Unix Security Features and TCP/IP Primer

Secure Software Programming and Vulnerability Analysis

Christopher Kruegel

Unix Security Features and TCP/IP Primer

Unix Features

- Multi-user operating system
- Process
  - implements user-activity
  - entity that executes a given piece of code
  - has its own execution stack, memory pages, and file descriptors table
- Thread
  - separate stack and program counter
  - share memory pages and file descriptor table

Unix - Process

- Process Attributes
  - process ID (PID)
    - uniquely identified process
  - user ID (UID)
    - ID of owner of process
  - effective user ID (EUID)
    - ID used for permission checks (e.g., to access resources)
  - saved user ID (SUID)
    - to temporarily drop and restore privileges
  - lots of management information
    - scheduling
    - memory management, resource management

Unix - User Model

- Unix is user-centric
  - no roles
- User
  - identified by user name (UID), group name (GID)
  - authenticated by password (stored encrypted)
- User `root`
  - superuser, system administrator
  - special privileges (access resources, modify OS)
  - cannot decrypt user passwords

Unix - Authentication

- Passwords
  - user passwords are used as keys for `crypt()` function
  - runs DES algorithm 25 times on a block of zeros
  - 12-bit “salt”
    - 4096 variations
    - chosen from date, not secret
    - prevent same passwords to map onto same string
    - make dictionary attacks more difficult
- Password cracking
  - dictionary attacks
  - `Crack, JohnTheRipper`

Unix - Authentication

/etc/passwd

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/bin/false
daemon:x:2:2:daemon:/sbin:/bin/false
adm:x:3:4:adm:/var/adm:/bin/false
lp:x:4:7:lp:/var/spool/lpd:/bin/false
chris:AcPyurst9Bfgz1:1000:100:Chris Kruegel:/home/chris:/bin/bash

username   password  UID  GID  complete name  home-dir  login-shell
```


Unix - Authentication

Automation Systems Group

- Authentication
  - prompt - `/bin/login`
  - user provides name and password
  - salt retrieved from `/etc/password`
  - zero block is encrypted
  - result compared to stored one
- Attacks
  - fake logins
  - tty tapping
  - social engineering

Unix - Authentication

- Shadow passwords
  - password file is needed by many applications to map user ID to user names
  - encrypted passwords are not

- `/etc/shadow`
  - holds encrypted passwords
  - account information
    - last change date
    - expiration (warning, disabled)
    - minimum change frequency
  - readable only by superuser and privileged programs
  - MD5 hashed passwords to slow down guessing

Unix - Group Model

- Users belong to one or more groups
  - primary group (stored in `/etc/password`)
  - additional groups (stored in `/etc/group`)
  - possibility to set group password
  - and become group member with `newgrp`

- /etc/group

```
root:x:0:root
bin:x:1:root,bin,daemon
users:x:100:chris

groupname : password : group id : additional users
```

Unix - File System

- File tree
  - primary repository of information
  - hierarchical set of directories
  - directories contain file system objects (FSO)
  - root is denoted "/"

- File system object
  - files, directories, symbolic links, sockets, device files
  - referenced by *inode* (index node)

Unix - File System

Automation Systems Group

- File System Object Attributes
  - type
  - size
  - reference counter
  - position on disk (disk block list)
  - UID and GID of owner
  - access and modification times
  - permission bits
  - but *no* file name!

- Directory
  - holds mapping between name and inode

Unix - File System

- Access Control
  - permission bits
  - `chmod, chown, chgrp, umask`
  - file listing:

