Email Security

Network Security

Jim Binkley

warning:

- lecture title has large oxymoron potential
- email attachments largest source of security woe?
  - buffer overflow in 2nd place?
- click on me ... leads to perdition

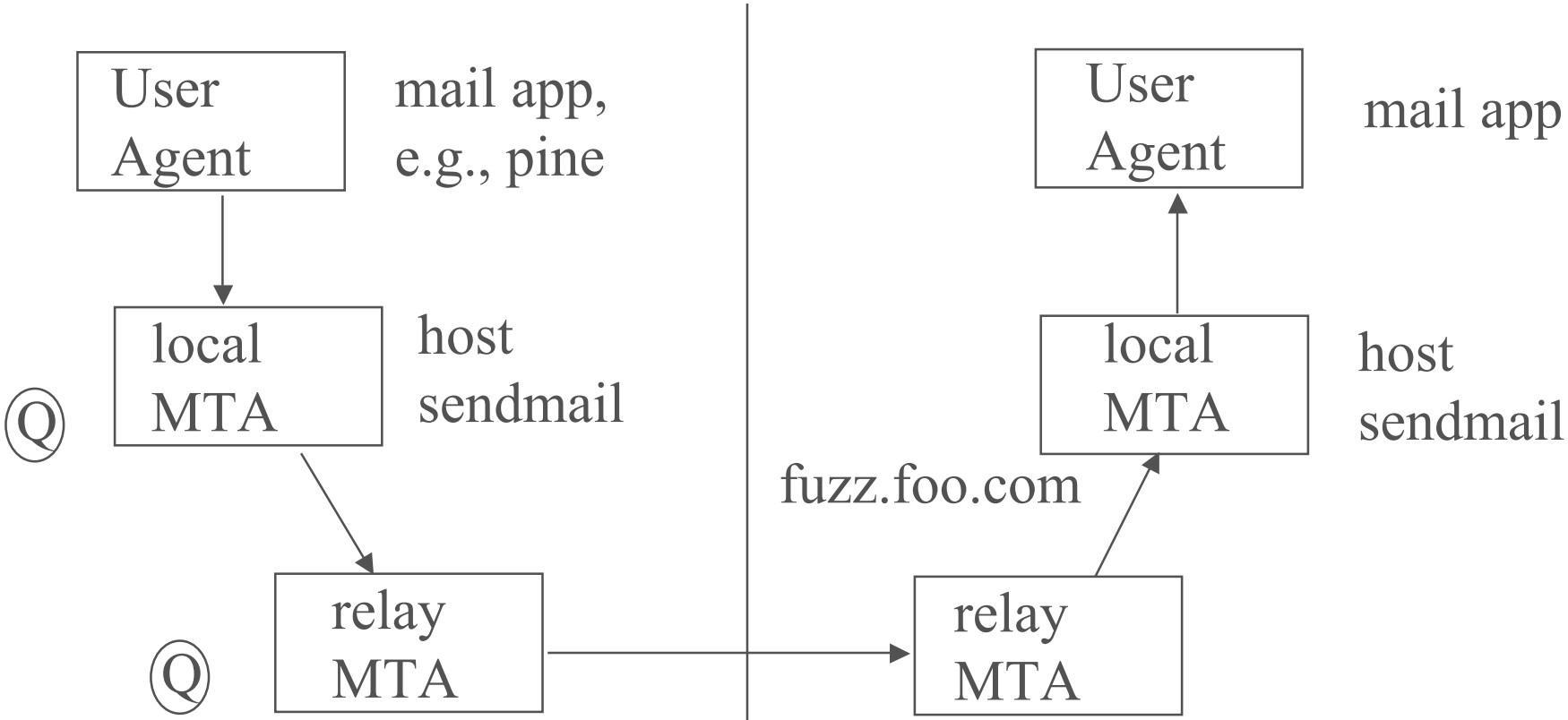
outline

- architecture
- threats
  - and what we can do about those threats
- viruses/hoaxes/trojans/spam
- cryptography and email
- conclusions

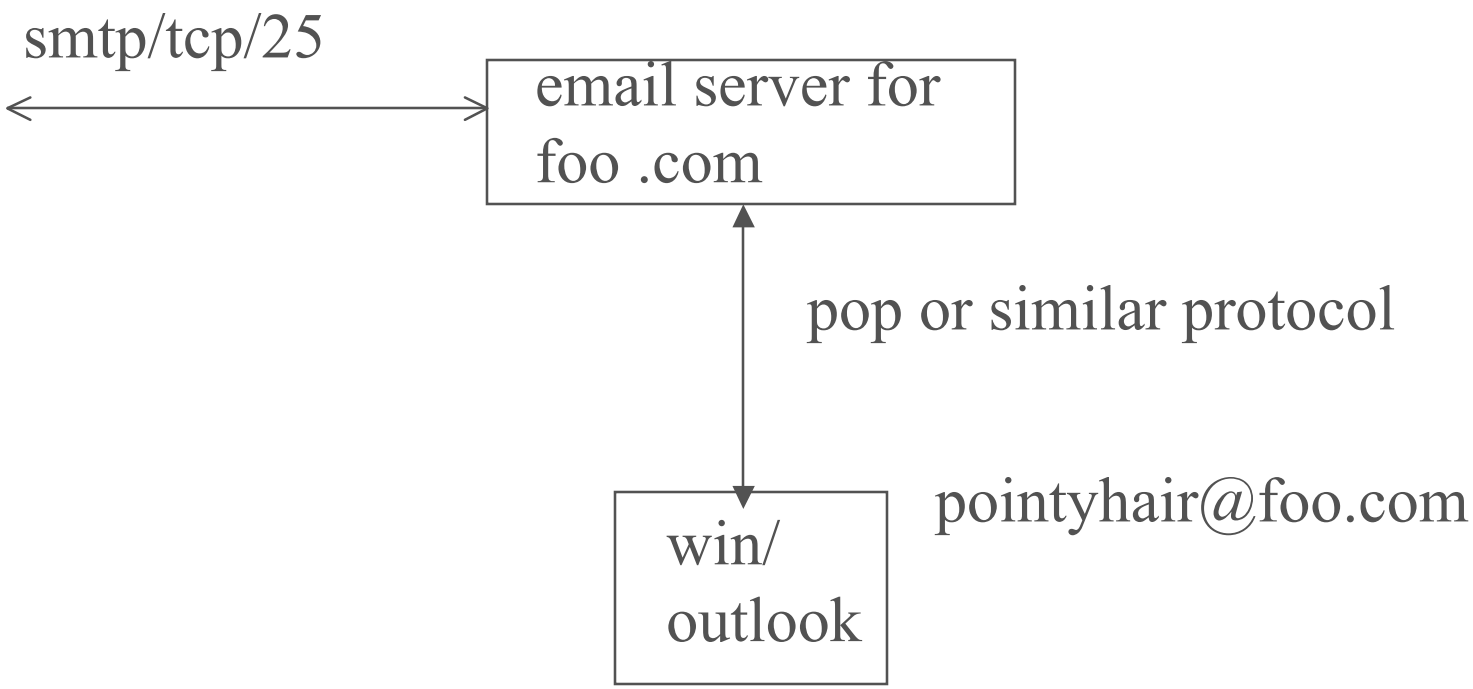
email server architecture

- by definition email servers are L7 gateways
- or put another way: proxy servers
- email sent to company gateway (foo.com)
  - then forwarded to final recipient via:
  - 1. SMTP
  - 2. POP/IMAP
- therefore the following slide is fundamental

