Network Security - Firewalls

Jim Binkley

outline (more like high points)

- intro
- network design
- ACLs
  - cisco
  - ipfw
- proxy servers (e.g., tis)
- other mechanisms, socks, tcpwrappers, IDSen, Linux iptables

great: define firewall

- denies packets …
  - not allows packets
- what attributes are there? what instances?
- a web-proxy that filters http based on RULES
  - is a firewall
- a linux router using iptables and snortsam is what? (is it an IDS or a firewall?)
- how about linux router + Layer 7 pattern matching?
- what properties should a firewall have?

is this a firewall?

- dns server
  - has rule base (evil zone names)
  - denies access to local hosts if they lookup
    - evil.org
  - http://www.emergingthreats.net/rules/emerging-botcc.rules
- email server with clamav
  - drops email if it mentions X

one sacred rule for firewalls

- it is highly like to do something you didn't expect
  - misconfigured
- what do we do about this?

bibliography

- *Inet Firewalls FAQ*: Ranum/Curtin http://www.clar.net/pub/mjr/pubs/fwfaq
- ***Building Internet Firewalls*** - Chapman/Zwicky, ORA book, 2nd edition
- *BCP 38, RFC 1918*
- *Firewalls and Internet Security*
  - Bellovin/Cheswick, Addison-Wesley, 1994

why firewalls?

- you have 1000 WNT 4.0 hosts/servers
- winnuke appears on the planet
- what do you do
  - patch 1000 WNT boxes?
    - » and restore all the apps ...
  - block winnuke at the firewall?
  - disable Inet access to the WNT boxes?
  - nothing (call your lifeline?)

policy

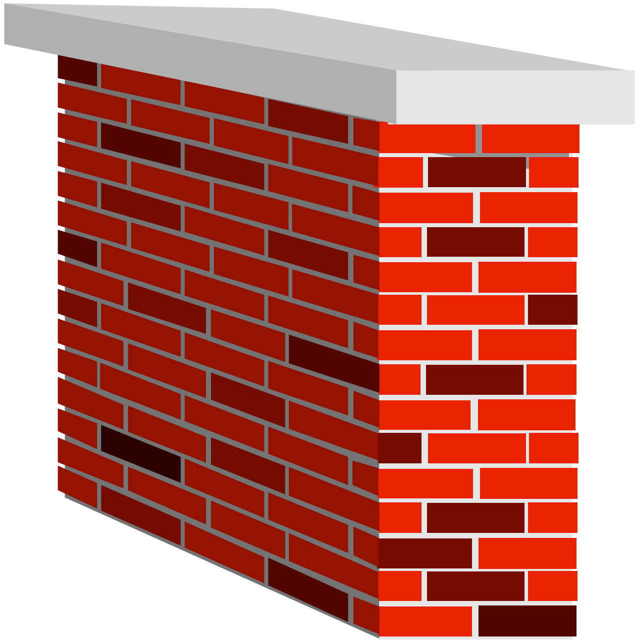
- ◆ you need to decide what you want to protect and
  - – inventory what you are doing (email/web/modems/NFS/distributed database)
- ◆ then decide how to protect it
  - – wall it off (firewalls ...)
  - – throw it away
  - – improve authentication (one-time keys ...)
  - – use XYZZY to solve all known problems

theoretically

- policy should be top-down
  - write it and implement it
- often bottom-up
  - evaluate current practice and improve it
  - especially may happen post disaster

no silver bullet

- no matter what the firewall vendors say ...



assume ipsec, M. got what?

AND HE, LIKE, YOU KNOW...
BETRAYED ME!
IP SEC
YOU CAN
SAY THAT
AGAIN...

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) or your employees
- a single VPN or firewall by itself does not give cross Inet security
  - you still have to trust the people
- and have sane security processes/practices

firewall not enough because

- social engineering attacks
  - I'm from IT and I need General BigNeck's password
- lack of physical security for computer console
  - can you say "L1-A?"
- secrets in the dumpster
- secrets on the floppies (usb these days)
- secretary mails business plan to alt.general
- employees have found real-video South Park site
  - this could be a real problem if you are in the cartoon biz

end-to-end thesis and firewalls

- they disrupt end to end transport relationship
- as does NAT
- as does QOS (ahhh ... but we have soft state)
  - implicit tie to fate-sharing is true
- hope is for **world without firewalls**
- **this is not a practical hope ...**

Marcus Ranum - the 6 dumbest ideas in computer security

- see www.ranum.com
- 1. default permit as opposed to default deny
  - firewall
  - install any app on host
  - where else (think about google)?
- 2. enumerating badness (variation on above)
  - just how many bad sites on the web
  - is google.com ever bad?
  - sometimes we have to do this
    - » it is what an IDS does even if it isn’t the firewall

4 more

- 3. penetrate and patch
  - his point: testing by trial and error as opposed to designing good software from day #1
  - we always have more patches
    - » more 3rd party than major vendor these days
- 4. hacking is cool
  - therefore pay hackers big bucks to penetrate and patch

2 more

- 5. educate users (and the world will be better)
  - isn’t it better to remove the dynamite and lock it up? e.g., remove executable attachments from email
  - instructor doesn’t agree
- 6. action is better than in-action
  - ancient Chinese principle of wu-wei
  - let somebody else be an early adopter

firewall/IDS basic ideas

- stateless vs stateful
- stateful means “connection table”
  - IDS may have it, FW may have it, NAT
- inline by definition (can’t be out of line)
- host or intermediate (aka network-based)
- stop a moment and define
- packet
- flow

our friend the packet

- IP hdr
- ip src, ip dst, next proto UDP/TCP/ICMP,ESP,
- TCP/UDP hdr
- well known/dynamic ports
- how useful are they?
- TCP flags

the relationship between errors and L4

- TCP SYNs to empty port gets TCP reset
- plus some ICMP errors
- UDP packet to empty port gets ICMP unreachable
- firewalls may use this or abuse it
- “great firewall of China” syn spoofing plus resets (IPS)

flows

- a MESS of packets from IP src to IP dst
- from
  - IP src -> IP dst with ESP
  - IP src, L4 src -> IP dst, L4 dst TCP,UDP
- when does it stop (how do you clock it?)
  - probably with a state table and a timer
- STATE needed for stateful firewalls, router flow optimization, NAT, IDS systems
- note that L7 info may be lost or unavailable
- this mechanism may be about information aggregation

flow example

- 131.252.X.Y, port 1024 -> google IP, port 80, TCP, syn | fin | 12 packets, 1400 bytes
- google IP, port 80 -> 131.252.X.Y port 1024, etc (reverse flow)
- 131.252.X.Y, port 6666 -> random IP, port 6666, 1 packet
- 131.252.X.Y, port 6667 -> random IP, port 6666, 1 packet
- 131.252.X.Y. port 6668 -> random IP, port 6666, 1 packet

flows found in:

- Cisco netflow tools (NFSen, cflow, silktools, etc).
  - network traffic mgmt, security possible
- Snort (can be stateful)
  - goal can be capture “connections” and make connection state decisions for IDS, as opposed to per packet
- NAT/stateful firewalls
  - allows “smart” decisions about what gets in or gets out
  - might be able to block syn scanning

intro

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

Chapman/Zwicky definition

- Firewall:
  “A component … that ***restricts*** *access between a protected network and the Internet …”*
- note: **restricts** does not mean **enables**
- **security reality-check: just say no**
  - it’s harder than it looks
  - **fundamental test of management support**
  - does not support programmer “add one more feature”

choke point means logging

- allow you to monitor/log what is going on
- you can watch one place better than 1000 places
- you CANNOT log everything
  - or log sufficient with lower-level tools like ACL-based systems in routers
  - proxy/host-based/apps better at this

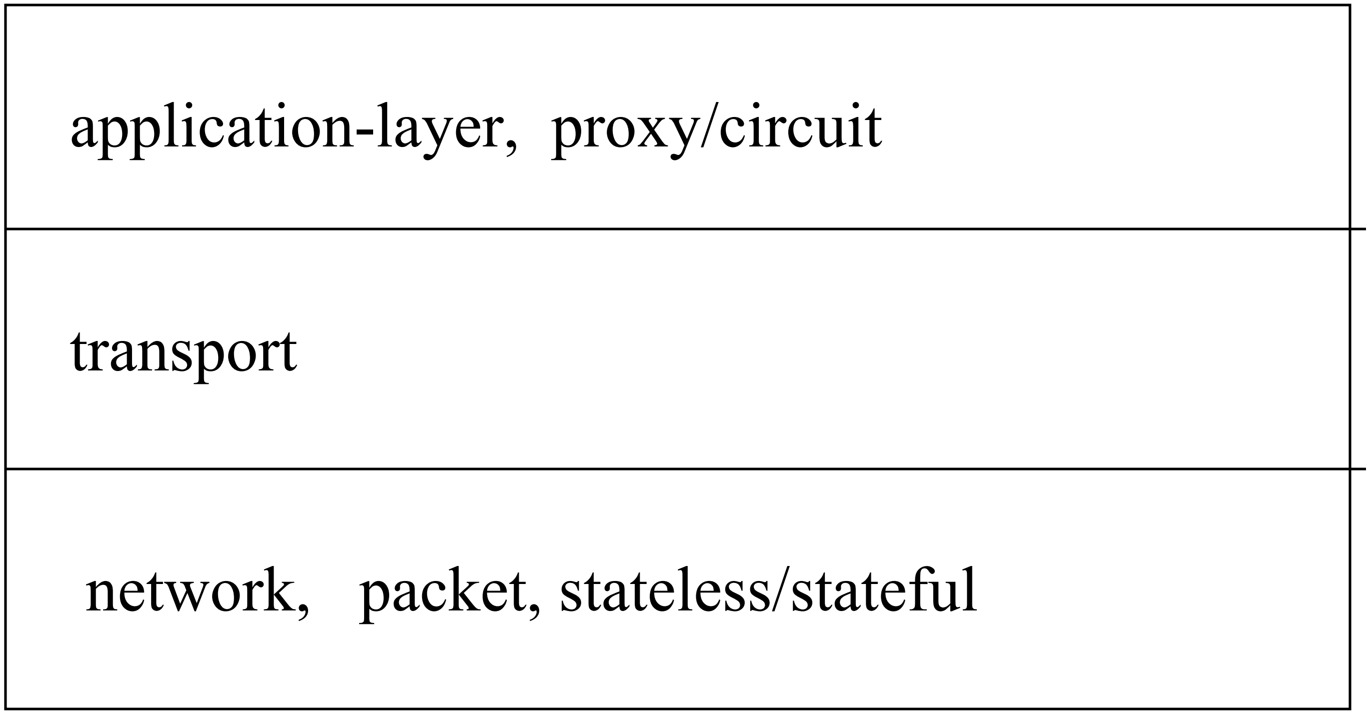
2+2 kinds of firewalls

- access-control-list mechanisms; i.e., **packet filters** at network layer
  - typically in routers (NLC), but may be found in hosts (ipfw, etc., e.g., in Linux/freebsd)
- application-level gateways, **proxy server**
  - **bastion host** typically has such a service
  - TIS firewall toolkit classic example
  - web-based proxy very common now

two more possible forms (sub-forms)

- **stateful packet** systems
  - e.g., “stateful inspection”
  - use state machine so you can learn what to expect in terms of response
    - » e.g., ftp out means ftp connect back in
    - » e.g., dns out means dns from X back in
- **circuit proxy** - use TCP, and talk to server that turns around and acts as client
  - good for logging/acl control, no content understand for a protocol

in general, stack-wise



application-layer, proxy/circuit

transport

network, packet, stateless/stateful

some example systems

- access lists - major router vendors/Cisco/Bay/etc.
  - even hosts - linux/freebsd have ipfw, iptables, etc.
  - and windows both usoft and 3rd party
- bastion host/TIS FW Toolkit
  - runs on UNIX platforms
  - gauntlet is commercial version (history)
  - http://en.wikipedia.org/wiki/Secure_Computing_Corporation (sidewinder may qualify???)
- stateful inspection
  - Checkpoint/Cisco PIX

some buzzwords

- **bastion host** - system that is made more secure due to Internet exposure, typically workstation
- **screened host/network** - host or network behind firewall/router, amount of protection depends on rules in firewall. said router is a screening router.
- **perimeter network/DMZ** - network (often internal) between internal secure nets and outside world
- **secure enclave** - what you get with perimeter-based security (secure all the exits/entrances)
- **defense in depth** - the notion that in addition to firewall one, you have host protection and internal firewalls, etc.

etc.