```
             -      rwx      rwx      rwx
(file type) (user)  (group)  (other)
```

Type	r	w	x	s	t
File	read access	write access	execute	suid / sgid inherit id	sticky bit
Directory	list files	insert and remove files	stat / execute files, chdir	new files have dir-gid	files only delete-able by owner

Unix - SUID Programs

- Each process has *real* and *effective* user / group ID
  - usually identical
  - real IDs
    - determined by current user
    - `login`, `su`
  - effective IDs
    - determine the "rights" of a process
    - system calls (e.g., `setuid()`)
    - `suid` / `sgid` bits
  - attractive target for attacker

Unix - Resource Limits

- File system limits
  - *quotas*
  - restrict number of storage blocks and number of inodes
  - hard limit
    - can never be exceeded (operation fails)
  - soft limit
    - can be exceeded temporarily
  - can be defined per mount-point
  - defend against resource exhaustion (denial of service)

- Process resource limits
  - number of child processes, open file descriptors

Unix - Signals

- Signal
  - simple form of interrupt
  - asynchronous notification
  - can happen anywhere for process in user space
  - used to deliver segmentation faults, reload commands, …
  - `kill` command

- Signal handling
  - process can install signal handlers
  - when no handler is present, default behavior is used
    - ignore or kill process
  - possible to catch all signals except SIGKILL (-9)

Unix - Signals

- Security issues
  - code has to be be re-entrant
    - atomic modifications
    - no global data structures
  - race conditions
  - unsafe library calls, system calls

- Secure signals
  - write handler as simple as possible
  - block signals in handler

Unix - Communication

- Half-duplex pipes
  - connect output of one process to input of another
  - information flows uni-directional
  - classic use in shell programming (via | character)
  - represented by a file (inode) in kernel but not in file system

- Named pipes
  - much like regular pipes
  - exist as a device special file in the file system
  - processes of different ancestry can share data
  - when I/O is done by sharing processes, the named pipe remains in the file system

Unix - Communication

- AT&T System V IPC
  - inter-process communication primitives
  - shared memory, semaphores, message queues
  - standard access control mechanisms apply

- BSD Sockets
  - mostly used for network connections
  - local sockets possible
    - e.g., to implement pipes
  - appear as objects in file system
    - but cannot use open
  - more on sockets later in the TCP/IP section

Unix - Shared Libraries

- Library
  - collection of object files
  - included into (linked) program as needed
  - code reuse

- Shared library
  - multiple processes share a single library copy
  - save disk space (program size is reduced)
  - save memory space (only a single copy in memory)

