An Introduction to TCP/IP Network Security

Jim Binkley- jrb@cs.pdx.edu

outline

- overview
  - what I am talking about (and not ...)
- policy
- attacks (theory and not so theory)
- crypto
- building secure enclaves (aka firewalls)
- protocol layers and security services

overview

- focus here on Network and Secure Network Design
- network protocols + a few basic tools
- NOT system administration && os
  - don't care about data in file systems
  - do care about data across network
- NOT cryptography algorithm internals (e.g., how does RSA really work?)

but 1st a word from our sponsor

- useful books:
- *Building Internet Firewalls* - Chapman/Zwicky, ORA book
- *Network Security* - Kaufman/Perlman/Speciner
  - about application of crypto to network protocols
- *Applied Cryptography* - Bruce Schneier
  - cryptogram plus other things
- *Hacking Exposed* - McClure, Scambray, George Kurtz
  - (attacks on specific OSen)

security policy and application

- you need to decide what you want to protect and
  - inventory what you are doing (email/web/modems/NFS/distributed database)
- then decide how to protect it
  - back it up
  - throw it away or wall it off
  - improve authentication, add encryption
  - use XYZZY to solve all known problems

goals 1st, then implement

- write down a list of (achievable) goals:
  - 1. only do SMTP to one box and only allow the outside world to do email to that box (establish an email bastion host)
  - 2. only allow one box real web access (run a web proxy)
  - 3. use only strong authentication (oops, there goes telnet/ftp) for remote virtual terminal use (or pc anywhere ... remote windows))
  - 4. don't use product X from vendor Y (bad track record)

and do a little homework

- what kinds of attacks are possible and have been made in the past?
- what kinds of attacks can you practically hope to deter?
  - small business can deter Joe Bob Hacker, can't deter nation state security agency
- what the heck are you doing now with networking (and for the future)
  - and be totally right ...

bottom line

- policy means what you allow and what you deny ..
- users need to be educated
- management must buy in
- security is NOT a matter of one silver bullet
- but a matter of

  the weakest link in the chain

know and study net protocols in use; e.g.,

- X - block at firewall (at least try ...)
- NFS - can’t proxy it ... block at firewall
- telnet/ftp - hmmmm...anon ftp is ok though
- lpr - block block block
- Sun NIS - see previous line (hard to fwall acl)
- DNS - control access to your DNS server
- NNTP - network news - block outside world
- HTTP - maybe proxy server
- IPX? IP doesn’t forward IPX (modems?)

and understand them too

- RPC based, uses what ports? tcp/udp?
  - can proxy it? can block ports? ip addrs ?
  - Sun RPC (not NFS) juggles ports (ouch)
- X - TCP app.
  - client/server but server is terminal, reversed from normal way you think about client/server
  - clients run on arbitrary hosts out there
  - clients connect to port 600X.. range of them

need to know network topology too

- dialup/wireless access to what parts of network?
  - modem right into IPX server could be threat
- what “portals” to outside world exist
  - to Internet
  - dialup access (can clients act as routers?)
  - laptop with modem in it, wireless card, acts as router?

abstract security qualities

- **authentication** - proof that you are who you say you are
- **confidentiality** - keeping data secret
  - may include **encryption** technology
  - encrypt(plaintext data, key) -> ciphertext
  - might just make it impossible to get at data or keys
- **integrity** - data has not changed
- **anonymity** - ignored in past but may be of more interest RSN (web cookies?, etc.)

kinds of attacks

- **virus** - program gets free ride in over network (modem, floppy disk) as “java applet from hell” - proceeds to do bad things
- **worm** - program seeks to replicate itself over network
- **trojan horse** - looks safe on outside, has ancient and angry Greek Warriors on inside
  - download me! (it then mails your password file to a bad guy)

attacks

- authentication failures
  - password guessable, not strong enough
  - yellow sticky on computer ...
  - not strong enough system to begin with
    - » 4 letter PIN code [0-9][0-9][0-9][0-9] or plaintext over net
- **passive** (somebody reads your secrets as your packets go by)
  - including passwords or grade reports or fire letters
- **active** - somebody does a format c: on your pc
  - intrusion (bad guy is where he should not be)

attacks

- **masquerade** - somebody says they are you (and last you knew, you hadn't been cloned)
- **denial of service** - somebody prevents you from using a resource
  - your mail inbox always has 1000 "spam" letters in it ...
  - conventional wisdom: "hard to fix"
- **man in the middle** attacks
  - Alice to Bob with Kevin in the middle
  - Kevin can read (confidentiality), etc. and pretends to be Bob to steal Alice's letter to Bob (fire Kevin ...)

host OS vs network security

- UNIX divides world into root and non-root
- UNIX root can do anything, attacker seeks to use setuid and become root - famous sendmail trapdoor - Morris Worm
- this is called **escalation of privilege**
- may be exploited over network (so-called buffer overflow on root server)
  – or from multi-user o.s. (bad password ...)

