Internet Security [1]
VU 184.216

Engin Kirda engin@infosys.tuwien.ac.at

Christopher Kruegel chris@auto.tuwien.ac.at

Administration

- Challenge 2
  - deadline is tomorrow
  - 177 correct solutions

- Challenge 4
  - will be issued next week (around 10th May)
  - first "real programming" assignment (Java)
  - simple SMTP engine
  - demonstrates how easily email information can be spoofed

Internet Application Security

Internet Applications

- Traditional services
  - emerged to satisfy needs from the beginning of the Internet
  - often no (or little) security in mind

  - mail transfer (SMTP)
  - name resolution (DNS)
  - file transfer (FTP)
  - remote access (telnet, rservices)

- Secure replacements
  - introduced to address problems in traditional protocols

  - remote access (ssh)
  - file transfer (scp)

SMTP

Simple Mail Transfer Protocol (SMTP)

- initially specified in RFC 821
- de facto standard for email transmission
- simple, text-based protocol
- MIME used to encode binary files (attachments)
- listens on port 25
- push protocol (used to exchange emails between servers)
- clients have to retrieve emails via other protocols such as IMAP or POP

SMTP Session

```
S: 220 www.example.com ESMTP Postfix
C: HELO mydomain.com
S: 250 Hello mydomain.com
C: MAIL FROM: sender@mydomain.com
S: 250 Ok
C: RCPT TO: friend@example.com
S: 250 Ok
C: DATA
S: 354 End data with <CR><LF>.<CR><LF>
C: Subject: test message
C: From: sender@mydomain.com
C: To: friend@example.com
C:
C: Hello,
C: This is a test.
C: Goodbye.
C: .
S: 250 Ok: queued as 12345
C: QUIT
S: 221 Bye
```

SMTP

- Security Issues
  - mail servers have wide distribution base and are publicly accessible
    - software vulnerabilities
    - configuration errors
  - `sendmail`
    - one of the first SMTP implementations (MTAs)
    - long history of vulnerabilities
    - complicated configuration (M4 macro language)
    - e.g., buffer overflow in Sendmail 8.12.9 and before (2003)
  - `postfix`, `qmail`
    - secure replacements
  - no authentication of sender is performed
    - huge problem
    - makes unsolicited email such a problem

SMTP

- Lack of authentication
  - everyone can connect to a SMTP server and transmit a message
  - server cannot check sender identity (besides IP address)

- Mail relay
  - server accepts message that does not *appear* to be either for a local address or from a local sender

- Solutions for authentication
  - SMTH-AUTH
    - access control list with explicit login
    - clients must be aware of SMTP-AUTH
  - POP-before-SMTP
    - logins are simulated by POP request (which require a login)
    - when a client performs a POP request, its IP address is authenticated with the SMTP server for some time (e.g., 30 minutes)

SPAM

- Unsolicited email message
- Gather destination email addresses
  - brute force guessing
  - harvesting (web pages, mailing lists, news groups, …)
  - verified address are more valuable (social engineering, web bug)
- Delivering spam messages
  - own machine (not very smart)
  - other machines
    - open mail relays
    - open proxies
    - web forms
    - zombie nets (compromised machines)