Unix - Shared Libraries

- Static shared library
  - address binding at link-time
  - not very flexible when library changes
  - code is fast
- Dynamic shared library
  - address binding at load-time
  - procedure linkage table (PLT) and global offset table (GOT)
  - code is slower (indirection)
  - loading is slow (binding has to be done at run-time)
  - management issues (semantic changes)
  - classic `.so` or `.dll` libraries

Unix - Shared Libraries

- Management
  - stored in special directories (listed in `/etc/ld.so.conf`)
  - manage cache with `ldconfig`
- Preload
  - override (substitute) with other version
  - use `/etc/ld.so.preload`
  - can also use environment variables for override
  - possible security hazard
  - disabled for SUID programs

Unix Security Features
and
TCP/IP Primer

OSI Reference Model

- Developed by the ISO to support open systems interconnection
  - layered architecture, level n uses service of (n-1)

	Host A	Host B
7	Application Layer	Application Layer
6	Presentation Layer	Presentation Layer
5	Session Layer	Session Layer
4	Transport Layer	Transport Layer
3	Network Layer	Network Layer
2	Data Link Layer	Data Link Layer
1	Physical Layer	Physical Layer

OSI Reference Model

- Physical Layer
  - connect to channel / used to transmit bytes (= network cable)
- Data Link Layer
  - error control between adjacent nodes
- Network Layer
  - transmission and routing across subnets
- Transport Layer
  - ordering
  - multiplexing
  - correctness

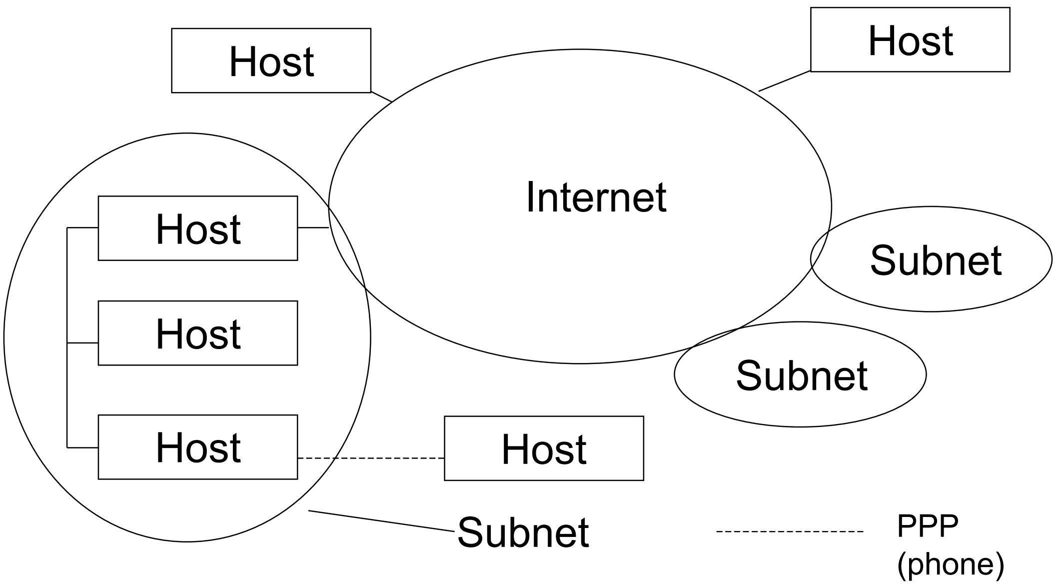
OSI Reference Model

- Session Layer
  - support for session based interaction
  - e.g. communication parameters/communication state
- Presentation Layer
  - standard data representation
- Application Layer
  - application specific protocols

TCP / IP

TCP/IP		OSI-Reference
Telnet	SMTP	Application
TCP		Transport
Internet Protocol (IP)		Network
Ethernet		Data Link Layer
NIC		Physical Layer

Internet



Internet Protocol (IP)

- Is the glue between hosts of the Internet
- Standardized in RFC 791

- Packet-based service
  – packets have a maximum size of 2^{16} bytes

- Attributes of delivery
  – connectionless
  – unreliable best-effort datagram
    - delivery, integrity, ordering, non-duplication are NOT guaranteed

Internet Protocol (IP)

- IP packets (datagrams) can be exchanged by any two nodes that are set up as IP nodes (i.e., that have IP addresses)

- For point-to-point communication
  – IP is tunneled over lower level protocols
    - Ethernet
    - Token Ring
    - FDDI
    - PPP, etc.

- Standardized data ordering
  – network byte-order = big endian
  – x86 host byte-order = little endian

IP Address

- IP addresses in IPv4 are 32 bit numbers
  - (class+net+host ID)
- Each host has a unique IP address for each NIC
- Represented as dotted-decimal notation:
  - 10000000 10000011 10101100 00000001 = 128.131.172.1