- **victim system** or **goat system**
  - experimental and sacrificial (honeypot qualifies)
  - maybe they are all victim systems?
- **intrusion detection** - looking for bad guys having landed (or little people?)
  - may take a number of forms
    - packet analysis, tripwire, log scanning, virus scans
  - may be regarded as defense in depth technique
  - may be regarded as internal defense technique

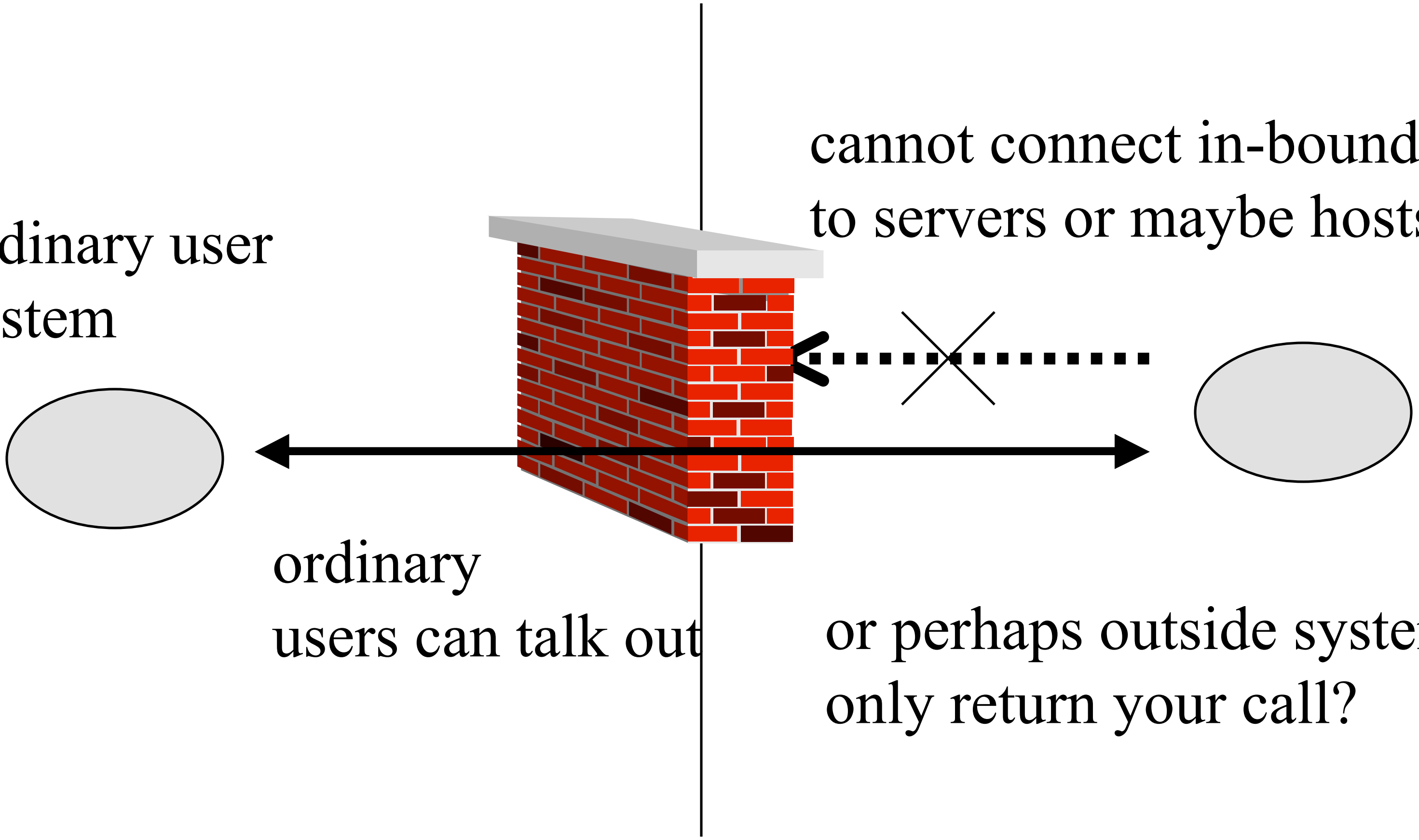
more ...

- honeypot - system or program on server that looks exploitable
  - but may actually serve as advanced warning
  - intrusion detection system
  - learn the motives, techniques, etc. of attackers
  - nepenthes - nepenthes.mwcollect.org
  - note that a sandbox is something slightly different (cwsandbox is example)

firewall architectures

- 1st of all - consider access to internal enclave systems
  - do they get to talk to Inet (and vice versa)
  - do they come in two classes (those that can and those that can't)
  - of course - no outside access is safer ...
- some possible firewall architectures follow

user systems can get out but bad guys are restricted getting in?