closer attack - easier attacks

- physical access usually means you own the computer
  - e.g., easy to break in as root on unix
- multi-user attacks - easy to become root/supervisor
  - single user or few users is more secure
- network attacks - fewer known “exploits” than multi-user attacks
  - common goal: break in as user X, then use escalation of privilege attack

the morris worm - 1988

- fundamentally used two mechanisms to break-in (then use rsh or password attacks to fan-out)
- **buffer overflow** on fingerd
  - exec'ed "sh" by loading new code and having it executed as root
- exploited sendmail debug feature
  - sendmail runs as root server
  - execute desired commands remotely

morris fanout attacks

- Morris Worm - attack on rsh "authentication" in terms of ~user/.rhost
  - worm 1st guess ~bob's password and then attack other systems through ~bob/.rhost
- therefore **IP address authentication is oxymoron**
  - authentication based on allowing service to IP src address X too easy as X may be spoofed
- X11/nfs/lpr/rsh (rcp/rlogin)/pop all protocols that have made this assumption one way or another
- dictionary attacks on passwds in /etc/passwd

other network-based attacks include:

- shared network password capture
  - break into box X with some other technique
  - fan out by using sniffer to capture telnet/ftp passwords (or whatever sends passwords in plaintext)
  - harder now due to ethernet switches - less promiscuous mode
- arp spoof on same link can allow you to make use of trusted IP authentication

acc. to Steve Bellovin (or someone)

there is a packet out there somewhere with your system's name on it ...

call this: "ping of death"

recent D.O.S. attacks

- tcp syn attack - tie up TCP control block
- land attack - "connect to yourself" (one tcp packet to any port)
- teardrop attacks - UDP based incorrect IP fragmentation (any port)
- smurf attacks - use directed broadcast so that multiple pings can use up WAN link and beat to death your enterprise www server

virus attacks

- you download java applet AND/or get MIME message AND/or Active X Microsoft word doc AND/or ftp download and execution of “shar.exe” and it does
  - rm -fr ./$USER   OR
  - format c:\   OR
  - del *.*   OR
  - something even more horrible

observation/s

- many attacks are due to bugs
- why do we have software bugs?
  - code rushed to market
  - no consequences for security bugs in commercial software?
  - code doesn't get fixed even when patches are available
    - » IT can't spend all of its time upgrading everything
- what did Turing have to say on the subject of bugs?

esoteric attacks

- not usually found in the real world ...
- including
  - 1. tempest radiation - Van Eck phreaking
    - » pick up/display of Electromagnetic radiation
  - 2. covert channels - party A can somehow extract a message from party B thru an unexpected communication channel
  - (two processes/shared register)

crypto

- overview
- symmetric crypto
- hash/MAC/message digest
- asymmetric crypto
- DH
- signatures
- certificates
- CAs

overview

- there are MANY crypto algorithms and MANY academic network secure protocols
- how they are used in network protocols is another matter
- traditional IETF RFC said under security considerations (at end of doc)
  - “not considered here” (another F. Flub)
- new IETF POV: must consider here

symmetric encryption

- both sides know OUT OF BAND shared secret (password, bit string)
- msg(key, P) -> C (encrypted)
- encode/decode use same key (symmetric)
- algorithms include: DES, 3DES, IDEA, BLOWFISH, RC4
- ssh uses 128 bit key'ed IDEA
- DES key 56 bits - 0xdeadbeefdeadbeef

pros/cons

- pros
  - faster than public-key crypto
  - can be arbitrarily fast with hw support
- cons
  - keys may need to be changed often if too short
  - shared secrets do not scale in general to many users
    - » more people know secret, less of a secret
  - secrets hard to distribute

challenge-response with DES

- assume client/server

client: server:

------------ send ID (bob) -->

<---- send random challenge X

compute E = f(X, DES key)

------ send E to server -->

decode(E, key)

== X

- authentication mechanism (shared secret)

media digest algorithms

- take a message, and produce a non-reproducible bit string (a **hash**)
- MD(msg) -> bit string/or digest
- MD(msg, shared secret)-> authenticator
  - in this case, call it Message Authentication Code (MAC)
- may be used for password mechanisms
  - longer strings better, FreeBSD 128 byte passwd length
- used with signatures for efficiency reasons as public-key crypto much slower (only sign hash)

examples

- MD5 - media digest 5, 128 bit string (key)
  - used with RSA public-key signatures
- SHA - secure hash algorithm (NIST), 160 bit string
  - used with Digital Signature Standard (FIPS 186)
    - » algorithm called Digital Signature Algorithm (DSA)
  - uses SHA for hash
- HMAC versions of above used with IP SEC and other secure protocols (md(md(key,msg)))