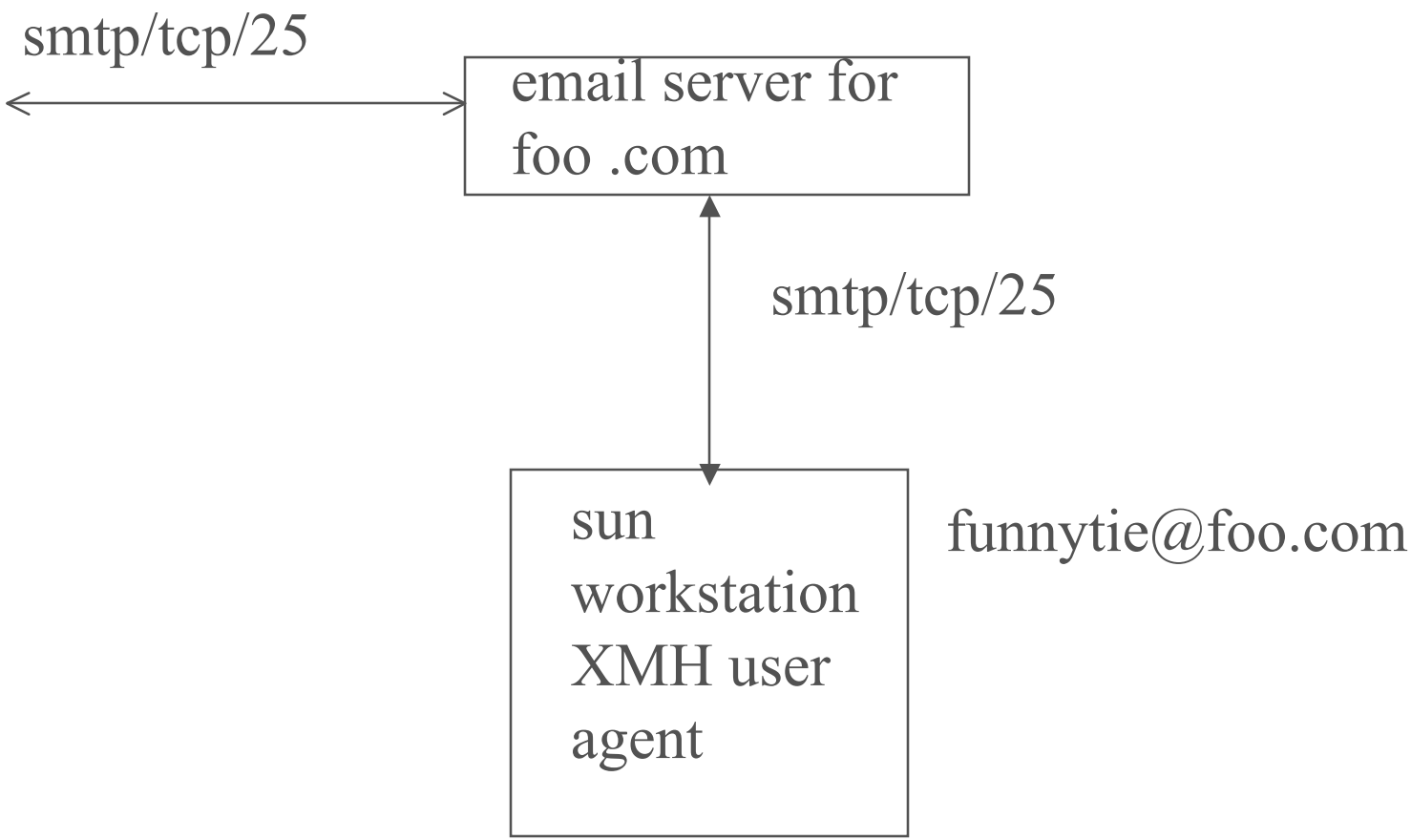
SMTP architecture (generalized)