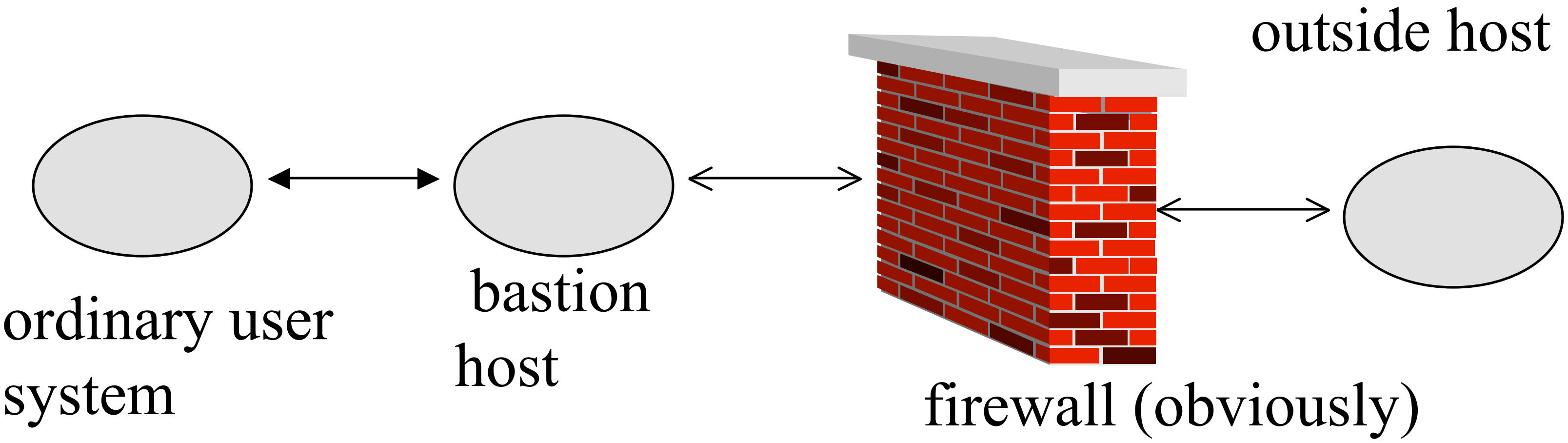
ordinary user
system

cannot connect in-bound
to servers or maybe hosts

ordinary
users can talk out

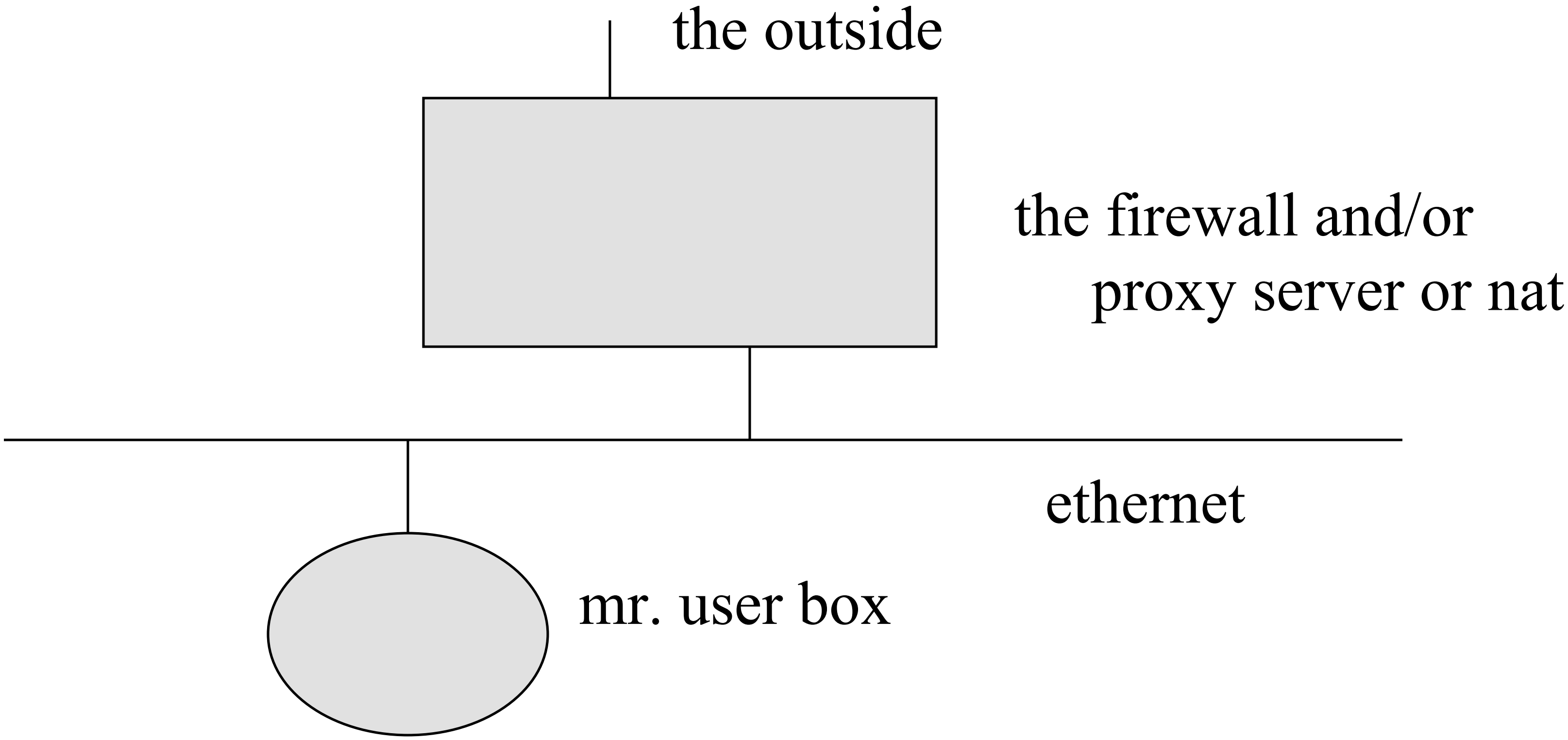
or perhaps outside systems can
only return your call?