Diffie-Hellman algorithm

- guess who invented it
- public key but doesn't do signatures/encryption
- allows two entities that share two public numbers to arrive at a shared secret that can be used for encryption of further messages
- one way to do "session key" algorithms
- share secure channel and periodically change key (e.g. use DH to start, DES for bulk work) for dynamic rekeying function

asymmetric or public-key

- key generation produces (Public, private) key pairs
- can give Public key away, secure private key
- two important services possible (RSA):
  - signature - append bit string that proves you signed a message, uses private key
  - confidentiality - uses public key

signatures

- can "sign" a message
- sign(M, private key)
  - but actually
  - use Media Digest algorithm to compute hash
  - say MD5 -> 128 bits (hash(M) -> bit string)
  - then run private key over bit string to get signature
  - send (Msg, signature)
- recv uses sender public key to verify

confidentiality

- you send me secure email
- 1st obtain my public key
- encrypt(Msg, public) -> encrypted message
- (ok you have to uuencode it ...)
- I decrypt with my private key
- ? how did you get my public key
- ? what if Joe spoofed me with his public key and you sent him a msg for me

so note four operations with RSA

- sign (mac hash) with private key
- verify (mac hash) with public key
- encrypt with public key
- decrypt with private key

session-key generation method

- server sends client its public-key
- client generates random number and encrypts with public-key
- sends random number back to server which decrypts with private-key
- at end: both sides have shared secret
- can use it for authentication and/or encryption with symmetric function

algorithms include:

- RSA - company and algorithm
  - invented by Rivest, Shamir, Adleman
  - key lengths 512/1024, etc.
  - block size is smaller than key length
  - output will be length of key
- DSS - US govt replacement (no encryption)
- Diffie - Hellman (older than RSA)
  - doesn’t allow signatures/encryption

certificates

- are a signed public-key
- basically (subject name, issuer’s name, subject public key, issuer’s signature, validity period, internal bits ...)
- signed by trusted authority (authority uses private key to form signature)
- to verify cert. public key, you must have public key of certificate authority
- cert. can be small file or part of network message

formats

- X509 (as used with netscape S/MIME email or HTTP/SSL)
- PGP (as used with PGP email)
- DNS signed public keys (signed by zone)

Certificate Authorities

- it is presumed that one way to solve the problem of public key distribution
- is to get a signed public key from a trusted 3rd party
- call that node a CA - certificate authority
- nodes need the CA's public key to start with
- can verify "certificate" signed by CA
- certificate = Joe Bob's public key, CA sig

certs, cont.

- certificate can be stored anywhere
  - only CA can generate them
- CA doesn't have to be accessible
  - but would be if network database of course
- so why don't we have CAs as public-key infrastructure (talk to with protocol)
  - who runs it?
  - netscape supports certificates and there are a few CAs
  - "cross-certification" as opposed to hierarchical cert. may not be reasonable due to trust problems

firewalls

- intro
- packet filters (routers)
- proxy services (application gateways)
  - bastion hosts

intro

- **firewalls** control access - one or more machines that constrain access to an internal network
- firewalls may allow you to implement rule-based policies
- "choke point" (moat and drawbridge with guard tower) - centralize admin
- don't serve to ENABLE but DISABLE
  - just say no ...

basis of firewall rule-set

- policies start from
  - 1: accept all packets and deny a few bad things
    - » (no NFS in/out, no TCP to port 139, else OK)
  - 2. deny all packets, and only accept a few
    - » (to bastion hosts that support email/http)