Classes:	<starts with>	<net-bits>	<host-bits>	<#of possible hosts>
Class A:	0	7	24	16777216
Class B:	10	14	16	65536
Class C:	110	21	8	256
Class D:	1110	special meaning: 28 bit multicast address		
Class E:	1111	reserved for future use		

IP Subnet

- It is often unrealistic to have networks with so many hosts
  - further divide the hostbits into subnet ID and host ID
  - saves address space

- Example: Class C normally has 24 netbits

  Class C network with subnet mask 255.255.255.240

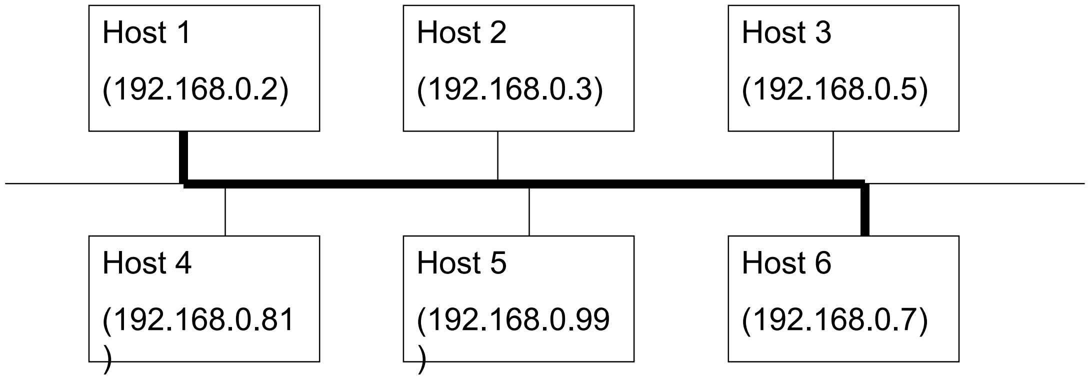
  240 = 1111 0000

  | host ID => 16 hosts within every subnet

  subnet ID => 16 subnets within this class C network

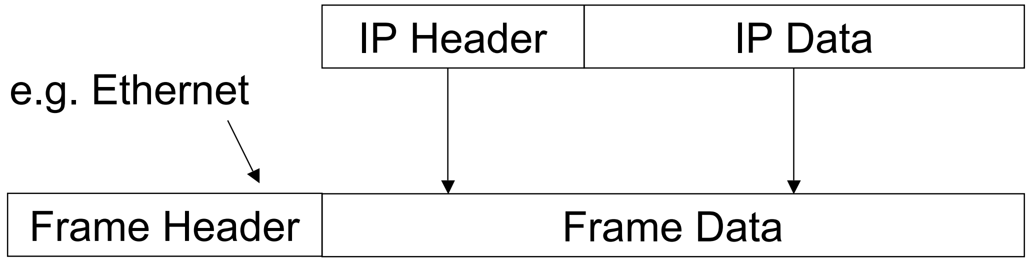
IP - Direct Delivery

- If two hosts are in the same physical network the IP datagram is encapsulated in a Layer 2 frame and delivered directly



IP Encapsulation

- IP packet included in Layer 2 frame
  - e.g., Ethernet (RFC 894 - IP over Ethernet)



Ethernet

- Widely used link layer protocol
- Carrier Sense, Multiple Access (CSMA) with Collision Detection
- Addresses: 48 bits (e.g. 00:38:af:23:34:0f)
- Frame
  - 2 x 6 bytes addresses (destination and source)
  - 2 bytes frame data type
    - specifies encapsulated protocol, IP, ARP, RARP
  - variable length data
  - 4 bytes CRC
- Frame Length
  - minimum of 64 bytes frame length
    - padding may be needed
  - maximum of 1518 bytes

IP - Direct Delivery

Problem:

- Ethernet uses 48 bit addresses
- IP uses 32 bit addresses

- We want to send an IP datagram
  but we only can use the Link Layer to do this

ARP

- Solution - ARP (Address Resolution Protocol)

- Service at the link-level, RFC 826
- Maps IP network addresses to Ethernet link-level addresses

- Scenario:
  - host A wants to know the Ethernet address associated with IP address of host B
  - A broadcasts ARP message on physical link (including its own mapping)
  - B answers A with ARP answer message

- Mappings are cached
  - `arp -a` shows mapping

Fragmentation

- Fragmentation
  - when datagram size is larger than data link layer MTU (Maximum Transmission Unit)
  - performed at
    - source host
    - or intermediate steps (e.g., routers)

- Reassembly
  - rebuilding the IP packet
  - only performed at the destination

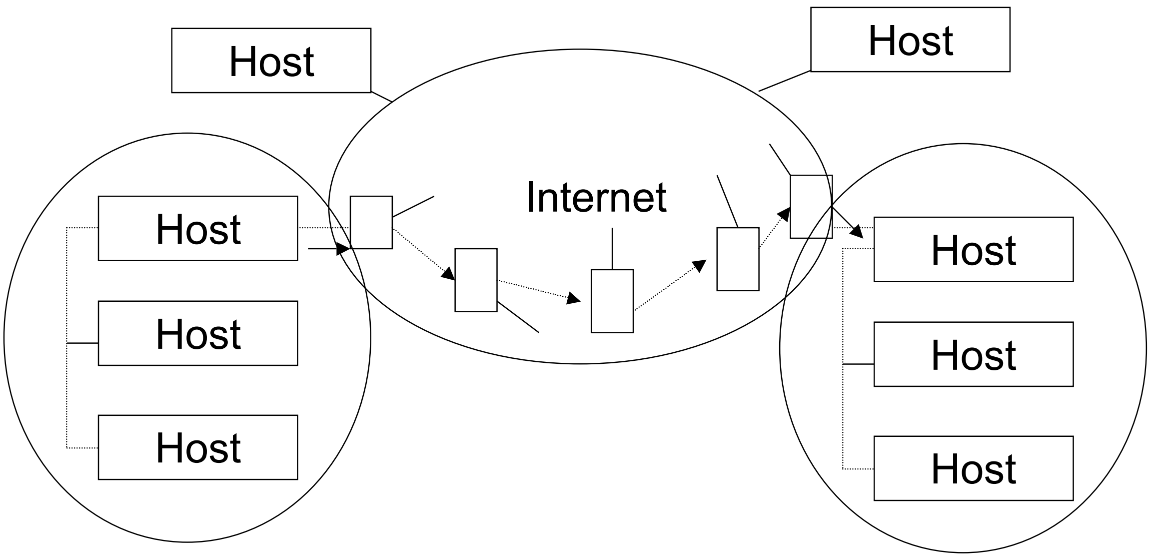
- Each fragment is delivered as a separate datagram

IP - Indirect Delivery

- Routing
  - needed if hosts are in different physical networks
  - packet can't be delivered directly
- Packet is forwarded to a router (gateway)
  - router has access to other network(s)
  - router decides upon destination where to send the packet next
  - this is repeated until packet arrives at network with target host
  - then direct delivery is performed
  - link level addresses change at every step, also TTL field

IP - Indirect Delivery

- Store and forward communication



Routing Table

- Contains information how to do hop-by hop routing

