with a segment marked with both the SYN and ACK flags and containing
  - an initial **random** sequence number  $s$
  - $c+1$  as the acknowledge number
- The client sends a segment with the ACK flag set and with sequence number  $c+1$  and ack number  $s+1$

# Three Way Handshake

- Three way because 3 TCP segments are necessary to set up a virtual circuit



# Initial Sequence Number

*Int. Secure Systems Lab  
Technical University Vienna*

- Needs to be random (unguessable) to prevent spoofing/hijacking attacks
  - in modern TCP/IP stack implementations, it is random
  - but the standard said otherwise!
- The TCP standard (RFC 793) specifies that the sequence number should be incremented every 4  $\mu$ s
- BSD UNIX systems initially used a number that is incremented by 64000 every half second and by 64000 each time a connection is established

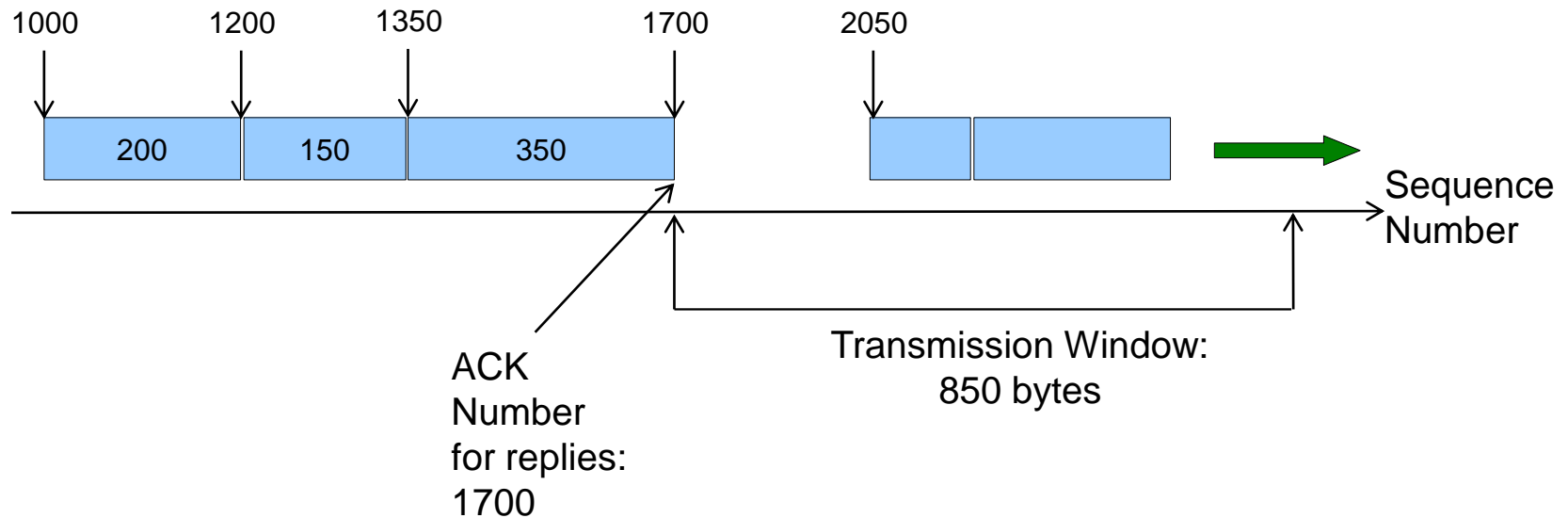
# TCP Data Exchange

*Int. Secure Systems Lab  
Technical University Vienna*

- Each TCP segment contains
  - sequence nr = position of data in stream (often, last ack number)
  - ack nr = sequence number of last correctly received segment increased by the payload size of that segment
- A partner accepts a segment of the other partner only if the numbers are inside the transmission window
- An empty segment may be used to acknowledge the received data
- Packets with no payload and SYN or FIN flag consume one sequence number