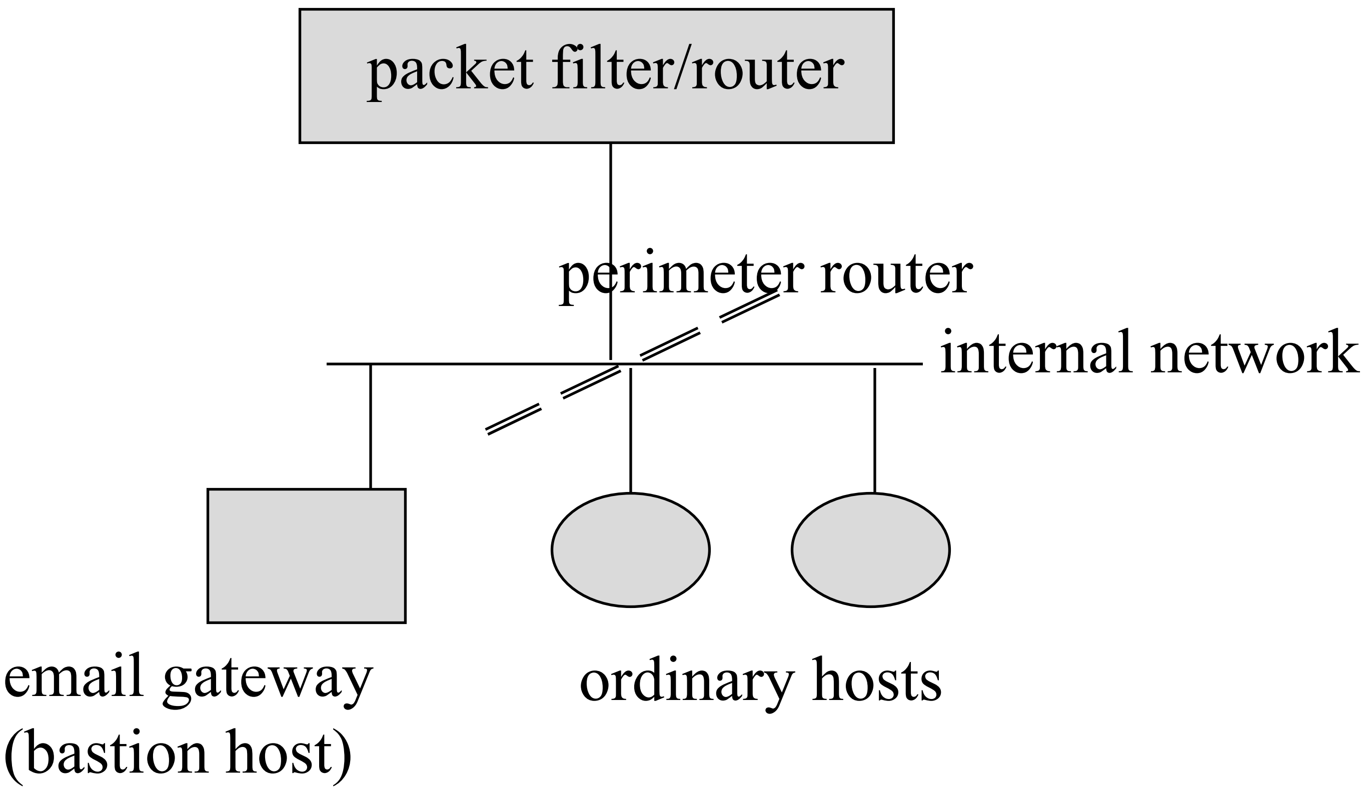
intro

- may act via **packet filtering**: (net layer)
  - router allows/blocks pkts acc. to IP src/dst, UDP/TCP port numbers, in/out port X,Y,Z
  - you setup rules that allow what goes through
  - e.g., block UDP port 2049 either in/out
- may have **proxy service** at app level
  - **bastion host -** system exposed to attack that typically offers up ONE service (email) to Internet

intro

- may choose defense in depth or due to admin. reasons have **perimeter network** (or DMZ)
  - have to get over TWO drawbridges
- dual-homed host - users can login to this system only to get out (unclean)
- victim machine - place to try out something new and dangerous (don't care what happens to it)

firewall picture



may have 2nd perimeter router

- put bastion hosts on DMZ
  - subject to attack by definition
  - allow access to host X for TCP and port 25 (email)
- wall off interior hosts via 2nd network/router
- attacker can attack bastion host and then interior host, but not interior host directly

packet filters

- typically associated with network layer/routing function (but peek at transport headers)
- use IP src/dst, protocol type, tcp/udp src/dst ports, ICMP message type
- router knows i/f packet arrived on or is trying to escape on
- can understand IP networks as well as IP host addresses
- stateless - makes per packet decisions

pros/cons

- pros
  - large scale tool - can turn off all telnet access or all access to subnet X or to proto Y
  - can deal with NEW service because it doesn't know about it
  - efficient (compared to proxy)
- cons
  - logging is harder because you may not have app/protocol knowledge
  - getting rule base right for ALL protocols is tricky (especially accept all deny a few)

proxy services/bastion hosts

- bastion host - typically one per service
  - NO user logins - users can bring their own programs with them
  - web proxy server
  - email proxy server (easy)
  - anonymous ftp server
  - cut down on all other ways to attack interior hosts
    - » rlogin is a bad idea ... or lpd ... or NFS

proxy service

- may require user to use a certain procedure (ftp to box X, then ftp out) OR
  set netscape client to point at X, port 8080
- a particular proxy service can be good at logging and offer better granularity access control
- may try and filter viruses, java applets
- may require modified software

proxy services

- pros
  - finer grain control over applications
    - understand the protocol
  - better logging
  - very tight accept a few, deny all (doesn't forward pkts)
- cons
  - need new code if something new comes along
  - can't do everything (proxy NFS is a weird idea?)
  - have to be careful with bastion host setup

systems exist that are hybrids

- firewall that contains packet filter AND proxy system and combination therein
- stateful inspection idea - smarter packet filter
  - can keep state machine, thus predict what next packets should be
  - see DNS/UDP out to box X, knows there should be reply

proxy services - examples

- TIS Toolkit
  - individual proxies for common apps
  - telnet client to TIS/box X,
    - » get prompt that allows you to telnet out only
    - » can't store files locally
  - ftp proxy
  - "generic" proxy called plug-gw
    - » specify limited range of addresses/ports, use with NNTP

examples - SOCKS

- TCP-only, and a redirection protocol
- need a socks server and socks-ified clients
- socks client library for UNIX boxes
- socks apps like telnet/ftp
- clients talk to socks server rather than real world
- not protocol specific, logging is generic
- access control by host/protocol

security up the network stack

- link layer
- network layer
  - ipsec
- transport layer and apps
  - pgp
  - ssh
  - kerberos
  - ssl

link layer

- HW encryption exists; e.g., all packets encrypted with DES
  – not so bad if point to point
  – LAN, multiple instances of shared secret
- needs to be fast as (or faster) than link
- PPP uses challenge-response authentication (CHAP) based on shared secret (password)
- con: security measures do not cross links
- pro: useful if link deemed less secure than average (radio)