users cannot get out period and vice versa

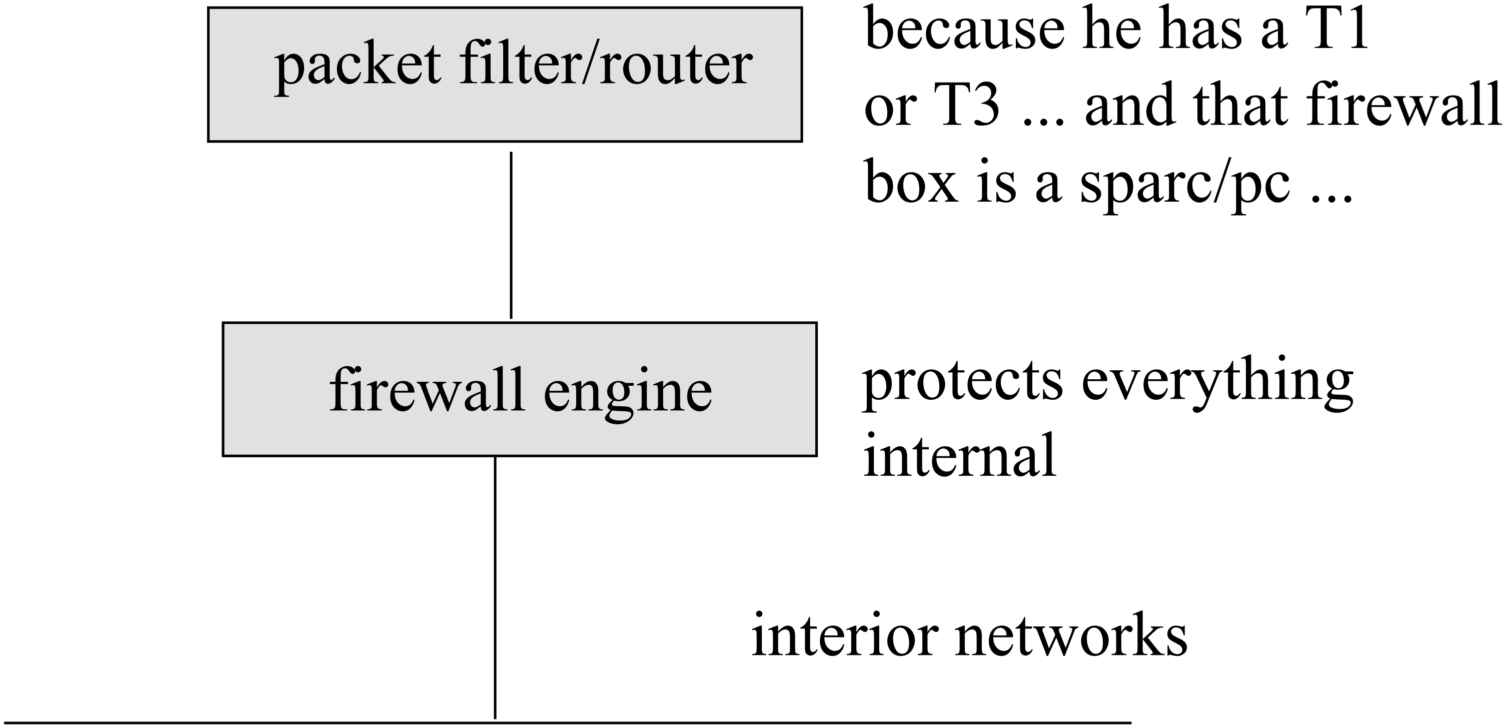


internal user systems cannot talk or be talked to from outside world - only through intermediary

arch #1, which can still vary internally depending on fw



silver bullet firewall picture



packet filter/router

because he has a T1
or T3 ... and that firewall
box is a sparc/pc ...

firewall engine

protects everything
internal

interior networks

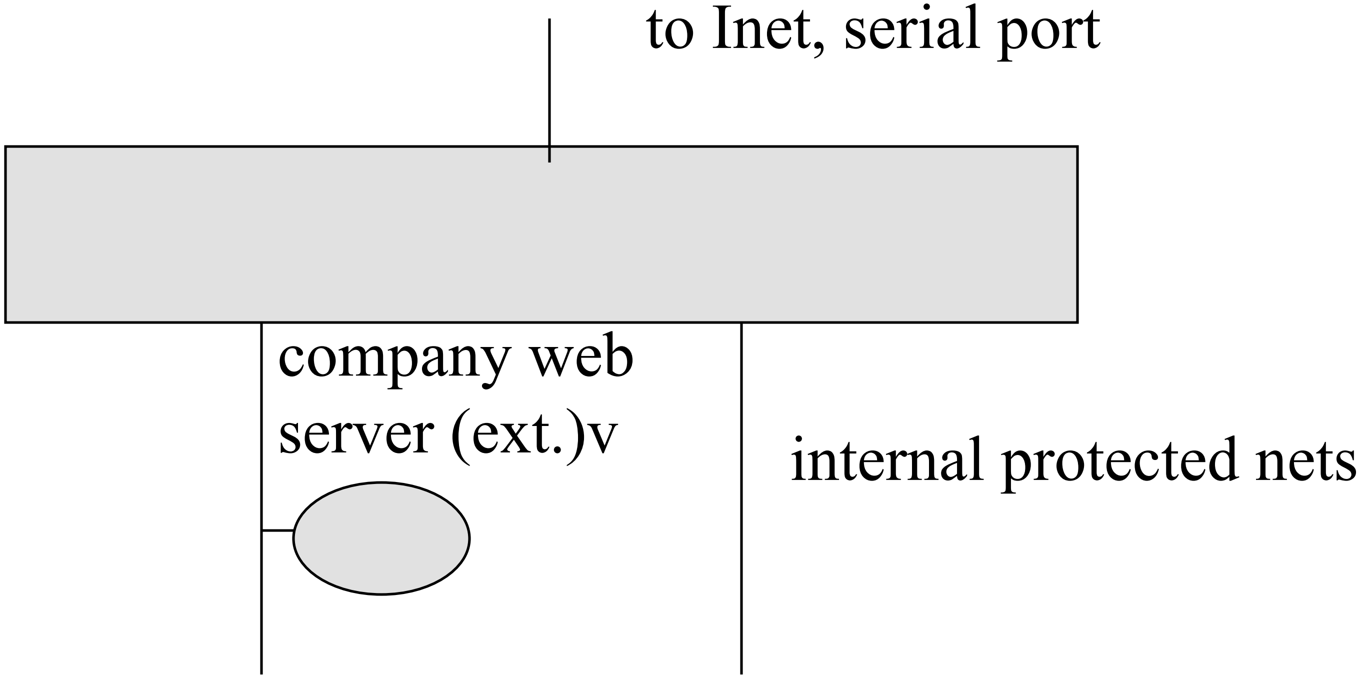
some scenarios

- a freebsd/linux pc, with proxy servers (email/web), possibly using host firewalling (acls) as well and/or NAT
- it's a cisco router with acls only
- it's an expensive firewall box
- the user host may or may not have access to the outside world (e.g., might only have proxy access to web/email)
- two box scenario - router can protect firewall with acls ... (can't telnet to it from outside world ...)

cont.

- **dual-homed** host with proxy not unusual
  - does not allow routing across
  - fairly secure/cheap solution
  - although there are cons
    - may be impossible with fancy WAN plumbing
    - hard disk is always a con in 7x24 access system

note: cheaper WAN router may look like this (cisco 26xx series)



to Inet, serial port

company web server (ext.)v

internal protected nets

two ethernet ports, 1 wan port out of box...