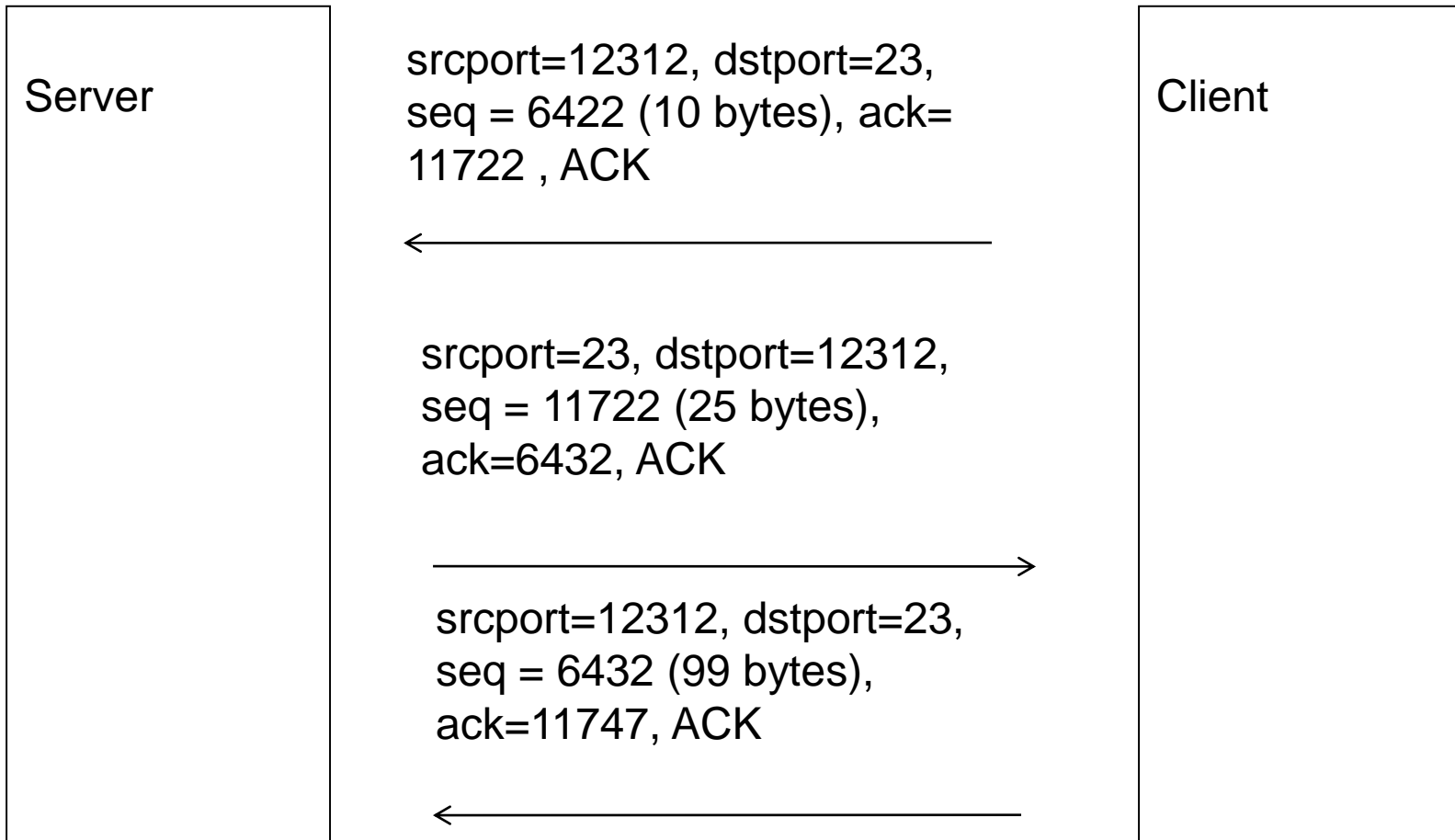
# Receiving TCP Data

Int. Secure Systems Lab  
Technical University Vienna



# TCP Data Exchange

*Int. Secure Systems Lab  
Technical University Vienna*



# Acknowledgement

- Not sent directly after data has been received
- delayed ACK: if some data has been received, the receiver waits up to 200 ms in hope that some more data will arrive, which can be acknowledged at once.
- delayed ACK is only used if no data has to be transported back to the sender
- If no ACK is received at the sender (timeout), retransmission takes place

# Virtual Circuit :: Shutdown

*Int. Secure Systems Lab  
Technical University Vienna*

One of the partners, e.g. A, can terminate its stream

- by sending a segment with the FIN flag set
- B answers with a segment with the ACK flag set
- From this point on A will not send any data to B: it will just acknowledge data sent by B
  - with empty segments
  - this is called a Half-Open connection
- When B shuts its stream down, the virtual circuit is considered closed

# Sample TCP Connection

*Int. Secure Systems Lab  
Technical University Vienna*

From	To	S	A	F	Seq-Nr	Ack-Nr	Payload
192.168.0.1	192.168.0.2	1	0	0	4711	0	0
192.168.0.2	192.168.0.1	1	1	0	38001	4712	0
192.168.0.1	192.168.0.2	0	1	0	4712	38002	0
192.168.0.2	192.168.0.1	0	1	0	38002	4712	,Login:\n' 7
192.168.0.1	192.168.0.2	0	1	0	4712	38009	,s' 1
192.168.0.1	192.168.0.2	0	1	0	4713	38009	,e' 1
192.168.0.1	192.168.0.2	0	1	0	4714	38009	,c' 1
192.168.0.1	192.168.0.2	0	1	0	4715	38009	,\n' 1
192.168.0.2	192.168.0.1	0	1	0	38009	4716	0
192.168.0.1	192.168.0.2	0	0	1	4716	38009	0
192.168.0.2	192.168.0.1	0	1	0	38009	4717	0

# TCP Security

*Int. Secure Systems Lab  
Technical University Vienna*

- Scanning
- OS Fingerprinting
- TCP Spoofing
- TCP Hijacking
- Denial of Service
  - SYN flooding
  - Process Table Attack

# TCP Scanning

*Int. Secure Systems Lab  
Technical University Vienna*

- TCP Portscan: information gathering phase of a network attack
- Used to check whether a port is open on a host
  - /etc/services lists standard port/service mappings
- Used to get some extra information about the host
- In the simplest form a TCP connection is opened to a port
  - if this succeeds a service is assumed to be available
  - this is reliable (unlike UDP scanning if port unreachable ICMP packets are not being sent out)

# TCP SYN Scan

*Int. Secure Systems Lab  
Technical University Vienna*

- Also known as „half open“ scanning
- The attacker sends a SYN packet (packet with SYN flag)
  - If the server answers with a SYN/ACK packet, then the port is open (or with a RST packet: the port is closed)
- The attacker sends a RST packet instead of an ACK
- Therefore the connection is never opened and the event is not logged by the operating system / monitor application

# TCP FIN Scan

*Int. Secure Systems Lab  
Technical University Vienna*

- The attacker sends a FIN-marked packet
- In most TCP/IP implementations (not Windows)
  - if the port is closed, a RST packet is sent back
  - if the port is open, the FIN packet is ignored
- Variations of this type of scanning technique
  - XMAS Scan: FIN + PSH + URG set
  - NULL Scan: no flags set

# OS Fingerprinting

*Int. Secure Systems Lab  
Technical University Vienna*

- Another step in information gathering phase of an attack
- allows to determine the operating system of a host by examining the reaction to carefully crafted packets
  - use of reserved flags in the TCP header
  - use of weird combination of flags in the TCP header
  - check the selection of TCP initial sequence numbers
  - analysis of response to particular ICMP messages
  - server response at a special port (Login)
- Each TCP/IP implementation is slightly different in handling corner cases

# NMAP

*Int. Secure Systems Lab  
Technical University Vienna*

- **Leading tool for portscanning** (and fingerprinting, and UDP scanning, and everything else except coffee)
- <http://www.insecure.org/nmap/>
- **supports**
  - IP scans
  - UDP portscans
  - TCP portscans (SYN, FIN scanning,...)
  - OS fingerprinting