network layer

- various research attempts to bind security in ABOVE IP header

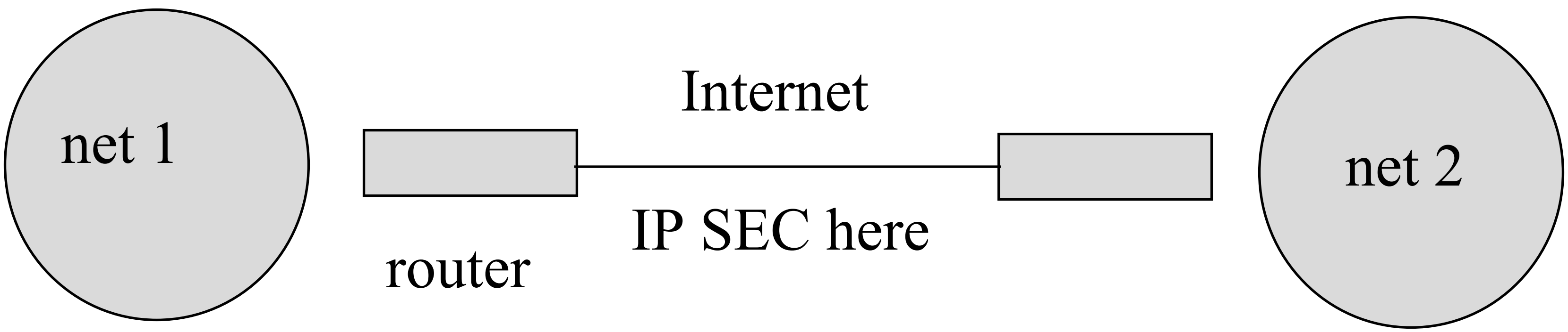
  IP <security header> <TCP>
- might apply to routes or to end to end transport
- current IETF work called IPSEC - IP security
- must apply to IPng, and can apply to IPv4

network layer pros/cons

- pros:
  - can be end to end or at least multi-link unlike link layer
  - could be hw/sw supported because in o.s.
  - can shield dumb apps from needing security support (and dumb hosts, or even nets of hosts)
  - can extend secure enclave across insecure areas
- cons:
  - harder to do as may be INSIDE O.S.
  - if not end to end, subject to certain kinds of attacks'
    - » proposed plaintext attack

Virtual Private Network

dumb hosts with dumb protocols



all pkts from net 1 to net 2 subject to
authentication/confidentiality
(and vice versa)

IP level security/bibliography

- Stallings - Cryptography and Network Security, Prentice Hall
- RFC 2401, “Security Architecture for the Internet Protocol”, Kent/Atkinson, 1998
- RFC 2402, “IP Authentication Header”, Kent/Atkinson, 1998
- RFC 2406, “IP Encapsulating Security Payload (ESP)”, Kent/Atkinson, 1998
- RFC 2407, “The Internet IP Security Domain of Interpretation for ISAKMP”, Piper, 1998.

we are not done yet ...

- RFC 2408, "Internet Security Association and Key Management Protocol" (ISAKMP), Maughan and others, 1998
- RFC 2409, "The Internet Key Exchange(IKE)", Harkins, Carrel, 1998
- RFC 2412, "The OAKLEY Key Determination Protocol", Orman, 1998
- RFC 2411, "IP Security Document Roadmap", Thayer, others, 1998
- per crypto "transform" documents for AH/ESP, e.g., md5/sha/des, etc.

IPSEC protocols

- AH - authentication header
- ESP - encapsulating security payload
- multiple headers above IP header, before transport headers
- AH + ESP are done per packet (bulk crypto)
- ISAKMP/OAKLEY - dynamic negotiation of session keys for AH/ESP
- now called Internet Key Exchange. IKE = ISAKMP + OAKLEY

AH

ip hdr	ah = spi, MD hash, next proto value, anti-replay	TCP

AH header breakdown (v2)

next hdr	length	reserved
Security Parameters Index (SPI)		
Sequence Number		
hash from one-way function (variable)		

ESP

ip hdr	spi, IV, anti-replay, may have authent. hash		esp next proto
	esp	tcp/data	trailer

encrypted parts ....

ESP header breakdown

SPI (SPY vs. SPY?)
Sequence Number
payload data (variable)
padding 0.255 bytes + pad len + next hdr
optional authentication bits (variable)

note: IV may appear at front of payload

IPSEC may be used

- router to router (so-called tunnel mode)
  - this means entire ES datagram encapsulated
- end system to router (still tunnel mode)
- end system to end system (transport, not tunnel mode)
- user to user, except that O.S. do not yet support this kind of functionality

tunnel-mode process

- router A takes packet from IP node ip src = 1.1.1.1 to ip dst 2.2.2.2
- A is 1.1.1.2 and B is 2.2.2.1
- A adds new IP header and required AH and/or ESP headers encapsulating entire datagram
- new outer IP hdr, ip src = 1.1.1.2, dst = 2.2.2.1
- A sends packet across as IP <IPSEC> , IP datagram
  - tunnel to B as destination
- note outer IP and IPSEC bound together, inner datagram including its ip hdr encrypted

router B gets packets

- B verifies contents acc to AH/ESP, decrypts in latter case
- strips outer IP and associated IPSEC headers
- routes packet (remaining datagram) with possible interior IPSEC/application security to final local net destination
- nested IPSEC can always occur

more IPSEC

- SA - security association: classically one way (as is routing):
  - (ip src, ip dst, AH or ESP, SPI)
- SPI is opaque number that is mapped to a particular algorithm (DES or IDEA say)
- SPI - security parameter index
- AH/ESP by themselves assume manual keys or session keys placed in kernel