SPAM

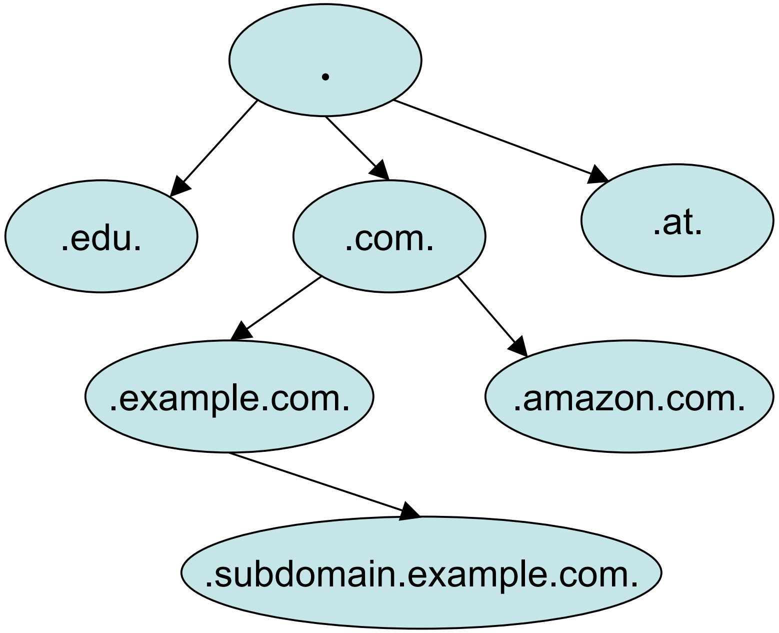
- Countermeasures
  - client
    - filter tools (e.g., SpamAssassin)
    - automatic report systems
  - blacklists
    - identify origins of spam messages and quickly distribute this information
  - infrastructure
    - Sender ID
    - resulted from a merge between SPF (sender policy framework) and Caller-ID
    - works by adding “reverse MX” records for a domain
    - only listed machines can send email from this domain

DNS

Domain Name Service (DNS)

- initially specified in RFC 1034/1035
- distributed database that maps names into IP addresses and vice versa
- name space is hierarchically divided in domains
- each domain is managed by a name server
- clients access name server resolution services through the resolver library
- uses mostly UDP
- sometimes TCP for long queries and TCP for zone transfers between name servers

DNS



Name Server

- Name servers are responsible for mapping names of a domain
  - example
    - subdomain.domain.com is managed by dns.subdomain.domain.com
    - domain.com is managed by master.domain.com

- Root name servers
  - 13 machines distributed around the world
  - associated with the top level of the hierarchy
  - dispatch queries to the appropriate domains

- Server types
  - primary (authorative for the domain, loads data from disk)
  - secondary (backup servers, get data through zone transfers)
  - caching-only (relies on other servers but caches results)
  - forwarding (simply forwards query to other servers)

Name Server

- A server that cannot answer a query forwards the query up in the hierarchy

- Then, the search is following the correct branch in the hierarchy down to the authorative server

- The results are usually maintained in a local cache

- Reverse lookup
  - mapping from IP addresses to names
  - also called pointer queries
  - use dedicated branch in name space starting with ARPA.IN-ADDR
  - example
    - if 128.131.172.79 is resolved, this is mapped into 79.172.131.128.in-addr.arpa

DNS Clients

- At least one name server has to be specified
  - e.g., Linux uses `/etc/resolv.conf`
- Queries can be
  - recursive
    - require a name server to find the answer to the query itself
  - iterative
    - instead of the resolved name another server‘s address is returned, which can be asked
- Lookup can be performed with
  - `nslookup`, `host`, `dig`

DNS Data

- unique message format for requests and replies
- contains questions, answers, authorative information

- DNS data is structured in Resource Records, which store the information.

- Different types of RR exist:

A	defines an IP address for domain name
HINFO	host information (CPU, OS)
NS	authorative name server for domain
MX	mail server for domain

Zone Transfer Info

```
➢  nslookup ...
➢  ls -d infosys.tuwien.ac.at.
[tunamea.tuwien.ac.at]
$ORIGIN infosys.tuwien.ac.at.
@                       1D IN SOA       uhura.kom.tuwien.ac.at.
   hostmaster.noc.tuwien.ac.at. (
                                        1985            ; serial
                                        8H              ; refresh
                                        2H              ; retry
                                        1W              ; expiry
                                        1D )            ; minimum
                        1D IN NS        tunamea.tuwien.ac.at.
                        1D IN NS        tunameb.tuwien.ac.at.
                        1D IN MX        25 nfs1
amd01                   1D IN A         128.131.172.56
amd02                   1D IN A         128.131.172.68
amd03                   1D IN A         128.131.172.69
```

DNS Security Issues

- DNS often provides rich information
  - IP addresses
  - HINFO records
  - WKS
  - can be gathered via exhaustive queries or via zone transfers
  - IP scanning is not necessary in many cases
- DNS hijacking
- Simple DNS spoofing
- DNS cache poisoning
- Daemon vulnerabilities
  - `BIND named` has a bad security history
  - latest problem was a buffer overflow in 2002