# TCP Spoofing

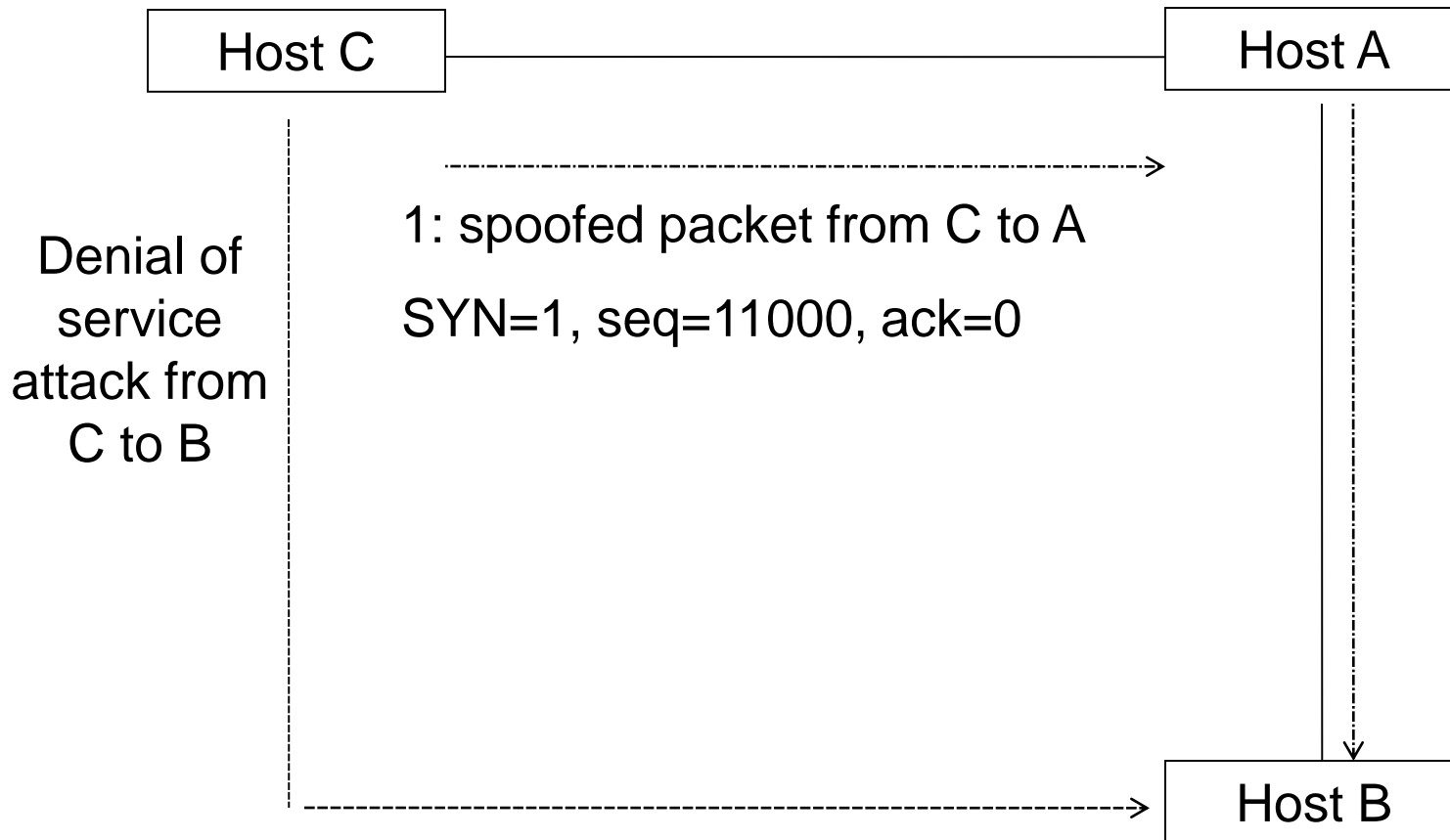
*Int. Secure Systems Lab  
Technical University Vienna*

- Attack aimed at impersonating another host
- mostly during the TCP connection establishment phase
- Node A trusts node B (e.g. login with no password)
- Node C wants to impersonate B with respect to A in opening a TCP connection
- C kills B (flooding, redirecting, crashing)
- C sends A a TCP segment in a spoofed IP packet with B's address as the source IP and an initial sequence number T