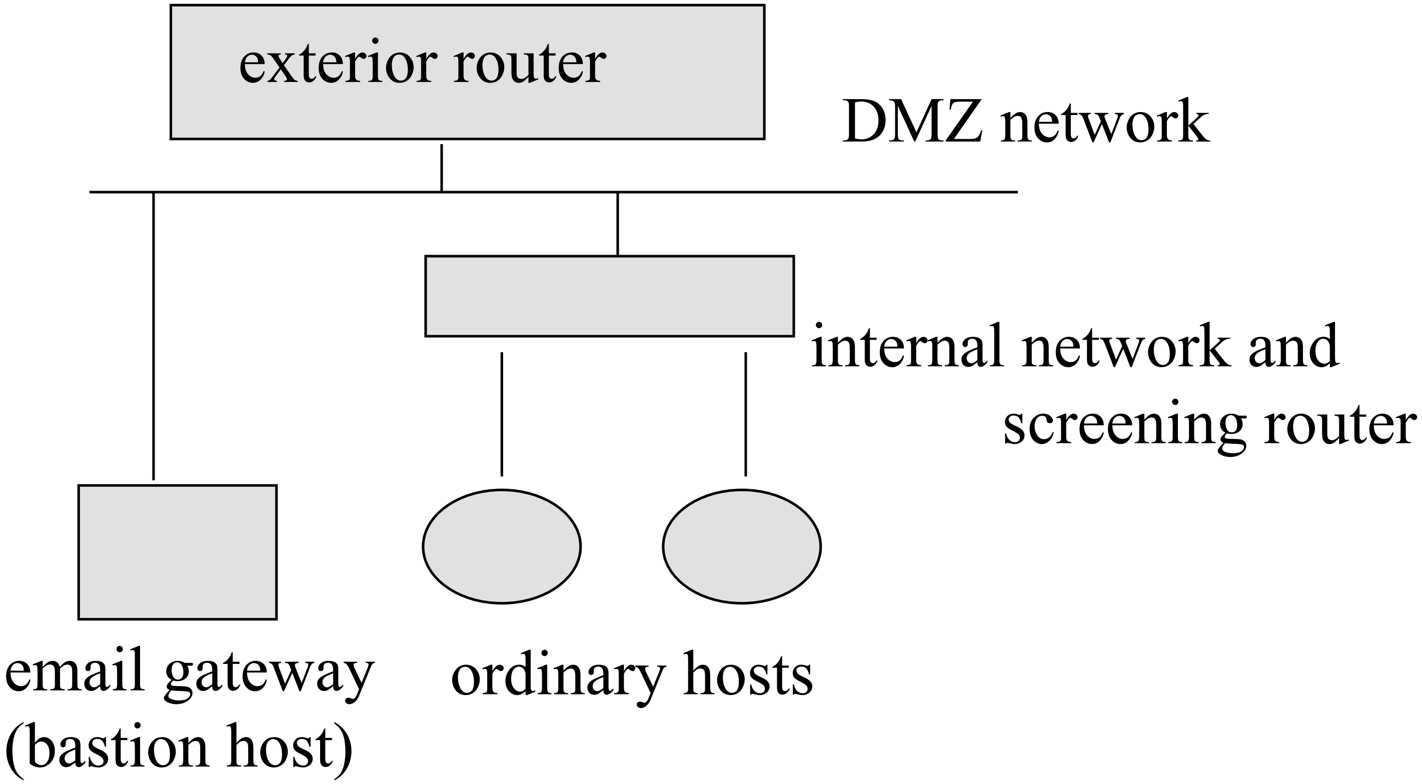
note to network engineers

- **the infrastructure has to be protected too**
- the routers/switches
- snmp writes ...
- the firewall is part of the infrastructure
  - if land succeeds on cisco router/switch or
  - brand X firewall
  - that is not a GOOD thing ...

RFC 1918

- 10.0.0.0 - 10.255.255.255 (10/8 prefix)
- 172.16.0.0 - 172.31.255.255 (172.16/12 prefix) - aka 16 class Bs
- 192.168.0.0 - 192.168.255.255 (192.168/16 prefix)

arch model #2 (classic)



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 that does screening
- 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, IP encapsulation types (ICMP, IPIP)
- router knows i/f packet arrived on or is trying to escape on
- can understand IP networks as well as IP host addresses
- should be able to log "denys"

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 (KISS because per packet decision)
  - more efficient than application gateway
- cons
  - logging is harder because you may not have app/protocol knowledge (no state machine)
  - getting rule base right for ALL protocols is tricky
    - especially if accept all, deny some is policy basis

packet filter plus steroids

- **stateful inspection**
- basically packet filters that are smarter and look at “connection” state (tcp or udp)
- e.g., can easily setup so that no internal access is allowed outside in
- external access is allowed inside out
- state: TCP out means expect TCP back in
- perhaps easy to teach about new protocols

policy considerations

- start with: **deny all, permit a few**
  - pro: most paranoid/proscriptive/most secure
  - con: cost to getting anything accomplished is the most high
  - **pro: less need to react to latest hacker discovery**
- start with: **allow all; deny a few** (known bad)
  - pro: least impact on Internet traffic
  - con: least secure, + need to stay up to date on hackerdom

oops - now we have to block port 10000

- https://isc.sans.org/diary.html?storyid=580
- note: interesting problem: what if some idiot host is using port 10000 dynamically for something other than veritas backup?