ISAKMP (now IKE)

- ISAKMP - key mgmt. protocol
  - OAKLEY is session key protocol “inside”
- e.g., use RSA to authenticate ISAKMP exchanges
  - sets up SPIs on both ends
  - uses Diffie Hellman to create session-keys
  - then AH/ESP per packet can go ahead using well-known MAC/symmetric encryption

pgp - pretty good privacy

- sign, encrypt email
- pioneered idea of using public keys/signatures/encryption for secure email
  - symmetric key signed by public key (RSA)
  - bulk encryption done by idea
- no CA, just send your public key "out of band"
  - finger/email/floppy …
  - note: private key on-line encrypted with passphrase

pgp, cont

- other folks public keys stored in “key-ring”
- use your public key to send you email
- send a encrypted letter:
  - get joe’s public key, store in keyring
  - make up letter
  - run pgp (using joe’s public key) to encrypt (and produce ASCII output)
  - suck letter into mailer and send it
- pgp can also encrypt files on disk

ssh

- secure replacement for BSD r* utilities
  - rlogin <- slogin
  - rsh <- ssh
  - rcp <- scp
  - rshd <- sshd
- OPINION: throw rsh* out
- v1 uses RSA authentication, idea encryption (or your choice, des, 3des, arcfour, blowfish)

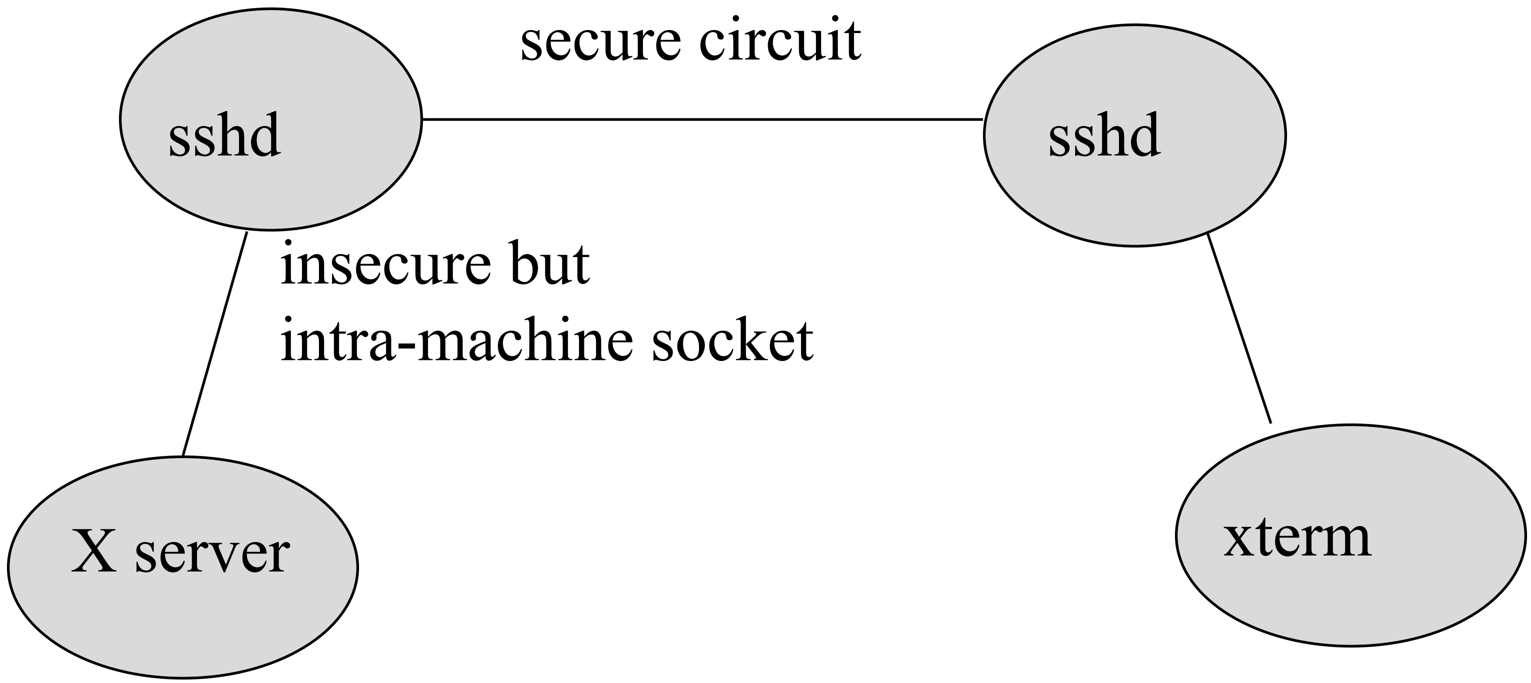
ssh

- no certificates (yet), user must get public key to both sides (**you are your own CA**)
- if you don’t have RSA public key on other side, prompted for password (still not sent in clear)
- ssh available for download from Finland for almost all UNIX systems
  - commercial windows client exist