DNS Hijacking

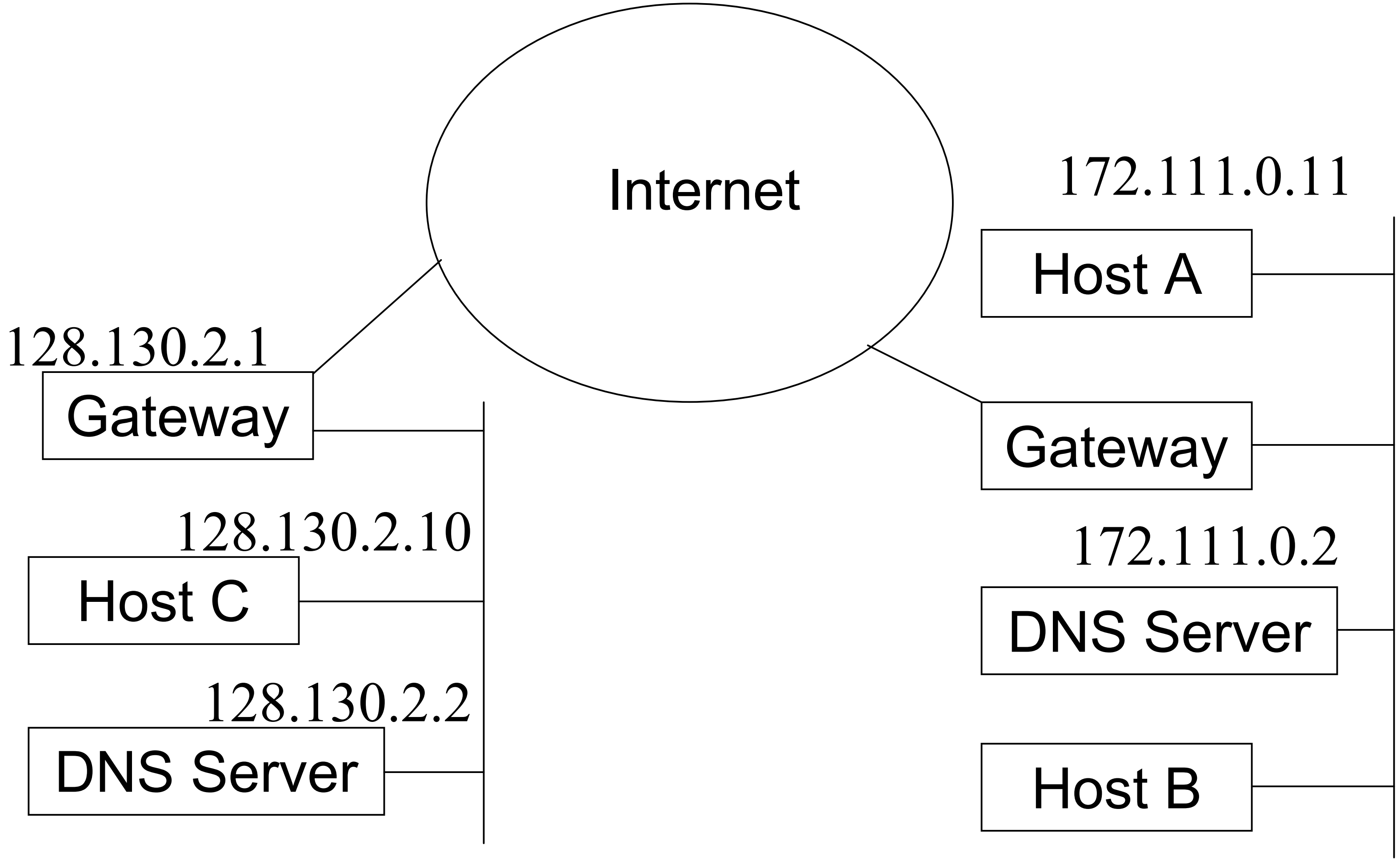
- Relies on the fact the UDP is used
- Usually, attacker has to see DNS requests
- Respond to a request with incorrect data
- Respond faster than legitimate server
- It is possible to perform DNS Hijacking by
  - racing with the server with respect to a client
  - racing with a server with respect to another server
- „Blind“ DNS hijacking
  - requires to guess the request ID
  - many implementations use sequential numbers