```
% route -n
Kernel IP routing table
Destination      Gateway          Genmask          Flags    Iface
192.168.1.0      0.0.0.0          255.255.255.0    U        eth0
loopback         127.0.0.1        255.0.0.0        UG       lo
0.0.0.0          192.168.1.1      0.0.0.0          UG       eth0
```

- Flags:
  - U: the route is up
  - G: use gateway for destination

Routing Mechanism

- Route-daemon searches for
  - matching host address
  - matching network address
  - default entry
- If no route can be found: ICMP message
  - „Host unreachable“ is sent back to originator
- Routing tables can be set
  - statically
  - dynamically (using routing protocols)

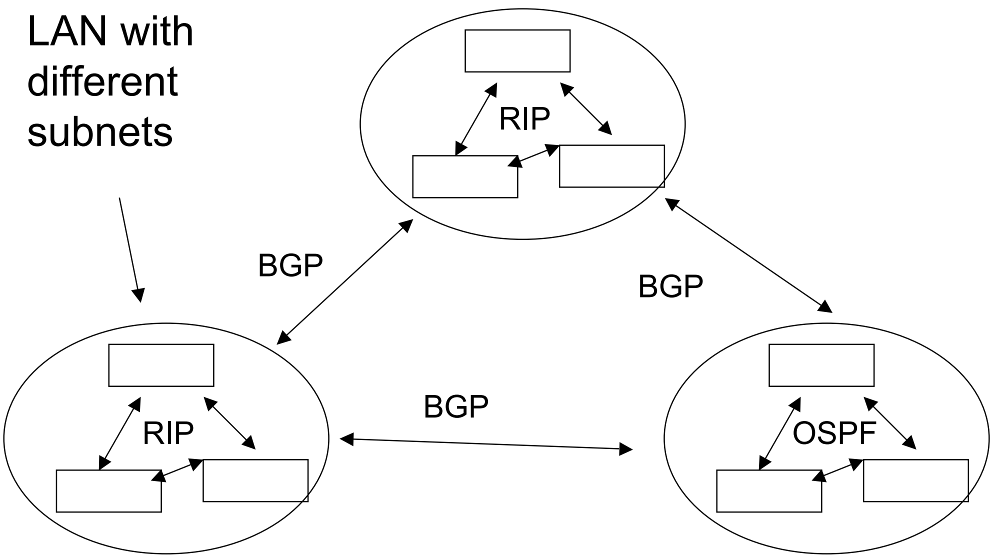
Routing Protocols

- automatically distribute information about delivery routes
- hierarchically organized with different scope

- divided in
  - exterior gateway protocols (EGPs)
    - distribute information between different autonomous systems
    - e.g., Border Gateway Protocol (BGP) for Internet backbone
  - interior gateway protocols (IGP)
    - distribute information inside autonomous systems, e.g. in LANs
    - e.g., Routing Information Protocol (RIP)
    - e.g., Open Shortest Path First (OSPF)

- autonomous means: under a single administrative control

Routing Protocols



User Datagram Protocol

- UDP (User Datagram Protocol)
  - based on IP
- Connectionless
  - based on datagrams
- Best-effort service
  - delivery
  - non-duplication
  - ordering are not guaranteed
- Unreliable (checksum optional)

UDP Message

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

UDP source port (2 bytes)	UDP destination port (2)
UDP message length (2)	Checksum (2)
Data (up to 2^{16})	

Transmission Control Protocol

- TCP (Transmission Control Protocol)
  - based on IP