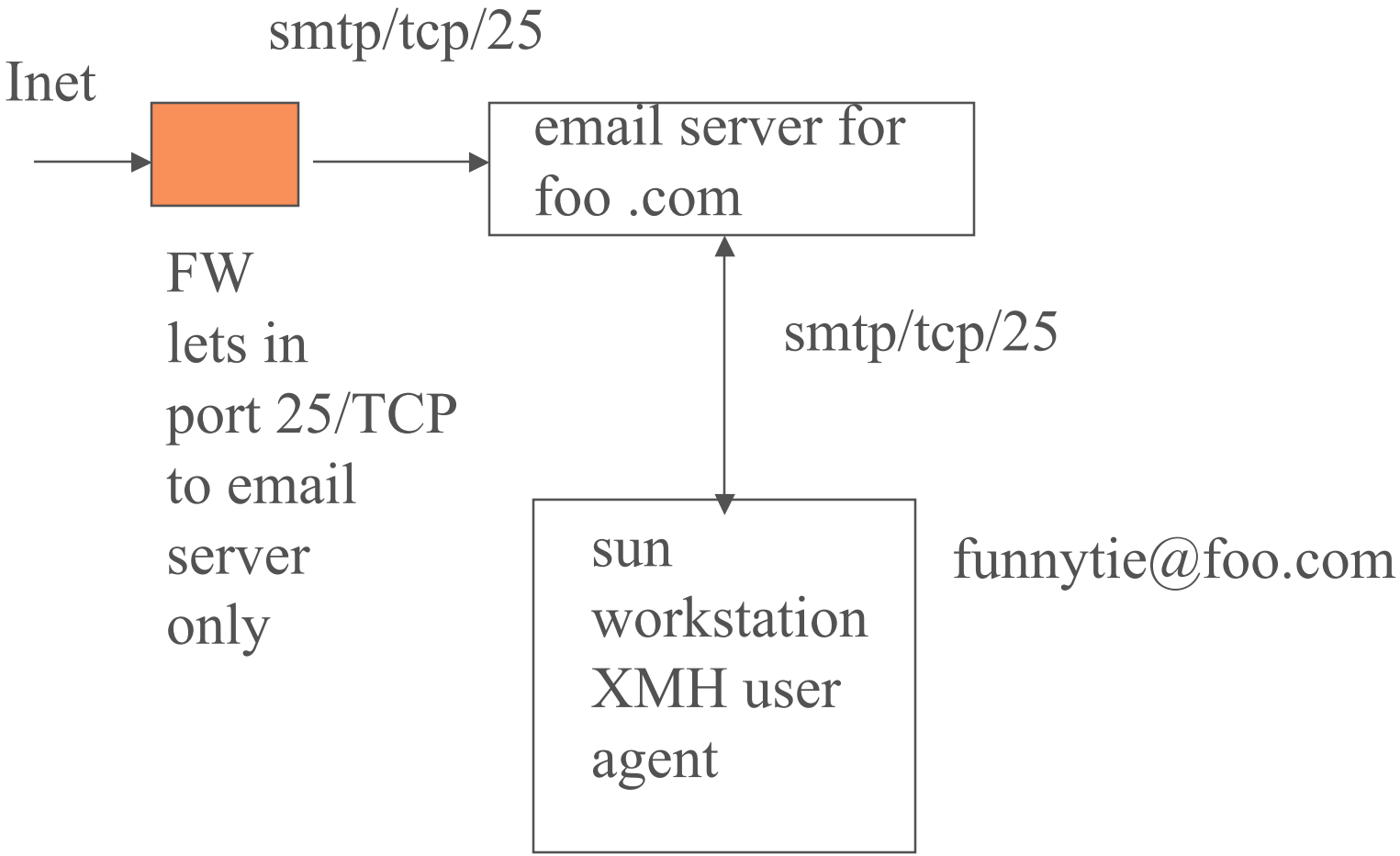
or perhaps like this



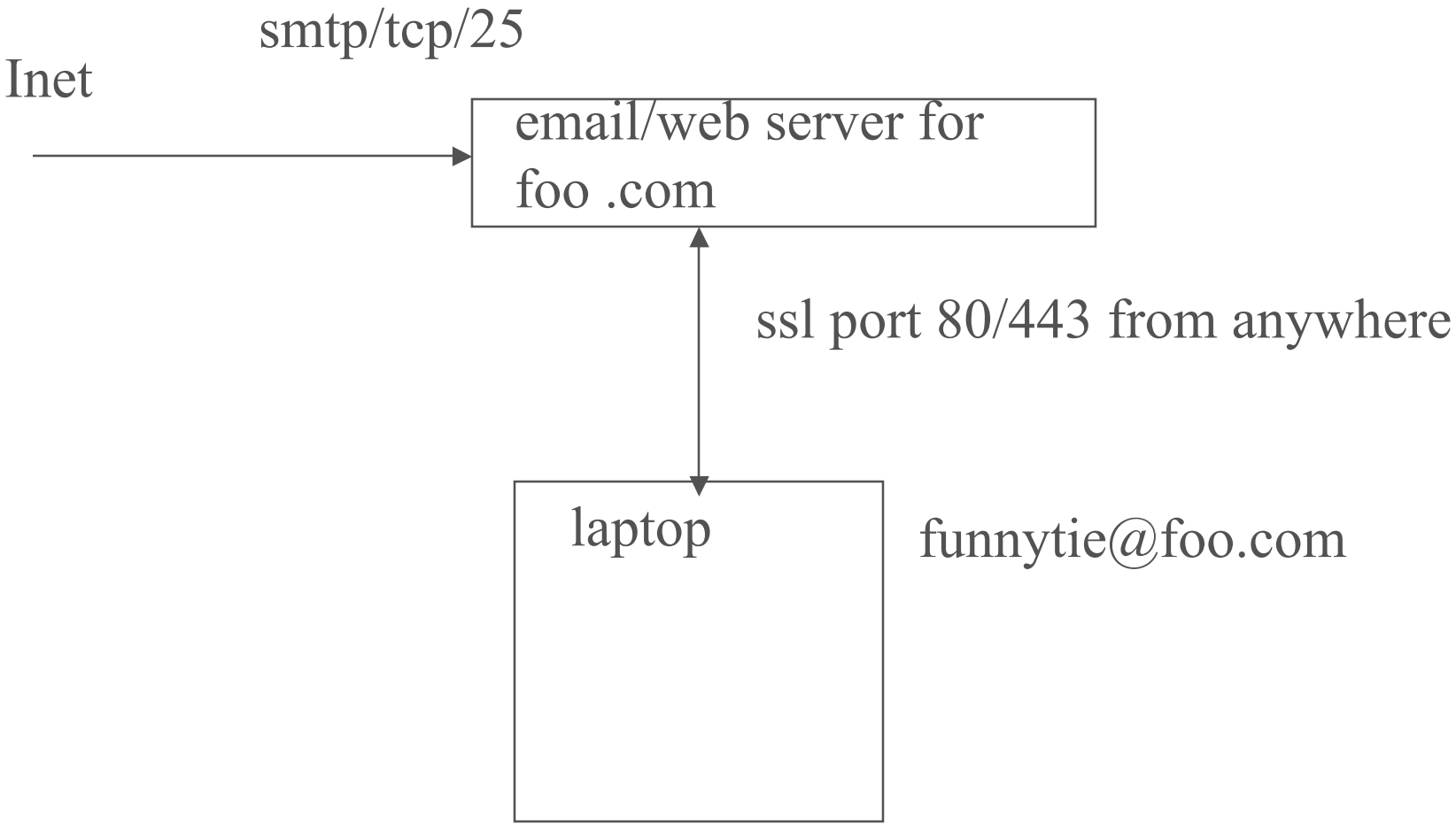
or perhaps like this



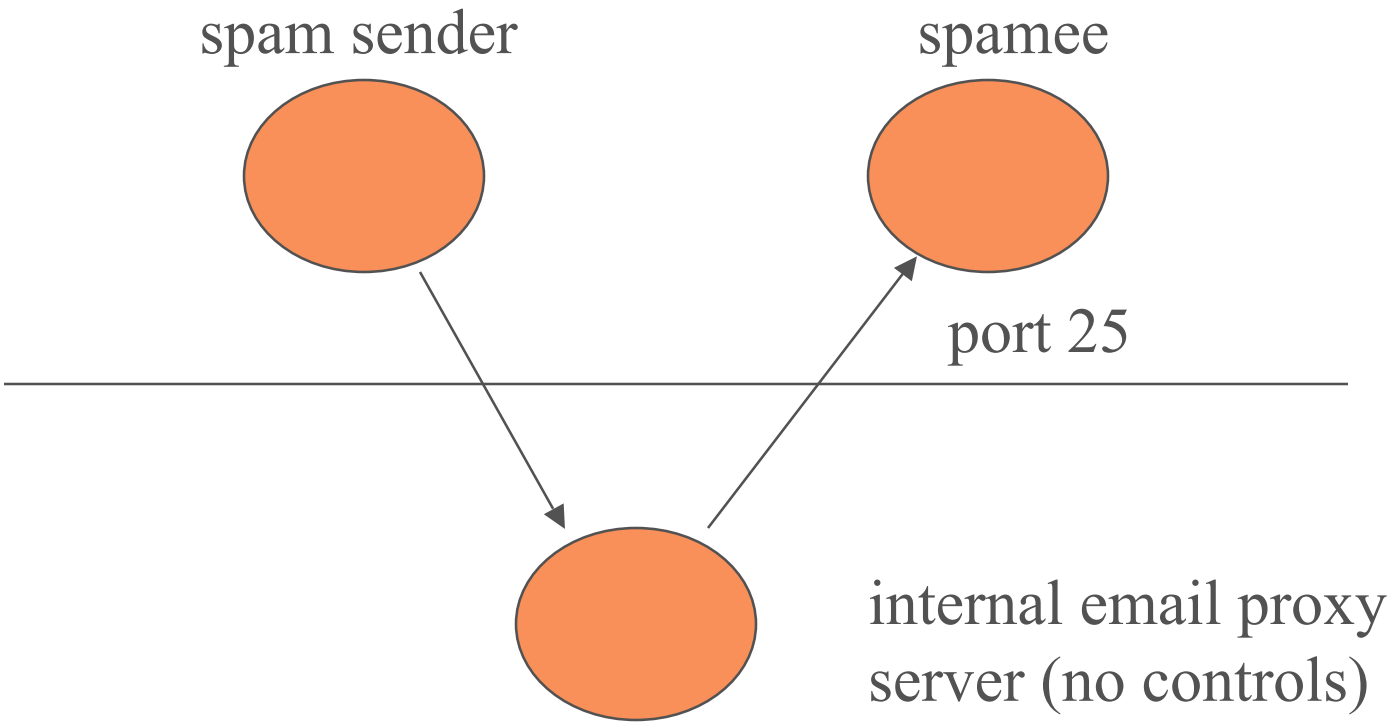
remember: firewall and bastion-host architecture



or web-based (usoft/yahoo/google/webmail)