ssh in action

- generate a key:
  - % ssh-keygen
- get key to the other host
  - put (cat) in ~user/.ssh/authorized_hosts
- slogin other.cs.pdx.edu
  - slogin -l jrb other (if no key over there)
- scp -r foo.dir jrb@sappho.cs.pdx.edu:
- can do remote X clients over ssh

ssh/X apps


secure circuit
sshd
sshd
insecure but
intra-machine socket
X server
xterm

kerberos

- not recent, MIT/1988, Project Athena
- provides authentication to services on hosts
- user/service shares symmetric key with KDC (key distribution center), local server
  - DES used as password for user
- does NOT use asymmetric keys, presumed to be less scalable as a result
- apps talk to kerberos servers to perform authentication

kerberos cons

- modify apps
- nontrivial to administer, and must be centrally administered (unlike ssh)
  – server must be secure
- doesn't scale beyond single admin domain

ssl (and ssleay)

- secure socket layer - ssl
- netscape designed
- goal: public-key authentication/encryption for TCP apps (web clients/servers)
- not use HTTP (shttp, secure http)
- can view as transport layer mechanism
- proposed now in IETF as Transport Layer Security (TLS == SSL v3.1)
- find in netscape products/elsewhere

netscape crypto - US version

- ssl/rsa/rc4/md5
- ssl/rsa/3des/sha
- ssl/rsa/des/sha
- your netscape browser speaks certificates ...

protocol ideas

- app protocol on top (say http …)
- ssl handshake protocol
  - authenticate client/server and choose encryption
- ssl record protocol
  - encapsulate packets in crypto
- tcp as underlying transport

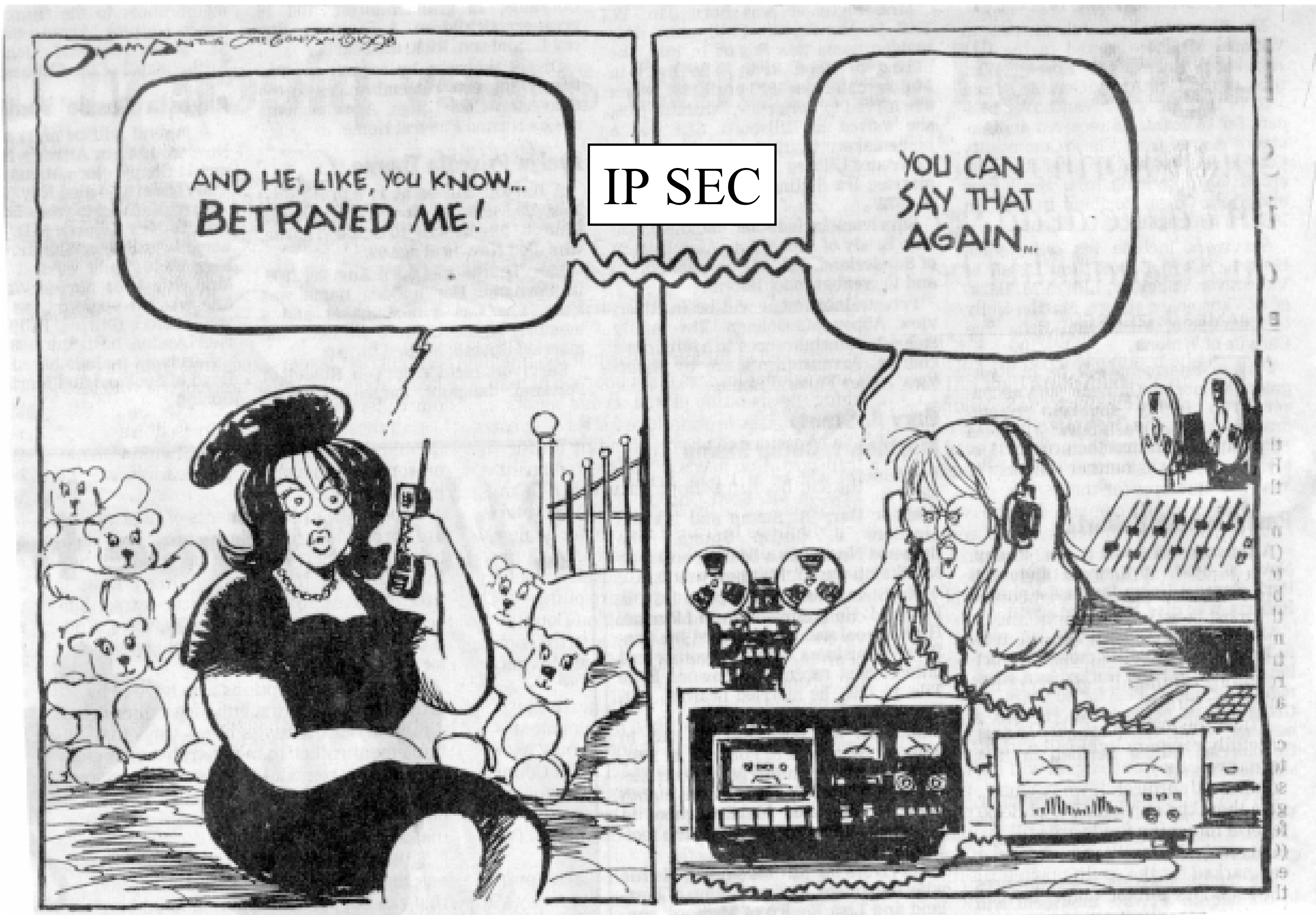
ssleay (see www.openssl.org)