Simple DNS Spoofing

- Used when authentication is performed based on DNS names with reverse lookup
  - e.g. trusted.example.com may login using rlogin without specifying a username/password

- Concept
  - a DNS query is forwarded to the authorative DNS server for the IP address that logs in (under control of the attacker)
  - this DNS server replies with the (faked) trusted name

Simple DNS Spoofing



Simple DNS Spoofing

- Host C (128.130.2.10) opens a TCP connection to Host A (172.111.0.11)
- Server A asks its DNS server (172.111.0.2) to look up the address 128.130.2.10
- A‘s DNS server can‘t resolve this address and forwards the query
- C‘s DNS server (128.130.2.3) gets the request and returns a reply with a wrong name (e.g. trusted.example.com)
- A gets from its DNS server the answer that 128.130.2.10 is trusted.example.com and allows C to log in without password

Simple DNS Spoofing

- Countermeasure
  - use double reverse lookup
  - given the IP address i obtain the name n
  - using name n, obtain IP address j
  - check if i=j

DNS Cache Poisoning

- This attack exploits a bug in some implementations of BIND
- A server stores in the cache anything that is contained in a DNS reply
- A malicious DNS server returns additional answers that are stored in the cache (preferably with a long TTL)
- Some implementations will even accept answer records in DNS requests, caching the information
- Attacker can control IP address mappings
- Traffic redirection and man-in-the-middle attacks possible

FTP

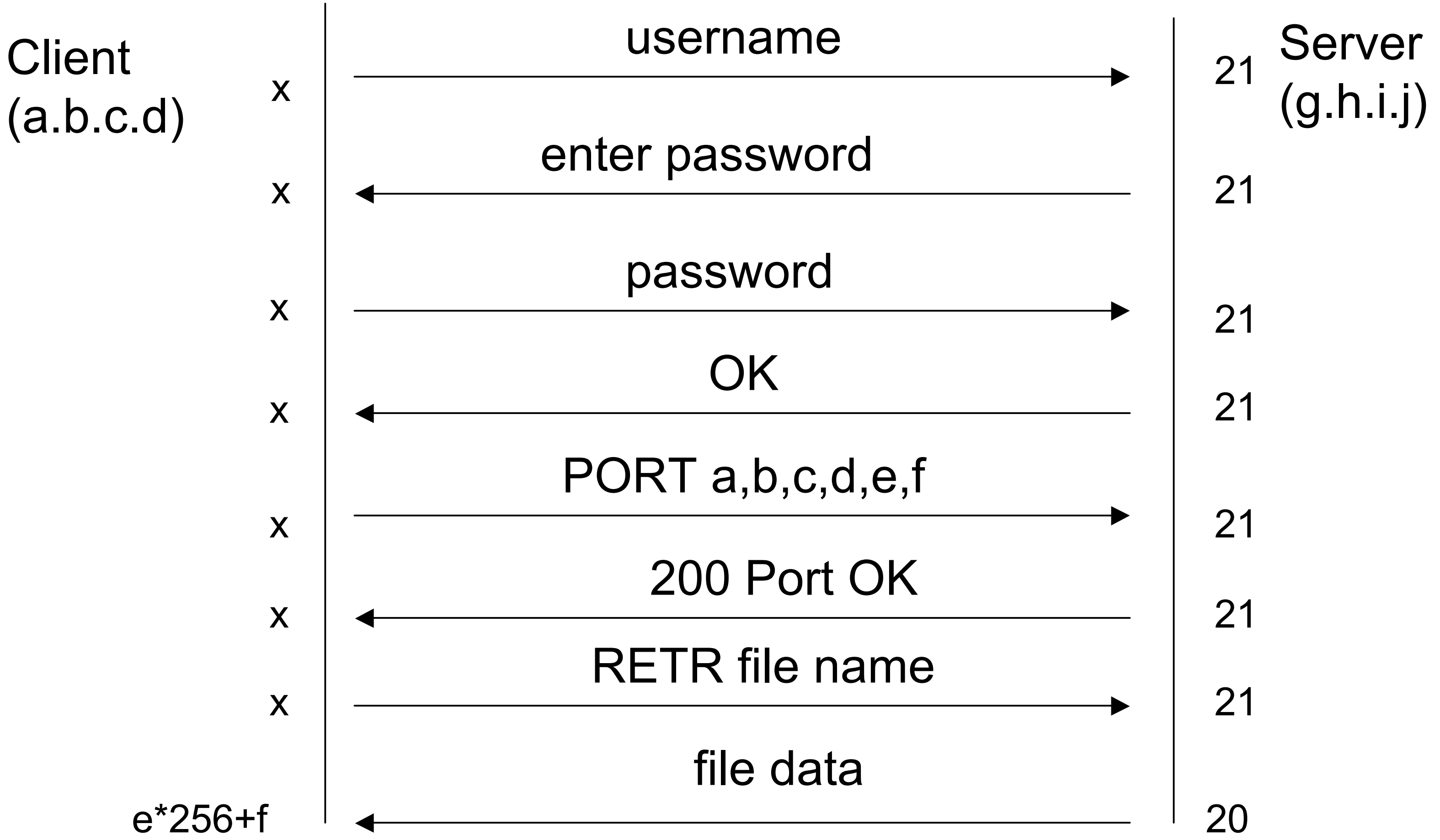
File Transfer Protocol (FTP)