Example: deny all; allow a few

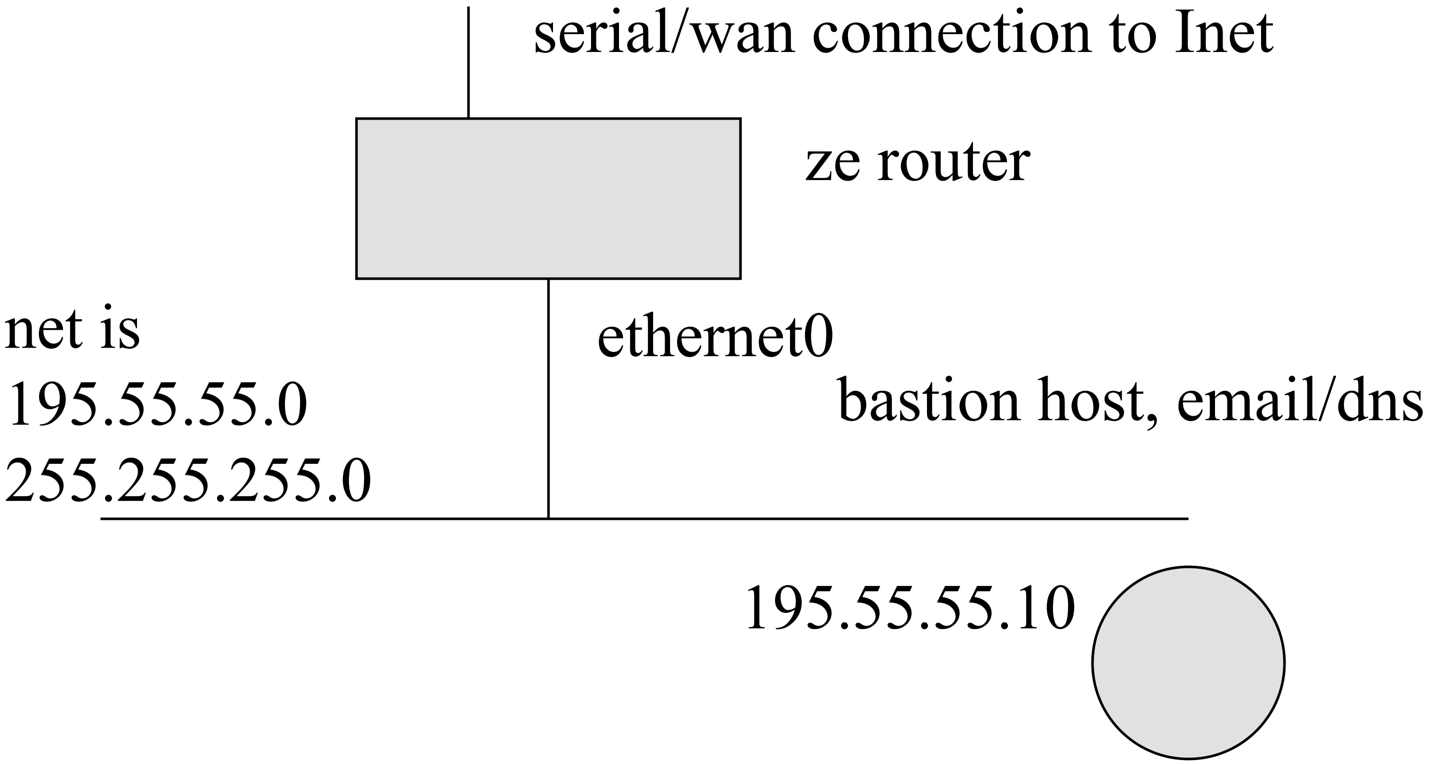
- no Internet traffic allowed to/from internal hosts except for proxies (application control gates)
- proxies include:
  - web proxy (easy/apache)
  - email proxy (easy/sendmail by definition)
  - telnet proxy
  - ftp proxy

Example: allow all; deny a few

- no IP spoofing (pkts leaving/entering must have IP src that make sense)
- no private IP addresses
- no directed broadcast 192.128.1.255
- no IP authentication-based protocols
  - lpr, X, nfs, rlogin, rsh
- no Microsoft TCP/NetBEUI (137-139)

Cisco acl example

- from Inet Firewalls FAQ



serial/wan connection to Inet

ze router

net is
195.55.55.0
255.255.255.0

ethernet0

bastion host, email/dns

195.55.55.10

but first, acl basics

- executed in order of list entries on a packet
- default deny at end (note: it’s invisible)
- basic form:
  - permit ip src-net src-mask dst-net dst-mask eq port
- permit or deny, log may appear at end
- access-list 101 permit ip 172.16.0.0 0.0.255.255 172.17.0.0 0.0.255.255
- mask sets bits for bits to ignore, therefore above means 172.16.X.X (any hosts in 172.16)
- net/mask may be replaced with **any** or **host 1.2.3.4**

Cisco deny all ACL example

- no ip source-route
- interface ethernet0
  - ip address 195.55.55.1
  - no ip directed-broadcast
- interface serial0
  - ip access-group 101 in
- access-list 101 deny ip 195.55.55.0 0.0.0.255
- access-list 101 permit tcp any any established
- access-list 101 permit tcp any host 195.55.55.10 eq smtp
- access-list 101 permit tcp any host 195.55.55.10 eq dns
- access-list 101 permit udp any host 192.55.55.10 eq dns

Cisco acl, cont.

- access-list 101 deny tcp any any range 6000 6003
- access-list 101 deny tcp any any eq 2049
- access-list 101 deny udp any any eq 2049
- access-ist 101 permit tcp any 20 any gt 1024 (note: ftp data connections from 20)
- access-list 101 permit icmp any any
- IMPLICIT DENY AT END OF LIST

Cisco ACL, cont.

- snmp-server community FOOBAR RO 2
- line vty 0 4
- access-class 2 in
- access-list 2 permit 195.55.55.0 255.255.255.0
- note: above allows snmp access from inside only and telnet access to router from inside only

egress filter on serial interface

- or input on ethernet interface
- interface ethernet0
  - ip access-group 102 in
- access-list 102 permit our-ip our-mask any
- access-list 102 deny ip any any
- thus no non-home packets in terms of ip src allowed out (hard on Mobile-IP)
- basic DOS mitigation

and now a word from Fergie

- BCP 38
- ingress filters
  - private IPs (net 10, and yourself coming in)
- egress filters
  - private IP addresses and not yourself going out
- 2 questions:
- 1. when does this help
- 2. what about bogon lists?

bogon lists and other things that go bump in the night

- 1. Cymru has nice list of unused net blocks and private Ips
- you know about 169.254/16 right?
- www.cymru.com/Documents/bogon-bn-nonagg.txt
- there are other more aggressive lists for “evil”