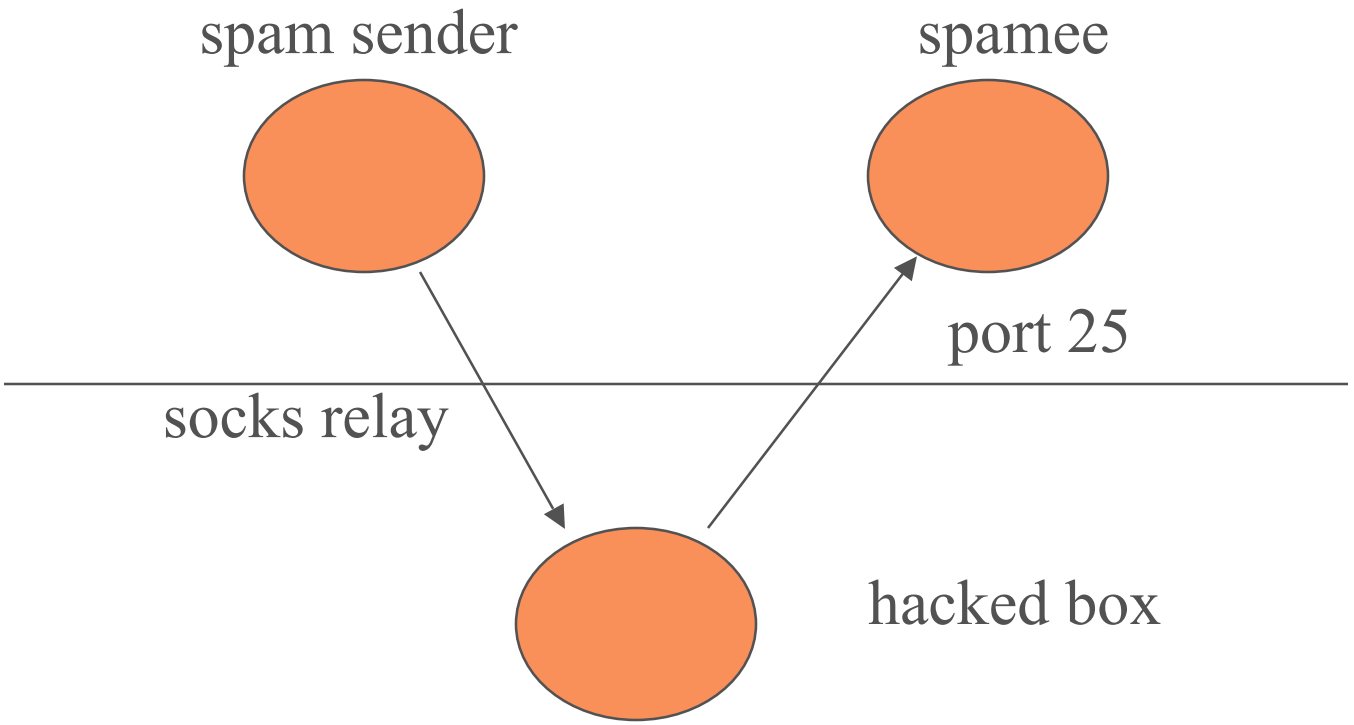


evil variation #1 (old) - email proxy



what are counter-measures?

evil variation #2 (newish) - circuit proxy (web proxy)



what are counter-measures?

re pop and similar protocols

- TCP-based
- username/password
  - password sent in the clear
- file fetching, where files are email of course
  - files are put in “in-box” or in folder
  - or whatever abstraction email client uses
- note pop protocol may be done on Internet (external) or intranet (internal)

pop2/pop3/imap

- pop2, tcp port 109 (outmoded)
- pop3, tcp port 110
- imap (versions 2/4), tcp port 143
- basic idea: host uses TCP
  - ftp-like protocol
  - to get (and send) email thru “local” mail-server
  - smtp used to send email usually

pop3 - RFC 1081, Nov 1988

- commands:
  - USER name
  - PASS string (plaintext)
  - QUIT
  - STAT # of messages for user, plus size of email in bytes
  - LIST [msgid] list of message-ids
  - RETR [msgid] - get a message
  - DELE msg
  - LAST - last msg-id

some evolution over time

- current RFCs
  - RFC 1939, May 1996
  - APOP name digest extension allows the use of a MD5 digest (shared secret)
  - not widely used?
- RFC 2449 talks about how to make pop more extensible
- so what capability are we missing so far?

imap (more complex)

- RFC 3501, U. Washington, March 2003
- operations supported include:
  - remote manipulation of folders on server a la folders on local host
  - create/delete/rename mailboxes
  - check for new messages
  - delete messages
  - possible authentication might include:
    - » TLS-based auth/encryption

MIME - a terrible thing to waste

- so the ever-popular MIME type is used
- for attachments, which could consist of:
  - an executable file (destroy.exe, mybot.exe)
  - a word document (with a word basic virus)
    - » same for powerpoint/excel
  - an interpreted file of some other kind
    - » pdf/ps
  - a picture/song/movie/ASCII text

what to expect of MIME?

- it is true that in general attachments are NOT directly executed upon receipt (anymore)
  - you should have to do it yourself
  - **know the defaults of your UA**
- nor should they be executed by simply looking at the email itself
  - **know the defaults of your UA**
- but “execution” of attachments is in general a bad idea (word on foo.doc is a bad idea)

smtp protocol aspects

- envelope has TCP connection
  - ip src, ip dst: these are not spoofable, why?
  - MTA log information can be useful here for admins
- email header has:
  - to: bob@dns (ip)
  - from: alice@foo.com (this is spoofable)
- may have distribution-list for recipient
  - or mail-list
  - 1-n expansion
- distribution-list explosion may be at gateway or sender User Agent

email header

- added by some combination of MTA/UA
- useful fields often suppressed by UAs
  - not all though
- **From**: possibly added by MTA. spoofable
- **Received**: usually added by MTA
  - multiple MTA additions common
  - added at the top (newer at the top)
  - at some point, not spoofable
  - this is what MTA uses to count for loop detection

email header

- **Date**: possibly added by MTA, but spoofable as UA can do it
- **To**: can be suppressed
- **Message-Id**: MTA should uniquely id sender
- **X-*:** custom fields added for UA or for documentation sometimes
- **Subject**: optional

email may have infinite loops

- A has .forward that says
  - B@foo.com
- B has .forward that says
  - A@bar.com
- email servers must detect this and delete messages
- mailing lists can have infinite loops too

the threats

- click on me for a:
  - trojan horse: (BO and friends)
    - » your host just became a porn-server
  - worm/virus like melissa/sql-slammer
    - » melissa goes thru your “address book” and forwards itself to the address book recipients
    - » sql-slammer immediately starts UDP thrashing of networking to forward itself
  - worm/virus like blaster
    - » tcp syn attack on usoft/SCO or whomever?
    - » what if they sue?

click on this ...

- click on me cont:
  - you just became an email proxy server for Nigerian spam to be sent elsewhere
  - you just installed a virus that will delete some or all of your files
  - you just installed welchia/nachi that is going to start doing ICMP scans of local/remote nets
  - you just installed a word document virus that will infect word docs that you send yourself

note social engineering potential available in subject line

- hey cutie, for a good time “click on me”
- “you just won 1 million dollars”
- “if you don’t help, 5 million dollars will go to waste”
- “hi from grandma”
  - it isn’t grandma
  - or it is grandma, but she sent you a virus
    - » hmmm....
- and things we haven’t thought of yet ...

more threats

- open email server (proxy server)
  - by accident
  - because of malicious intent
    - » malware installed it
    - » malware turned it on
- so 3rd parties can send email thru your site and possibly have it appear to be from you
- spam can cause blackholing in email land or worse (foo.com won't talk to you anymore)

pop password threat/sniffing

- somebody can read your password and spoof you
  - due to sniffer in “wrong” network location
- or simply read private email that doesn’t belong to them anyway via either SMTP or pop-like protocols
  - smtp/pop are plaintext protocols
  - data must be ASCII

spam threat

- amount of spam just keeps rising
- spam filtering is not perfect
  - and can make serious mistakes due to admin goofs
  - or because the algorithm/s are not smart enough (a la web filtering for kids)
- some spam is legitimate business
  - which does NOT mean that I want to get it
  - **some is criminal fraud** and some people fall for it

identity threats

- virus A on user box B (you are Z)
  - address book has Z@reallycool.edu
  - or web page from Z that has Z's email address in it in web cache
- Z now receives email from location X
  - hey Z, you tried to send email to Y@X that had a virus in it
- but Z uses MH mail on a unix system ...

buffer exploits on email server software

- sendmail has a spotty track record
- buffer-exploits and other bugs have lead to
- successful root exploits
  - loss of box ... which doesn't necessarily have anything to do with email/threat ironically