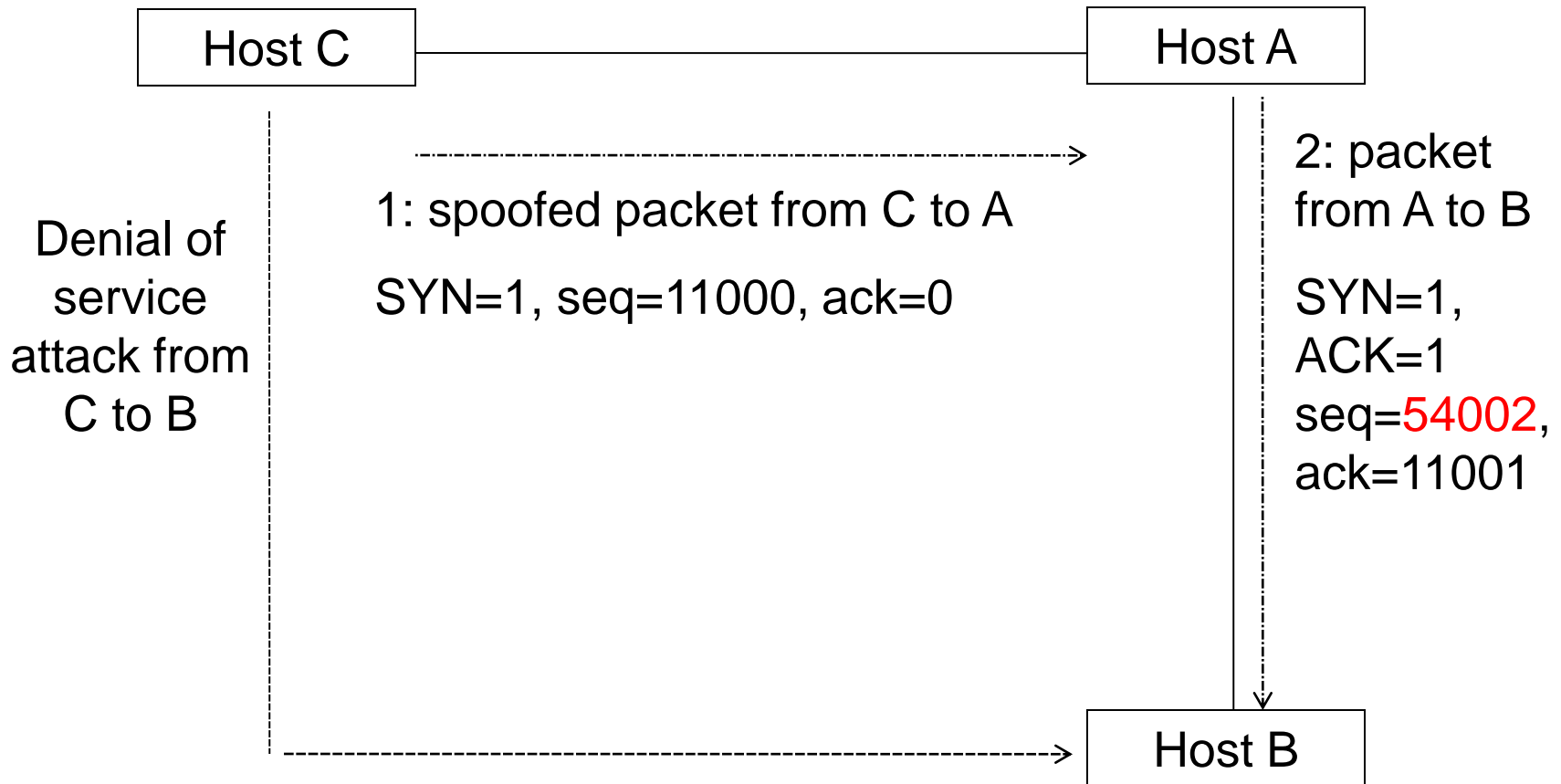
# TCP Spoofing

- A replies with a TCP SYN/ACK segment to B with S as the sequence number and T+1 as acknowledge number
- C does not receive the segment from A to B, but in order to finish the handshake it has to send an ACK segment with S+1 as the acknowledge number to A
- for this two possibilities exist:
  - C eavesdrops the SYN/ACK segment and calculates the number
  - C guesses the correct sequence number