- Connection-oriented
  - based on streams
- Reliable service
  - delivery
  - non-duplication
  - ordering are guaranteed
- Checksum mandatory
- Uses acknowledgements sent by receiver

TCP

- Provides port abstraction
  - like UDP
- Allows two nodes to establish a virtual circuit
  - identified with quadruples
    <srcip, src_port, dstip, dst_port>
  - virtual circuit is composed of two streams (full duplex)
- The pair <IP address, port> is called a *socket*

TCP Sequence Numbers

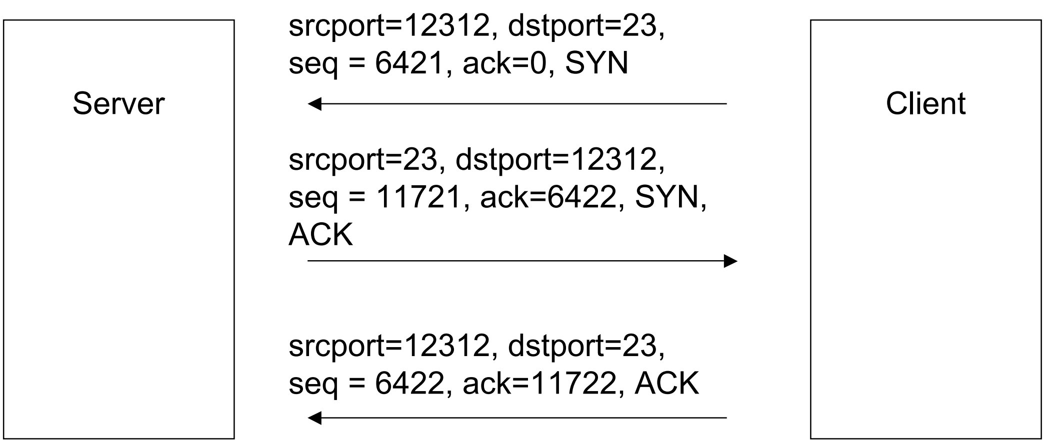
- Sequence number
  - specifies the position of the segment data in the communication stream
  - (SEQ=1234 means: the payload of this segment contains data starting from 1234)
- Acknowledgement number
  - specifies the position of the next expected byte from the communication partner
  - (ACK=12345 means: 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

TCP Virtual Circuit Setup

- 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

TCP Virtual Circuit Setup

- TCP Three way handshake



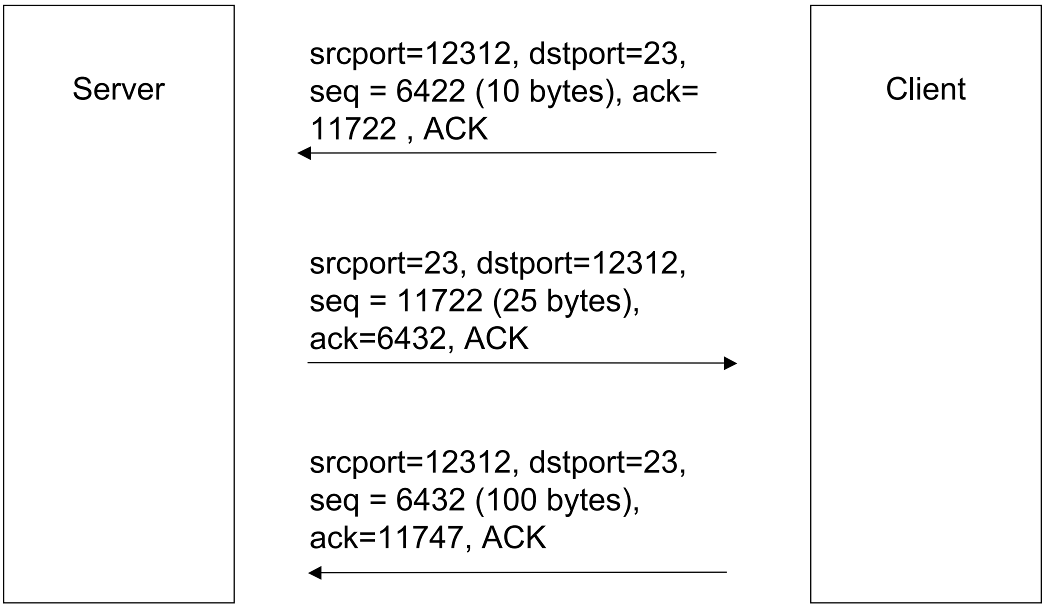
TCP Data Exchange

- Each TCP segment contains
  - sequence number = ack number of last received packet
  - ack number = sequence number of last correctly received segment increase by the payload size of this 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 set consume this sequence number

TCP Data Exchange



Virtual Circuit Shutdown

- One of the partners, e.g., A, wants to terminate its stream
  - sends 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
  - just acknowledge data sent by B with empty segments
- When B shuts its stream down, the virtual circuit is considered closed

Sample TCP Connection

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
192.168.0.2	192.168.0.1	0	0	1	38010	4717	0