- initially specified in RFC 542
- provides file transfer service
- based on TCP
- client / server architecture
  - client (ftp) sends a connection request to the server (ftpd)
  - server is listening on port 21
  - client provides username and password to authenticate
  - client uses the GET and PUT commands to transfer files

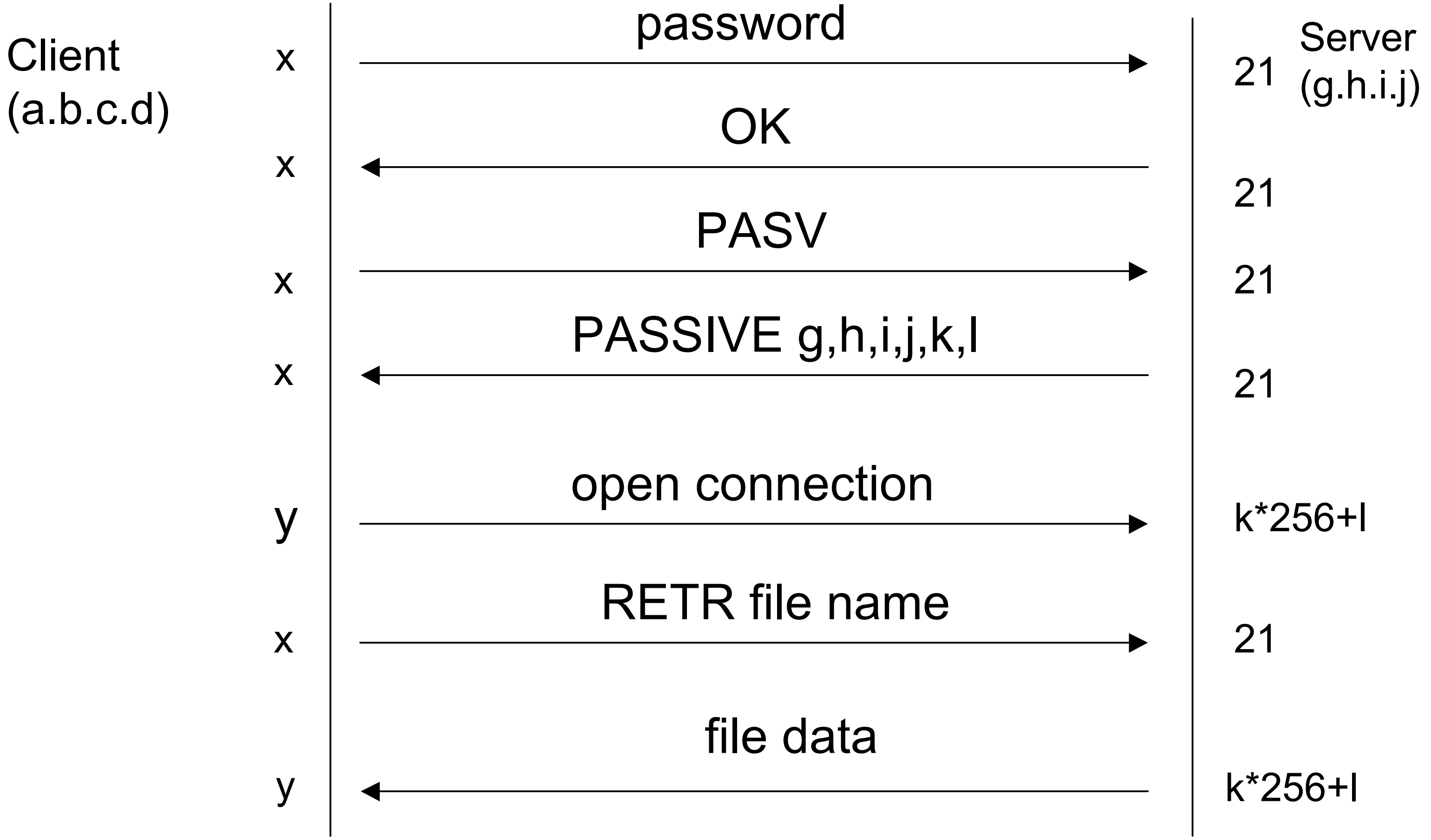
FTP

- Control stream and data streams are used
  - control stream for commands
  - data stream for the actual data that is transmitted
- Client tells the server to connect to one of its local ports using the PORT command
- Server opens a connection from port 20 to the port specified by the client
- Transfer is executed and the connection is closed

FTP Protocol



Passive FTP



Client
(a.b.c.d)

Server
(g.h.i.j)

x password 21

x OK 21

x PASV 21

x PASSIVE g,h,i,j,k,l 21

y open connection k*256+l

x RETR file name 21

y file data k*256+l

Writable FTP Home

- Can be abused to write files into home directories that are normally used for authentication (e.g. rhosts)
- If an anonymous user is allowed to put such a file in the home directory he can get access to the computer, using a file that contains “+ + “
- ftp to a site, put the file dummy in the home directory (as .rhosts) and then

  ```
  rlogin -l ftp target.com
  ftp@target.com:/usr/local/ftp> ls
  ```
- In general, the access of the file system via ftp should be minimized

PASV Connection Theft

- Attacker can connect to port that was opened by server before the legitimate client does
- Since the command connection is still established, client commands lead to file transfers between attacker and server