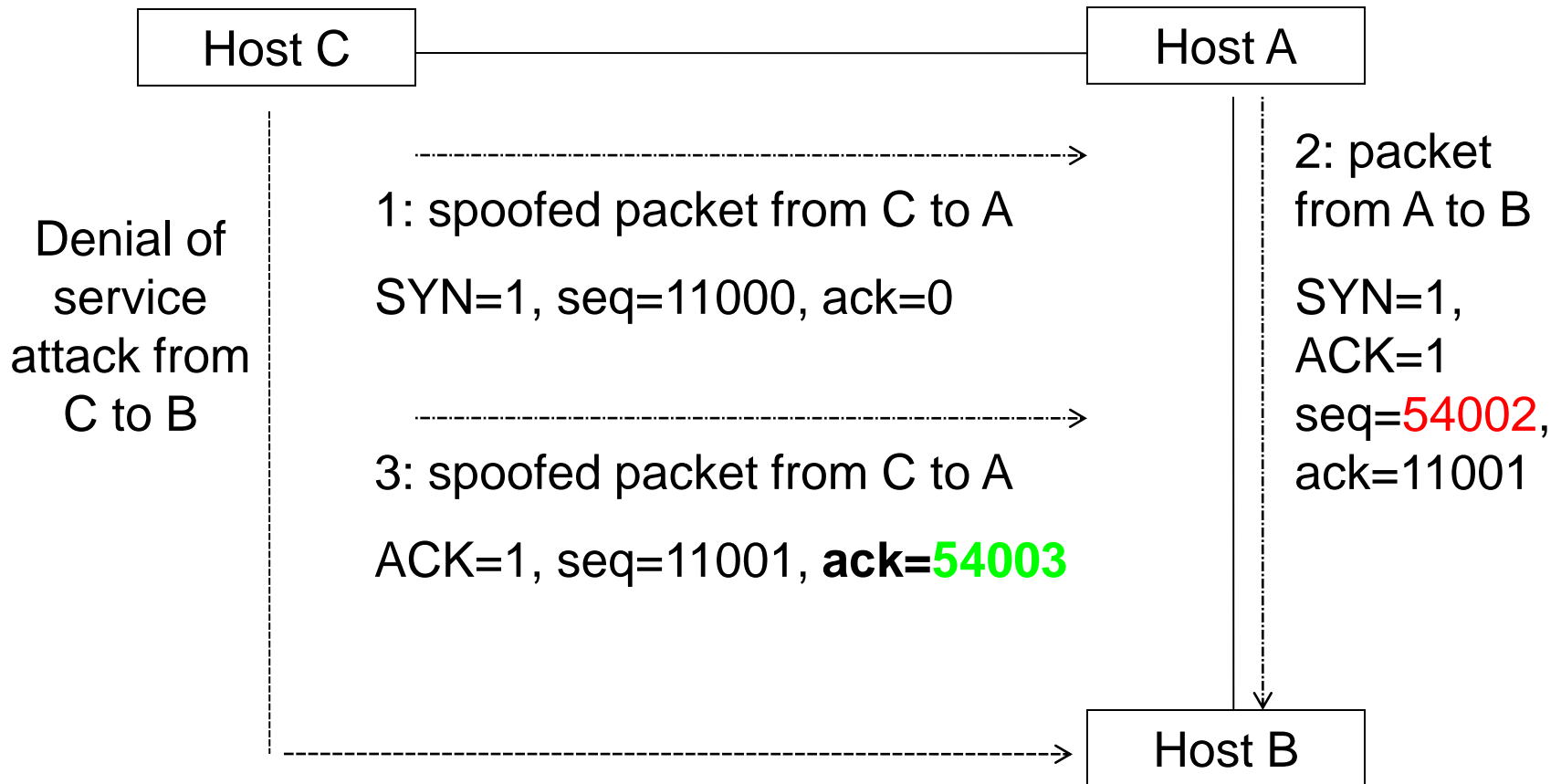
# TCP Spoofing



# TCP Spoofing



# TCP Spoofing



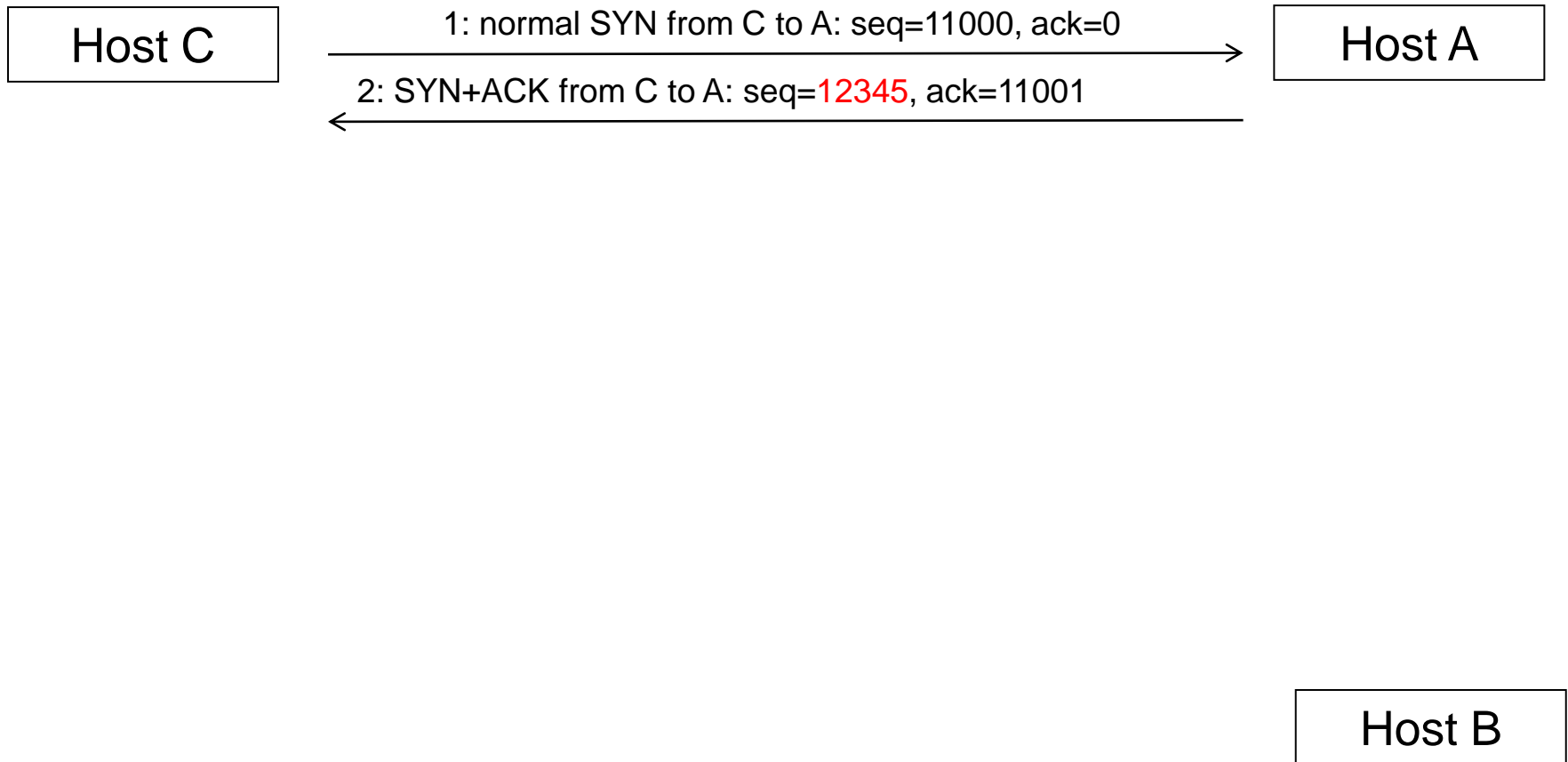
# Guessing Initial Sequence Number

*Int. Secure Systems Lab  
Technical University Vienna*

- Sequence number is 32 bits:
  - $2^{32} \approx 4$  billion possibilities
- If attacker cannot eavesdrop the SYN+ACK segment, how can he guess the **Initial Sequence Number**?
- If TCP/IP implementation provides suitably random **ISN**, TCP connections are mostly safe from spoofing
- Some implementations however do not generate new random **ISN** for each connection attempt
  - bad random number generator
  - increment by one on each attempt (example in following slides)
  - increment by one every  $\Delta t$