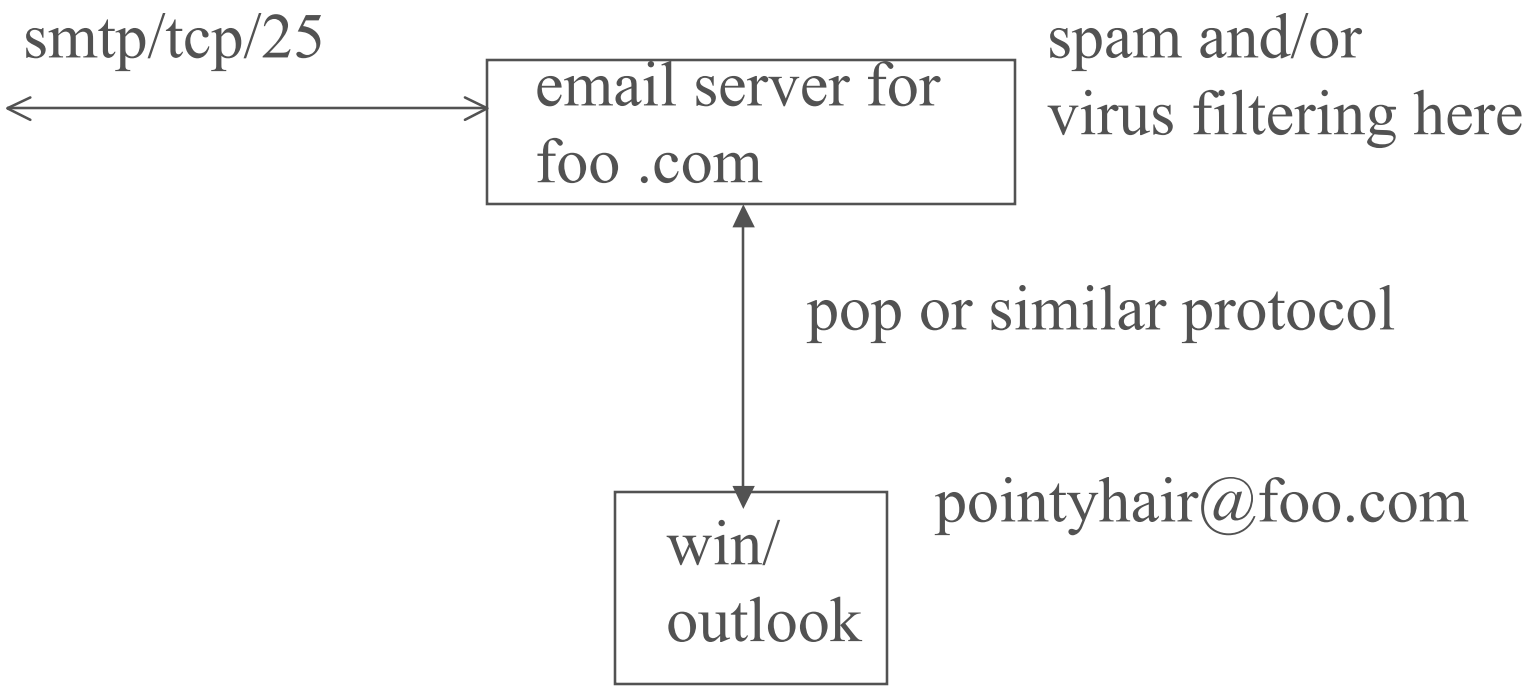
solutions:

- save attachments in a file
  - and run a virus checker on them
  - if you really really think you should get the file
  - unfortunately: you may have been the 1st person on the block to receive the new virus for which there is as of yet no signature
- have a virus checker and keep it up to date
- never or seldom accept attachments
  - which is nearly impossible

local admins MAY filter for you

- so local email server
  - runs spam filter
    - » spamassassin in CECS
  - runs virus filter
    - » just snip off those attachments in toto
    - » or clip off the ones with known worms/viruses
    - » signatured-based system here

email gateway filter



read your email on unix

- .exe isn't going to go anywhere
- feed your .doc file to star office or open office
- don't do attachments in email client
  - GNU uudeview app can take files out of email
  - attachments are just *files*
- some consideration has been given to notion of a "safe-house" or bomb-proof box

solutions for virus/spam detection

- 1. can be host-based
  - plenty of commercial possibilities
- 2. can be gateway-based
- 3. open-source systems?
  - clamav - clamav.elektrapro.com
    - » virus database and src on sourceforge
  - spamassassin - eu.spamassassin.org
    - » or see spamassassin.org

note existence of blacklist mechanisms

- site chooses to not accept email from you
- because you are listed on some other site or in some database as a spammer
- for example, see:
  - www.mail-abuse.org
  - ordb.org (open relay database)
- razor.sourceforge.net
  - collaborative spam-tracking database
- is shooting the victim a good idea?

some apps have a worse track record than others

- bad app list includes:
  - outlook
  - sendmail as MTA (buffer overflows and other problems, leading to successful root exploits)
  - pine/imapd have had problems
  - not just windows ...
- so: use something other than outlook on windows
  - eudora/web browser email client
- unix: use something other than sendmail as MTA
  - smail/qmail others I know little about

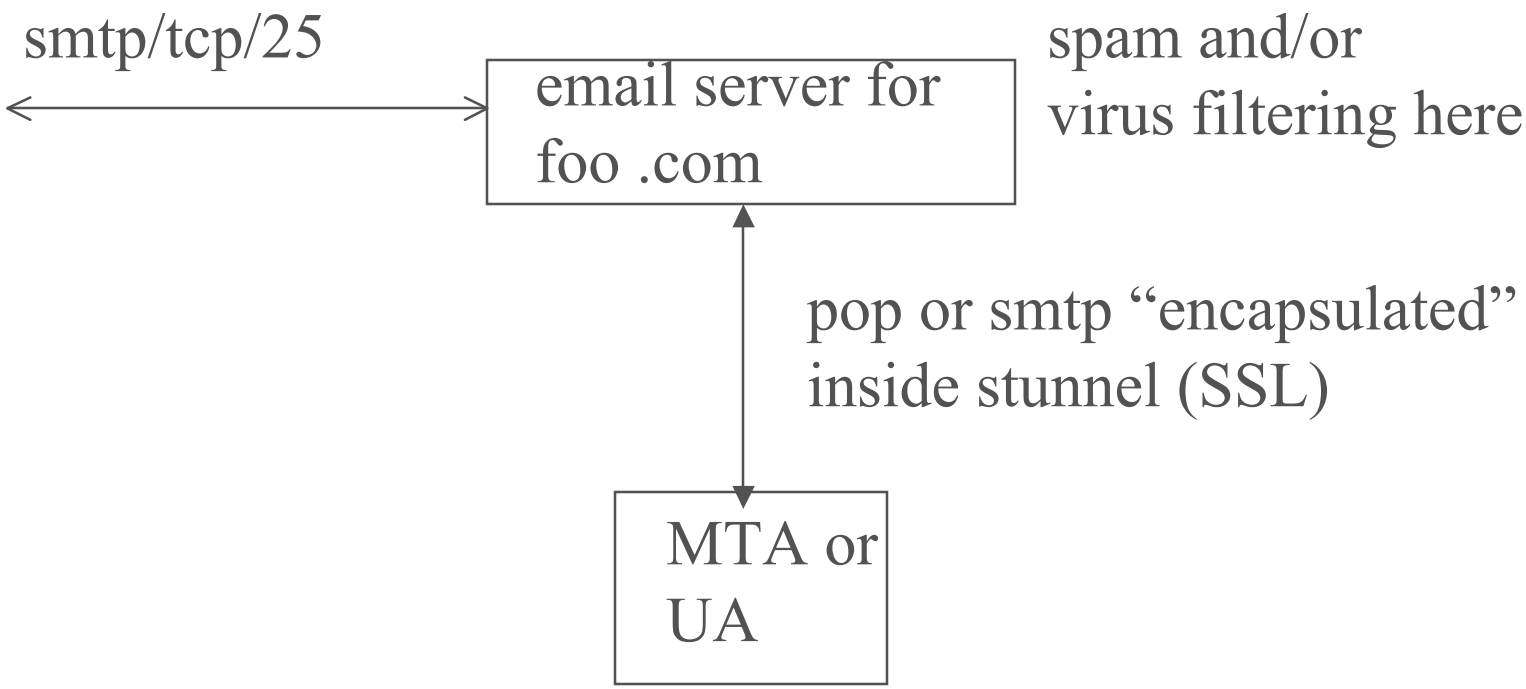
what could you do to?