FTP Bounce

- The PORT command is used by the client to tell the server the address and port to be used when opening a data connection

- According to the RFC 959 the address does not have to be the same as the one the client has
  - idea is to allow transfers between two hosts without having to go through the client

- Therefore it is possible to instruct a server to open a connection to a third host

FTP Bounce

- Can be used to perform a TCP portscan
  - The host running ftpd appears to be the source of the scan
  - It is possible to scan „behind“ a firewall (suppose that only port 21 and 20 are open at the firewall)

- Can be used to send data to arbitrary ports
  - if an FTP writable directory exists, a file can be transferred to a third host
  - can be used to bypass restrictions (IP based authentication)
  - connection laundry

Remote Access

- telnet, rlogin
  - horrible security
    - plaintext passwords
    - connection hijacking (`hunt`)
  - fortunately, it is virtually not used anymore

- ssh
  - secure replacement
  - ssh version 1
    - insecure because of possibility to insert data into remote stream
  - ssh version 2 is current, and should be used

Conclusions

- Traditional Internet applications
  - not built with security in mind
  - some could be easily replaced (telnet, rservices)
  - others cause significant problems
  - SMTP
    - sender authentication
  - DNS
    - simple UDP-based request / reply structure
    - root server bottleneck (denial of service danger)
  - FTP
    - transfer modes using different connections and port combination
    - difficult to firewall
    - connection laundry and bouncing attacks