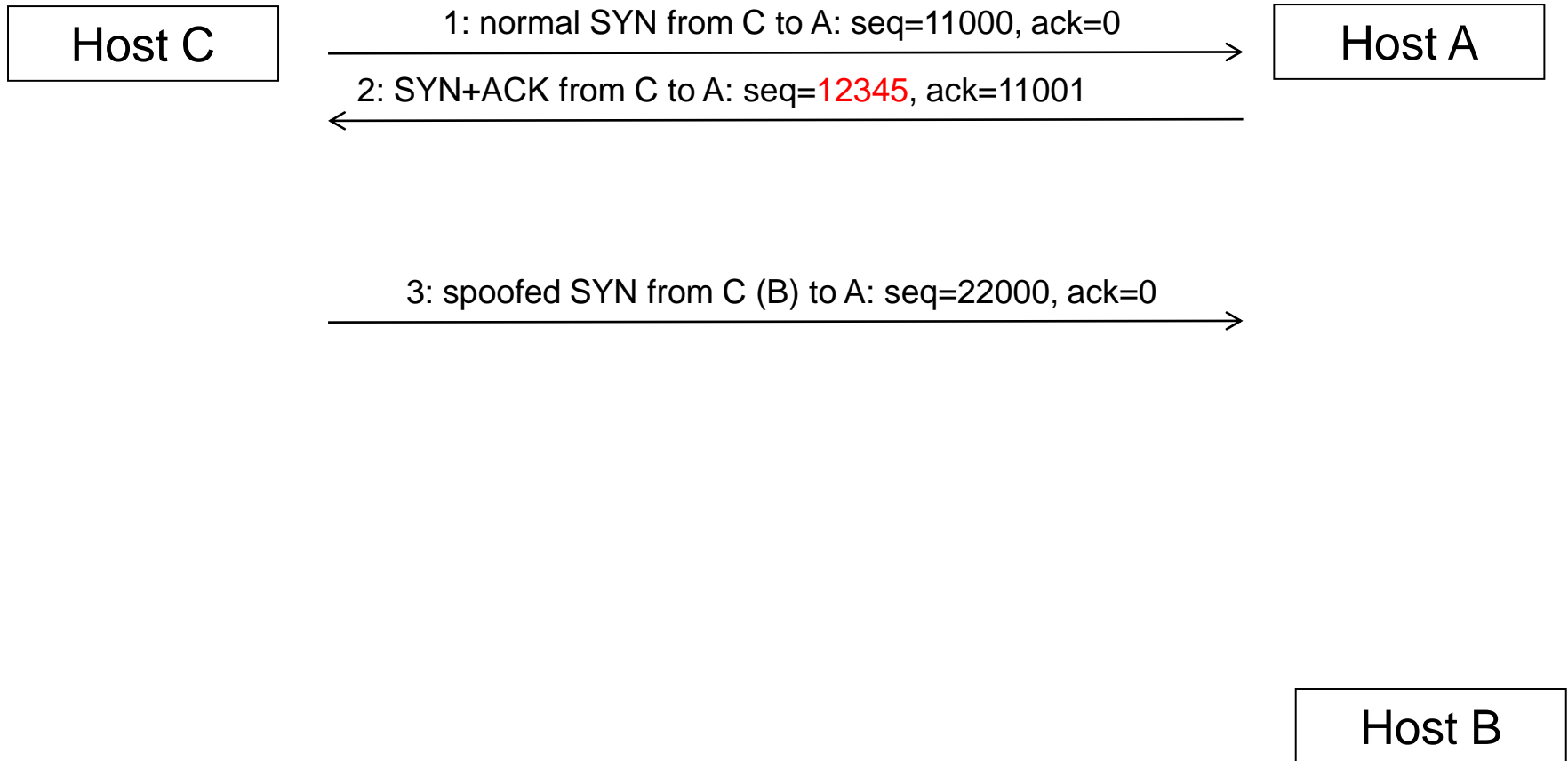
# TCP Spoofing

*Int. Secure Systems Lab  
Technical University Vienna*



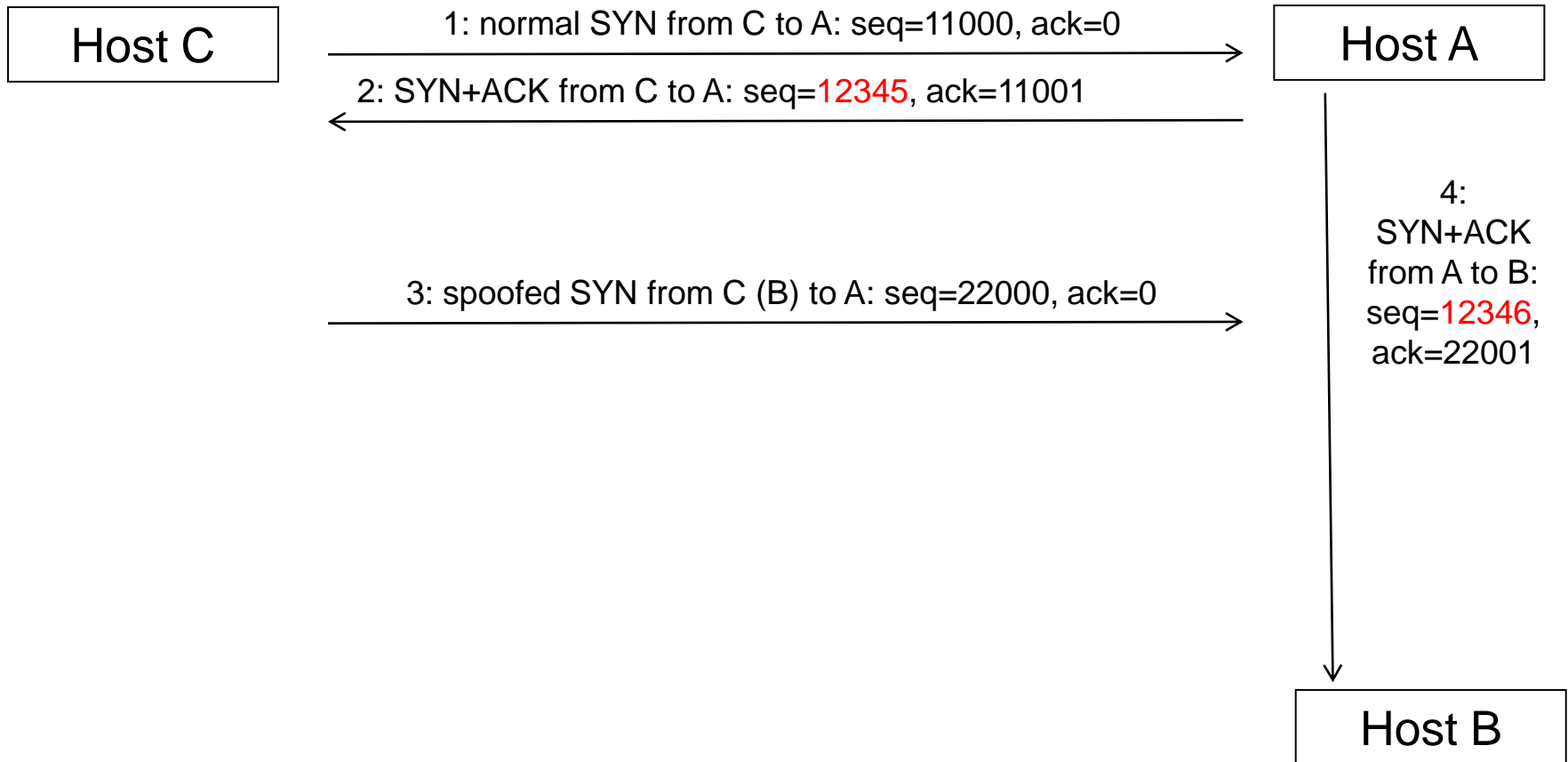
# TCP Spoofing

*Int. Secure Systems Lab  
Technical University Vienna*



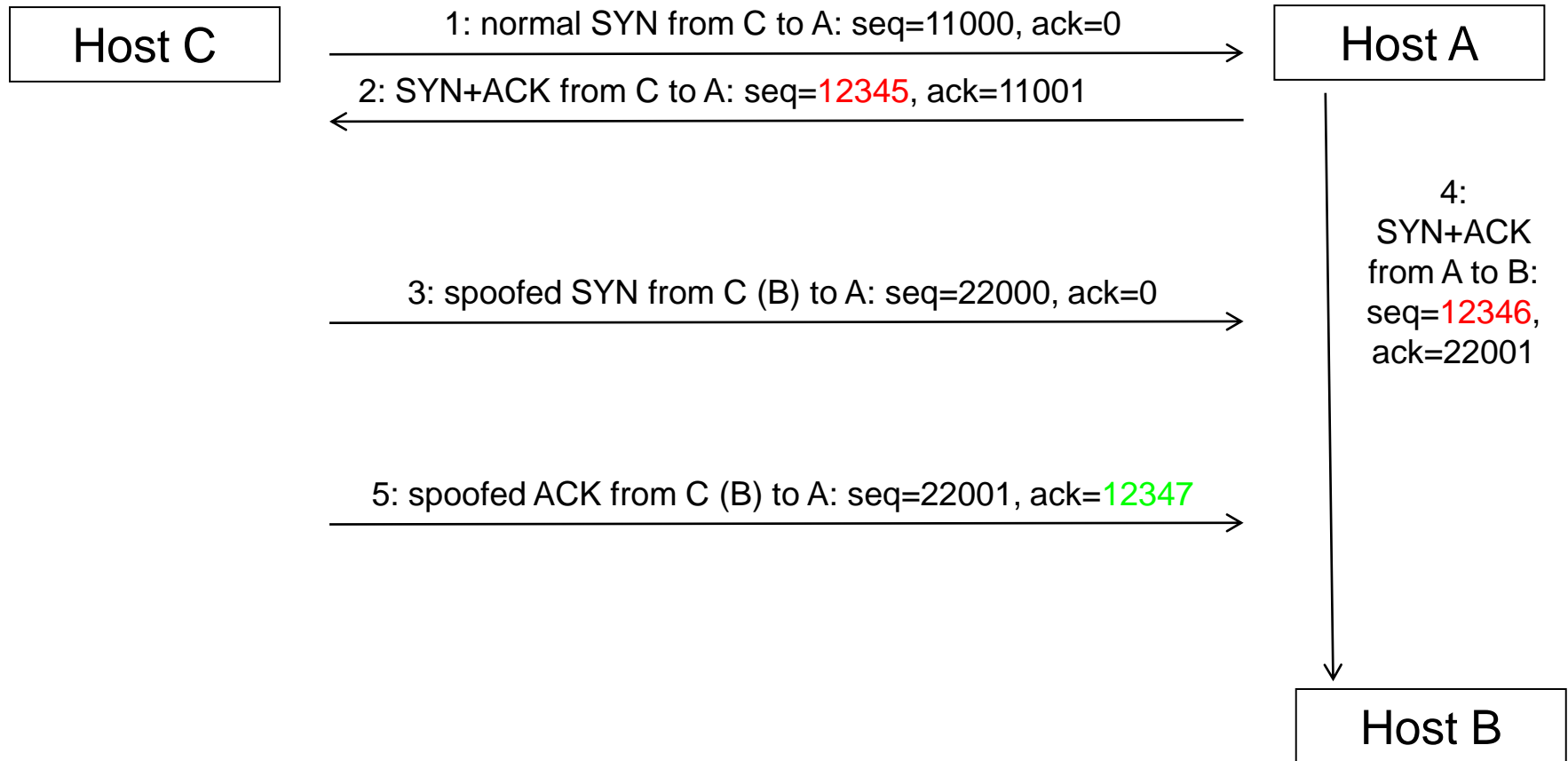
# TCP Spoofing

Int. Secure Systems Lab  
Technical University Vienna



# TCP Spoofing

Int. Secure Systems Lab  
Technical University Vienna



# TCP Hijacking

*Int. Secure Systems Lab  
Technical University Vienna*

- Technique to take control of an existing TCP connection
- The attacker uses spoofed TCP segments to
- insert data into the streams
  - or inject a RST packet to deny service
- But the correct sequence number must be used (guessed or eavesdropped by the attacker)