- make sure your windows system is NOT executing a worm/virus right now?
  - run a virus checker
  - use a netstat -a like app to see what ports you have open, and then periodically check for changes (you did that before you read email?)
  - run nmap from some other box to get the same information
  - ps would be nice ...

what role can crypto play in any email threat counter-measures?

- may be of use to protect email from MTA to UA
  - to prevent prying eyes looking at content
  - or seeing pop password
- may be use between UA/UA when content is secret
- **doesn't help us with viruses though**
  - hey it really is grandma and here is a nice virus for you ...

encrypted/email gateway filter



what is the trust model?

- for the previous slide
- using ssl ...
- how does this differ from the https://foo.com web transaction
  - where you just purchased a widget from foo.com
  - and sent them your visa number?

viruses/trojans/hoaxes/spam

- usual virus definition (F. Cohen):
  "a program that replicates by 'infecting' other programs so that they contain a (possibly-evolved) copy of the virus"
- emphasis is on: replication
- not: damage, mayhem, and destruction
- maybe a virus does good? is this likely?

how many viruses are there?

- **nobody knows**
- wildlist states there are a few hundred “in the wild”
  - http://www.wildlist.org
- some vendors state 60000 ...
- viruses have variations ...

virus piggyback possibilities include:

- floppy or harddisk boot sector
- media like floppy or cdrom (probably in a file)
- attached to an attachment (a file)
  - executable, or even an image file
- as a visual basic program in a word .doc
  - so-called **macro virus** (macro and doc in same file)
  - word and excel both have had them
- multipartite viruses (come back to this)
- scripting virus (come back to this)

virus might also

- infect memory but not store itself in a file
  - sql/slammer infected memory
  - would go away on reboot
  - however suspend of course wouldn't eliminate it
- might infect memory anyway from a file
  - so that it can periodically make trouble
- windows W32/Perrun virus
  - infects jpeg files, and makes them executable

ok, so what's a worm then?

- F. Cohen regards worms as a subset of virus
- some say: a worm is a program that copies itself
- a virus does NOT copy itself, merely goes for a ride
- we certainly have malware that does this:
  - click on it to activate it
  - then it acts as a worm to propagate itself (welchia)
  - or it sends more email for the next "click on me" cycle
  - so worm/virus is not an unfair term

virus activity along these lines:

- user executes a program (or boots ...)
  - note that one may have programs on windows installed to auto-run at boot
  - possibly the trojan runs at this point
  - UNIX system boot might start something out of /etc/initd or /etc/rc scripts
  - UNIX user (especially root) might have bomb in .login/.cshrc (time for a story)

virus overview, continued:

- virus code is SOMEHOW executed
  - instead of before the legitimate program
- virus code may terminate and hand control off to legitimate program
  - or run in background
- viruses often have bugs
  - and sometimes the virus bugs are more dangerous than the virus
  - commercial/open-source code has some pressure to remove the bugs. **virus writers do not seek bug reports**

virus components

- 1. infective routine
  - which should check to make sure that it doesn't reinfect the target over and over
- 2. a payload - possibly some annoying action that the virus takes
  - plays music or deletes a file or eliminates itself
- 3. a trigger - some event that triggers payload delivery
- trigger + payload == **logic bomb**

virus algorithm

- look for infectable objects
  - if any found, infect them
  - else
    exit (or wait a while and try again)
  - if trigger exists (next slide)
    deliver payload
- so virus may take direct action or be memory-resident

boot-sector infectors

- mostly dependent on DOS floppy disks being handed back/from
- their day may be past
  - especially if you do NOT exchange disks
- non-trivial in terms of system understanding
  - probably written in assembler for one thing
- if hard-disk infected, common for virus to infect any floppies inserted

file viruses (parasitic)

- worms here are probably most successful of this breed
- question: just how many files are infected when virus is executed?
  - all .exe files?
  - just the ones in this directory?
  - only win.exe ?
  - or some common .dll file?

more on file viruses

- .com, .exe, dll, vxd, screensaver (.scr)
- font files
- .pif (program info file), .bat, .lnk
  - pif file used to store info about dos programs executed under windows
- in theory, extensions mean something on windows
- and mean nothing on unix

virus types continued

- **multipartite virus**: a virus that uses more than one way to get executed
  - boot sector and file both infected
- **multipolar virus**: malware that contains more than one threat:
  - super-worm that uses Usoft dcom vulnerability, checks out sql bug, and includes BO as a side-dish

macro virus

- Microsoft Office apps are the target
- historically gave us first multi-platform virus
  - here is a .doc file, and you can infect your:
    - » 1. DOS box
    - » 2. apple box
- visual basic for applications
- macro language cannot be easily unbound from app's own command facility
- can infect global template, modify commands, initiate, etc.

virus types, continued

- script virus: fuzzy distinction between macro virus and script virus
- e.g., some script written in VB script
  - can be embedded in html scripts
  - executed by html-aware email clients thru Windows Scripting Host facility
- VBscript and Jscript seem more friendly to viruses than javascript
- UNIX shellscript always possible

one last type:

- **memetic virus**: meme is unit of cultural transmission
  - a gene of culture ...
- this simply means: “a virus of the mind”
- these are simply hoaxes about viruses in the strict sense
  - and in the loose sense, email like “chain letters” or bad jokes ...

good times virus (doesn't exist)

- good times virus: famous example of memetic virus
- email arrives that claims that a good times virus may arrive real soon now
- may delete your hard disk files, cause your CPU to catch on fire, or make your mouse leap out the window
- a "hoax" could be real: "quick, delete
- be aware that hoaxes do exist, but you still should probably check with local IT, or virus sites

good point re virus containment:

- let's say you get a modern commercial virus checker system for windows
- and it auto-updates its signatures everytime you login
- a so-called "flash worm" (like the sql-slammer) can cross the Inet in 5 minutes
- on the other hand a virus/worm that rides on the back of email takes time
- so: what are pros/cons of auto signature update?

characteristics of viruses

- stealth - virus attempts to conceal its presence
  - if payload is HIGHLY noticeable does tend to be a giveaway, huh?
  - there are 2 kinds of tools for detecting viruses:
  - 1. anomaly detectors (something changed)
  - 2. signature-based detection (pattern X was found in file Y, or memory location Z)
  - stealth virus may present a new form of anomaly ...

characteristics, cont.

- polymorphism: polymorphic viruses attempt to change their "body" when they infect
- goal: defect signature analysis
- examples:
  - change order of instructions
  - introduce noise bytes (nops)
  - or use encryption

antivirus utilities

- functions may include:
- 1. integrity checking (checksum-based)
- 2. behavior monitor (establish baseline and watch for deviation)
- 3. may look for signatures in various ways
  - including database of signatures
- 4. or for back-doors, dos and ddos malware as well
- 5. may simply check for garbage files
- 6. look for so-called “spyware”

what can virus detector do?