- public domain effort to make ssl more widely available (site in OZ)
- can download ssl library
- do up various apps
- lots of them at this point
  - web servers and telnet …
- can setup your own CA

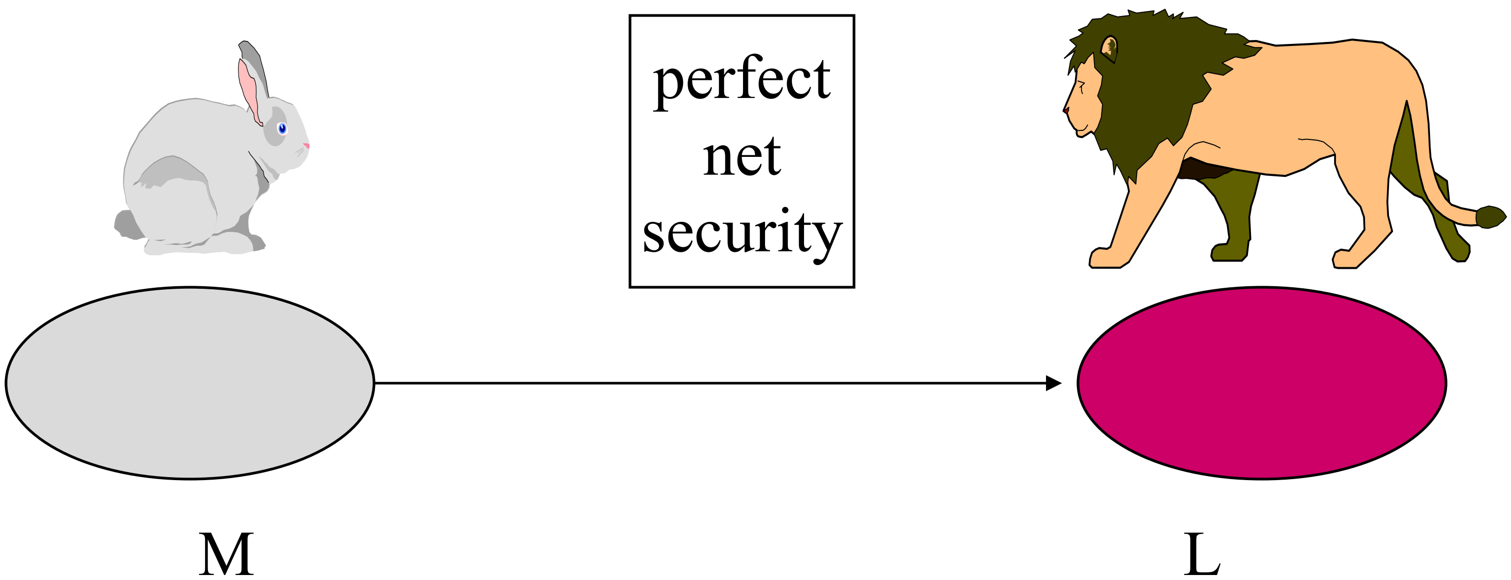
layer summary

- which layer is right?
  - note tendency of upstairs apps to be TCP only
- ssh or kerberos or ssl or pgp?
- certificates (what kind, what model of trust, how to authenticated names work) not done yet, but started ...
- DNS security is incredibly important …
  - not just for dns but for what is stored in dns

assume ipsec, M. got what?



assume ipsec, M. got what?



security is based on trust/risk

- as well as security tools
- assume: **perfect Inet-wide IPSEC**
- does this mean “**perfect security**” ?
- **no** ... you still have to trust the other side or the other network (engineers)
- a single VPN or secure web transaction by itself does not give cross Inet security

what can we do to make computers less insecure?

- minimize sw bugs
  - avoid buffer overflows
- minimize exposure of any given host
  - turn it off if you don't use it
  - find out which ports in use ...
- patch it or update it with new sw
  - hard to keep up
- avoid unsafe apps with lousy track record
- use cryptography where possible
  - ssh as opposed to telnet/ftp

conclusions

- security ultimately relies on human trust and human relationships
- many/most sw/security flaws are sw engineering failures
- and/or management failures
  – oops. should have *tested* the backup redundancy plan
- new sw exists (mail/ipsec/ssh) that can be useful, but caveat emptor

no silver bullet

- no matter what the firewall vendors say ...