# Blind TCP Injection

*Int. Secure Systems Lab  
Technical University Vienna*

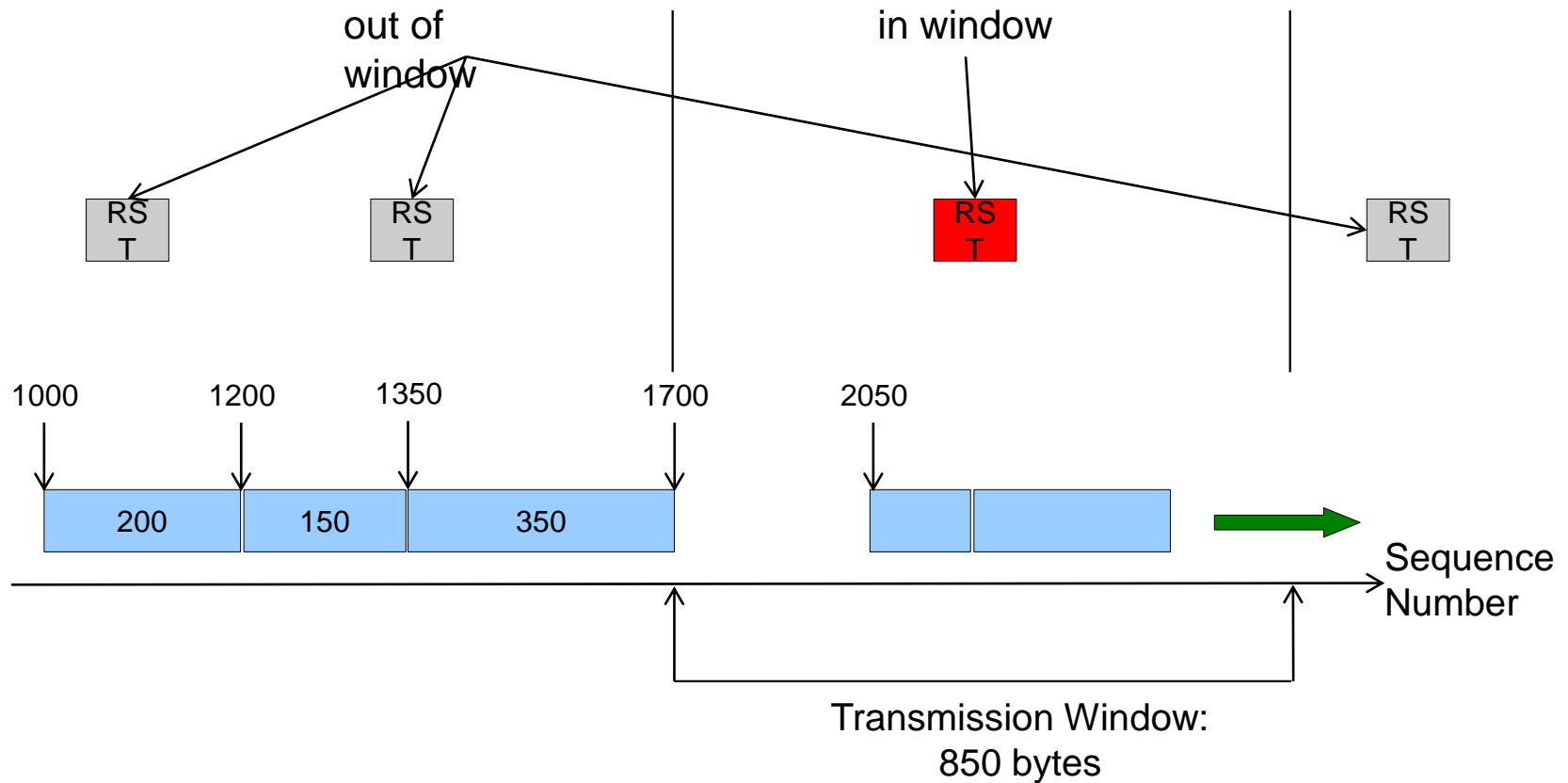
- Example: inject a RST packet to drop the connection
  - a new connection can be established, but this can create problems for some protocols (BGP)
- For the RST to be accepted attacker needs:
  - correct 4-tuple: client <IP,port> + server <IP,port>
  - correct seq number
- server port: generally known
- IPs: target server and spoofed client

# Blind TCP Injection

*Int. Secure Systems Lab  
Technical University Vienna*

- client port (16 bit):
  - if it is random, would need brute force
  - many OS use sequential port numbers (predict it)
- seq number (32 bit):
  - hard to brute force
  - do not need to guess exactly!
  - just needs to be **in window**.
  - window size is 16 bit (up to 64k)
  - window scaling allows it to be up to 30 bits!
- Attack:
  - Predict client port
  - Brute force seq number (send many spoofed packets)

# TCP RST Injection





# SYN Flooding Attack

*Int. Secure Systems Lab  
Technical University Vienna*

- very common denial-of-service attack
- attacker starts handshake with SYN marked segment
- victim replies with SYN-ACK segment
  - victim OS allocates data structures for the connection (reassembly buffer, etc)
- attacker's host stays silent
- a host can only keep a limited number of TCP connections in half-open state.
  - to limit memory usage
  - After that limit, connections are not accepted.
- Current solution
  - drop half open connections in FIFO manner
  - SYN cookies

# Process Table Attack

*Int. Secure Systems Lab  
Technical University Vienna*

- Daemons are programs that listen on a particular port for connection requests
  - When a new connection is established the daemon forks a new process that will handle the connection
- Many daemons run with root privileges (no restrictions)
  - A huge number of connections fill up the process table and no new processes can be created

# Conclusions

*Int. Secure Systems Lab  
Technical University Vienna*

- We finished our discussion of TCP/IP basics
  - Transport layer protocols and attacks (UDP/TCP)
- Good luck with challenge 1!
  - don't wait until the last minute to solve it, there will be NO extensions
- Any Questions?