- tell you that you have a problem
- possibly cleanup the damage
  - fix boot-sector
  - delete macro virus
  - delete file? or part of file
- system file deletion is risky
  - backups are important and must be part of the process
- windows registry mod is risky

some anti-virus vendors

- avg anti-virus: www.grisoft.com
  - free home version
- Network Associates
  - www.nai.com
- Norton
  - www.symantec.com
- F-prot anti-virus
  - www.complex.is and/or www.f-secure.com

some rules:

- 1. check on hoaxes, they could be true BUT
  - don't forward it ...
- 2. don't trust attachments
  - even if they come from somebody you know
  - you could ask person X (over the telephone) if they sent you an attachment
- 3. re virus detection software
  - keep it up to date
  - remember there could always be a new virus that they haven't dealt with as of yet
  - however, in general the vendors are fast

more rules

- if you are an admin, think twice about turning on this “feature”
  - automatically inform sender X that they sent you a virus
  - remember *Melissa*
- try not to install random software on your box
- turn off auto-execution of macros
  - maybe they can send you .pdf, .ps, .rtf ?

more rules

- patch it until you bleed
- back it up (see previous rule)

trojans

- **trojan horse: a program that does something unexpected**
- in virus terms, the payload does the unexpected thing
- this definition is very ambiguous
  - could apply to all buggy programs ...
  - does it apply to all Microsoft software then?
- usually we mean it does something bad ...
- it may do something "good" or at least innocuous as a stealth technique

trojans, cont.

- some suggest that a trojan is not a virus
- **because it cannot replicate**
- others disagree ...
- trojan might:
  - 1. try to gain unauthorized access
  - 2. deny service
  - 3. modify or destroy data with authorization
- **social engineering often important**

trojans, cont.

- social engineering is often important part
  - "but the giant horse statue on wheels was really beautiful ..."
- some therefore define a trojan as:
  - a **worm** (or virus) with a high degree of social engineering
  - "click on me cutie!" is therefore a trojan/virus/worm thingee
- so just what does trojan mean?

trojans, cont.

- so is a rootkit kind of a giant mega-trojan?
- See Dave Dittrich's rootkit faq:
- http://staff.washington.edu/dittrich/misc/faqs/lrk4.faq
- note that windows and unix both have had root kits "published" in the hacker community

destructive trojans

- common for trojan to do its damage at once
- might even simply exec del/deltree/format
- pkzip “trojan” deleted files
  - trojan didn’t bother to act like pkzip
  - possible that worry over it was worse than actual impact
- chernobyl virus: attempted to overwrite the system BIOS and erase hard drive

privacy-invasion trojans

- passwords are a common target
- old unix hack:
  - put login up on serial console
  - save passwords in a file/email to somewhere
  - login attempt may succeed or fail

back door trojans

- Ken Thompson and his trojanized C compiler
- just what is a back door anyway?
  - Morris Worm: sendmail DEBUG is example
- this term is also used for remote access systems like back orifice, netbus, etc.

spam

- spam is basically just like a weed:
- weed: a plant you don't want
- spam: email you don't want
  - attempt to sell you something
  - may attempt to steal from you though
    - » identity theft as a side effect, steal visa card info
    - » bank account info, kidnap you for ransom
  - email addresses gleaned from the web, USENET news, and lists sold by spammers

what can be done about spam?

- blacklist spammers
- prevent open-relays
- auto-detect spam at the gateway and delete it
  - but spammers are fighting back by inserting lots of "invisible" words in html
  - OR AVOIDING UPPERCASE!!!
- or via legislation?
  - "hey spammer, please put ADV in your subject line"
- or suggestions for charging for email?
- any ideas?

encryption and email

- terminology and basic ideas
- pem
- s/mime
- pgp

security services for email

- privacy - 3rd party can't see your content
- authentication - Bob knows it came from Alice
- integrity - Bob knows the content didn't change
- non-repudiation - recipient can prove that sender sent the mail (sender can't deny it)
- proof of submission - sender knows that mail was indeed put into the system
- proof of delivery - sender knows that recipient got it.

a few more from the KPS book

- message flow confidentiality - third party cannot even know that you sent a message
- anonymity - recipient can't tell who the sender is
- containment - network can keep security levels of messages from leaking out to certain regions
- how many of these principles exist in the real world of SMTP email?
  - common/uncommon/maybe in military circles?

key distribution basics

- depends on public-key or private key
- as well as
  - alice to bob (1/1)
  - alice to alice-fan-club (1/N)
  - funnytie-the-admin to alice (email gateway to UA)
    - » pop can be put in an encryption wrapper
    - » MTA to MTA can be put in an encryption wrapper

ways to distribute public keys

- Alice and Bob exchange public keys out of band
  - brief-case man or IETF floppy/pgp party
- Alice gets Bob's key from "some kinda" key infrastructure
  - PKI - public-key instrastructure
  - it might exist locally
- Alice sends public-keys in her email signed by her (Bob has to have her public-key though)

ways to distribute private keys

- out of band
  - brief-case man
  - telephone conversation
  - of course it doesn't scale
- Alice and Bob get tickets from a KDC
  - this scales to an enterprise but so far has not scaled beyond an enterprise

privacy/threats

- sniffer may see your email in plaintext
- email gateway admin may read your email
  - or have been compromised by a black-hat
  - or FBI may want to read it to find terrorists
- end to end encryption is a reasonable goal
  - as end to end encryption is always better than any intermediate measure (say gateway to UA)

privacy, really

- even if it is public-key based:
- 1. we generate a symmetric session key and use it because we want to minimize exposure of the long-term key
- 2. we use symmetric encryption because it is faster than asymmetric encryption

logical steps as follows:

- alice generates a random number N
- alice uses N as a symmetric key and encrypts the msg:
  (msg(cybercrud), K(s))
- K(s) is encrypted with Bob's public key
- Alice then sends (msg(cc), (encrypted K(s))
- possible algorithms include: AES, and RSA

authentication of the source

- spoofing can happen easily
- and in point of fact IS HAPPENING A LOT these days ...
- alice can digitally sign the message
  - OR SEND A CHAIN OF CERTIFICATES
- bob can verify with alice's public key
- note that message here can just be:
  - ASCII message (signature cybercrud)
- recipient may NOT have sender's public key (may not care)

certificate chain

- Alice signs her email
  - and includes her public key signed by goodbart the admin (cert), cert for goodbart-the-admin
  - which is signed by uberbart-the-admin
- Bob already has uberbart-the-admin cert
- therefore can verify goodbart/alice

in the real-world what cons exist

- for the notion of using public-key crypto
- to sign messages
- can all messages be signed?
- what if all messages were signed?
- would a system that uses a “callback” help here:
  - A sends B email. B sends email back to A to see if A sent the message?

how to do source authentication with public-key crypto:

- use message-digest algorithm to produce hash for message: (msg, hash)
- Bob knows what md algorithm is used (say HMAC-SHA)
- Alice signs hash not msg with her private key: (msg, hash, signature-cybercrud)
- remember: **email is ASCII so cybercrud must be ASCII too** (even if still cybercrud)

now let's do it with private keys

- alice can prove to bob that they both know the same key
- call this MIC - message integrity code or
- call this MAC - message authentication code
- value also serves as integrity checker
- various ways to compute this

MIC/MAC example:

- take MD of msg == hash (128 bits say)
- encrypt hash with secret key
- send {msg, encrypted hash}

integrity problem

- Juliet sends Romeo this message:
- “forget me not!”
- Juliet’s father intercepts it and changes it to
- “forget me now!” (one letter change ...)
- if we authenticate the message, we should also make sure it does not change
- either due to malice, or accident
- secure mail schemes due both or neither

non-repudiation

- to repudiate means to deny you sent the message
- government might want the opposite
  - U.S. president can deny his leaked invasion plan for France that he sent to the newspapers
  - call this **plausible deniability**
- with public keys, non-repudiation is easy, hard to provide repudiation for src auth.
- private keys are the opposite

public-keys

- non-repudiation, Alice signed it with her private key
- Bob verified it, therefore it is Alice as
- long as Alice has her own private key
- she could claim that Evil Bart stole her computer and took it her private key ...
- but wait Alice, your authentication system uses all 3 auth. schemes ... (you know/are/have)

plausible deniability/public key

- Alice picks a secret key S
- encrypts S with Bob's public key {S}bob.
- signs {S}bob, with her private key.
- uses S to compute a MAC for message m.
  - use DES to compute CBC residue of m
- sends the MAC, signed S, and M to Bob
- Bob can't prove that Alice sent him M,
- he can only prove she signed S

non-repudiation with secret keys

- there exists notary N trusted by Bob and the judge
- Alice sends M to N, and N knows it came from Alice
- N does a computation on M with a secret key, getting H, which N seals to the message
- e.g., MD(Alice's name, message, S(n), time)
- N sends message on to Bob with seal
- Bob can later get N to state to judge that message is real ...

anonymity

- anonymous remailers have existed for quite some time
- historically have been cracked down upon
- why would you guess?
- if you could send anonymous email, could you send it to an anonymous destination?

3 types of cryptographic email

- 1. PEM - early development in IETF
  - digital signatures and privacy
  - assumed certificate hierarchy
- 2. S/MIME - MIME with PEM-like crypto
  - assumes same certificate hierarchy as found with ssl in web-world
- 3. PGP - similar crypto to PEM
  - several versions
  - “web of trust”; i.e., exchange of public keys is not PGP’s problem

ASCII versus the world?

- SMTP email uses ASCII by definition
- line in theory uses <CR><LF>
- unfortunately we also have email clients that want to mix html with email
- or creative ways to send binary data encoded in ASCII cybercrud (base64)
- we can pack characters with 6 bits of data into ASCII bytes, expanding info by 1/3rd
- ASCII cybercrud is needed for cryptoemail

crypto email mechanisms

- must use ASCII, but encode parts of it for cryptographic needs
- resulting message if not encrypted should be readable by humans but may not be
- message may be sent in two forms then, plaintext and in cybercrud format

Privacy-Enhanced-Mail/PEM

- 4 RFCs
- RFC 1421 - message formats
- RFC 1422 - CA hierarchy
- RFC 1423 - base set of crypto algorithms
- RFC 1424 - mail message formats for certificates
- MIME was also on the way, RFC 2045
- S/MIME, RFC 2633, took PEM design principles and plopped them into MIME format

infrastructure note

- we assume pgp is at the client/server
- but email gateways do not understand it
- so this (as with most L4/L7 uses) is

–end to end

PEM designers

- assumed both private keys and public keys would be used
- S/MIME sticks to public keys
- assumes many protocols including
  - RSA, DSS
  - DES, 3DES, AES

PEM message

- PEM block has:
  ------ BEGIN PRIVACY-ENHANCED MESSAGE ----
  cybercrud
  -------END PRIVACY-ENHANCED MESSAGE ----
- PEM can deal with these types of info:
  1. plaintext
  2. integrity-protected only (MIC-CLEAR term is used)
  3. intregrity-protected encoded data (MIC-ONLY)
  4. encoded, encrypted, integrity-protected (ENCRYPTED)

order of operations for the last for encryption, not signing

- compute integrity check on message
- create random encryption session key
- encrypt message, and hash
- then encode key, hash, encrypted message so that mailers can deal with it

see text, p. 531 and 532 for examples

- ...

PEM certificate hierarchy

- defined hierarchy based on X.500 names
- this is hierarchical tree
- e.g., assume /world/us/oregon/multnomah
- /world/us/ CA that issues certs for /world/us/oregon, etc.
- eventually there must be global hierarchy
- PEM designers wanted PEM to work before said hierarchy existed, therefore mail could include chain of certs

a word from Ancient Rome

"Sed quis custodiet ipsos custodes?"

Juvenal's satires

not: "who cleans up after the custodians" ...

(thanks to Dave Aucsmith)

problems include:

- we may assume organizations are strict about issuing certificates
- but what if commercial cert-authority X gives a cert. to anyone?
  - how trustworthy is that?
- or if organization B refuses to accept certs from organization X as a matter of policy
  - they are at war ...
- what if CA private key is compromised?
- what if private key for the ROOT CA was compromised?

other problems

- how does a university and its students differ
- from a defense contractor and its employees
- Intel and its employees?
- should a university require mandatory drug testing?
- RSA patent existed and did not expire until 2000, some did not care for RSA monopoly

Certificate Revocation List

- obviously certificates need to time out
- how do we notify the world?
- proposal: list old/bad certificates and circulate it
- what problems can you see with the idea of a certificate revocation list?
- any other ways certificates might be revoked?

S/MIME

- naturally uses MIME to deal with encoding
- S/MIME info is placed inside MIME wrapper
- can send cleartext signed message
- can encode said message
- Context-type: application/pkcs7-signature
  - a signature is encluded as a mime-type

GAAAAA!

- S/MIME uses ASN.1 to encode header info and data.
- not as readable as PEM (in a twisted sort of way)

S/MIME certificate hierarchy

- does not assume ONE public key infrastructure
- may use pubic certifier like Verisign/Thawte
  - different levels of assurance for customers
- may get certs within an organization
  - list certs within organization in directory like LDAP
- Alice gets Bob to mail her his certificates
  - perhaps Bob has cert signed by self-signed root certificate that Alice already has

so what about the following scenario?

- Krazyizona decrees that digital signatures are legally binding
- Attorney General of Krazyizona sets up state CA for issuing certs
- Alice gets such a cert and intends to use it
  - for signing her bills
  - and sending secret messages to Bob, who she is dating
- what could go wrong in such a scenario?

PGP

- homework assignment will be issued at this point

PGP

- created by Phil Zimmerman as “guerilla freeware”
- classic version used RSA and IDEA
- author wanted it to be distributed freely
  - but US considered it dangerous at the time
  - Phil got to go to court
- PGP was therefore free abroad, because RSA patent was US-only

Phil's Quote

"If privacy is outlawed, only outlaws will have privacy"

P.Z.

several versions

- do not necessarily interoperate
- PGP classic version (idea/RSA)
- patent-free version used DSS, DH, 3DES
  - src code was published as book as books had no export restrictions
- IETF redesigned and called their version "Open PGP"
  - Gnu Privacy Guard (GPG) is a variation on that

PGP overview

- pgp can send
  - authenticated
  - encrypted email
- can also
  - encrypt files
  - protect file integrity

key distribution

- you decide which users to trust
- and how trustworthy are the keys anyway
  - depending on how you got them
- you need the other party's public key
- PGP fingerprint: crypto hash of key
  - you can thus use this info (say from a web site, or on a business card) to sanity check a key that you get, and avoid a MTM attack

certificates

- are possible
- and so are certificate paths
- you may have a key for Eduard
  - signed by Jim
  - signed by Bob
- servers exist with PGP keys on them
- PGP signing parties have occurred

key ring

- a key ring is a PGP data structure that contains public keys
  - info about people
  - certificates
- you can decide how much you trust certain keys/people
  - none/partial/complete
  - you might not trust certs signed by Fred, but you will still verify messages from him

final thoughts

- consider the trust model for email:
- you get email from
  - strangers
  - business partners inside/outside enterprise
  - friends/family
- so email from grandma has a virus ...
  - if you and grandma use PGP does that help?
- where exactly could crypto/email be useful?

what are the real threats with email?

- how does the speed of virus/worm transmission impact things?
- do you think spam is a fixable problem?
- when can we send attachments securely?
- what about the problem of identity spoofing
  - anyway to fix that?
- can we detect spam and delete it before it even gets to the user?
  - all email from AOL/yahoo must be spam?