RBLs and C/Cs

- spamhaus.org has 3 lists (mail servers)
- 1. SBL - spam block list
- 2. XBL - xploits block list
- 3. PBL - list of hosts that should not be doing email (policy block list)
- OR www.bleedingthreats.net/fwrules
  - suitable for snort

cisco acl handout time

- more elaborate allow all deny a few
- deny all allow a few
- note mixture is possible
- next look at FreeBSD ipfw (from FreeBSD handbook)
  - similar to linux ipchains

host acl example - FreeBSD ipfw

- kernel must be configured with:
- options IPFIREWALL   # ipfw on
- options IPFIREWALL_VERBOSE # logging
- options IPFIREWALL_DEFAULT_TO_ACCEPT
- note: default deny can lead to damaged feet; i.e., be very sure the acl will allow you to access the box
- ipfw defaults to deny all ... otherwise
- IPFIREWALL_VERBOSE_LIMIT=10
  - limits logging on a per entry basis

ipfw toolkit

- simple packet filter
- also accounting stats for ip
- could be used as end host or for BSD-based router of course
- ipfw(8) utility is used for setting up rules
- command categories include:
  - addition/deletion, listing, flushing, clearing
  - flushing means wipe rules, clearing wipe accounting stats

ipfw

- ipfw [-N] command [index] action [log] protocol addresses [options]
- -N - resolve addresses and services in output
- commands: add, delete
- index specifies where in the "chain" (the list of rules) a rule goes, default is the end
- default rule is index 65535, deny
- if log specified the rule is logged

ipfw

- actions:
  - reject - drop and send ICMP host/port unreachable error
  - allow - pass it of course
  - deny - drop it, no ICMP
  - count - count it, but don’t accept/deny
- protocols
  - all/icmp/tcp/udp

ipfw

- address
  - from <address/mask> [port] to <address/mask> [port]   via <interface>
  - port can only be used with tcp/udp
  - via is optional and may be IP/dns or interface name (ed0), ppp* would match all ppp ports
  - address/mask-bits or address:mask-pattern
  - 192.1.2.1/24   mask-pattern is ip address
  - any may be used for any ip address

ipfw

- options
  - frag - matches if packet is not the first fragment of datagram
  - in - matches if the packet is input
  - out - matches if the packet is headed out
  - ipoptions <spec> -- for ip options
  - established - matches if TCP established state
  - setup - TCP syn
  - tcpflags <flags> - specific tcp flag bits
  - icmptypes <types> - specific icmp messages

ipfw commands

- ipfw l # list
- ipfw -a l # accounting counters too
- ipfw -t l # last match times for each rule
- ipfw -N l # dns resolve desired
- ipfw flush # wipe the chain
- ipfw zero [index] # zero stats

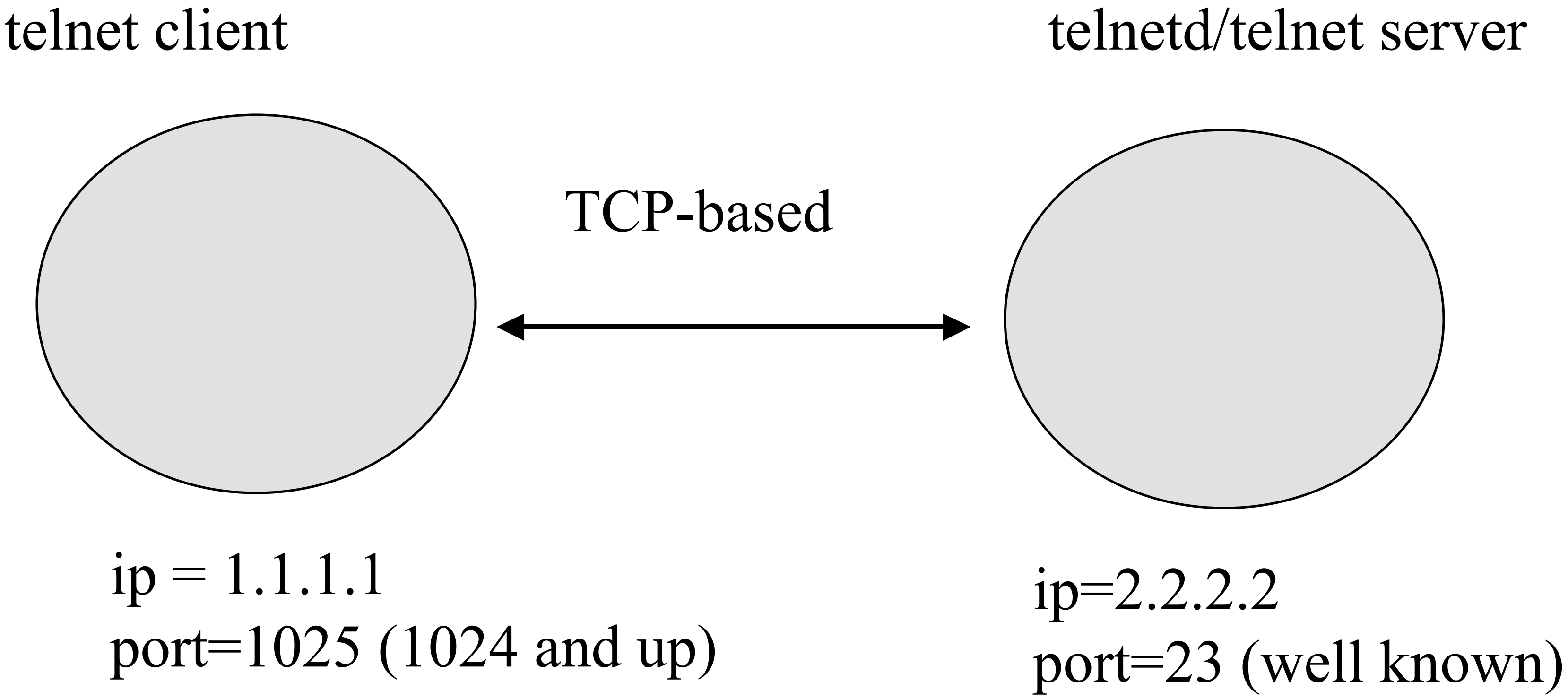
examples

- if we were a router:
  - ipfw add deny log tcp from evil.hacker.org/24 to nice.people.org 23
- deny all but allow web server traffic
- ipfw add allow tcp from any to me.me 80

application considerations

- we will look at some app behavior situations
- consider application port behavior
- this is historical and leads to complexity:
  - if deny all, how do we accept this app?
  - if access all, how do we deny it?
- the winner is probably still: h323

client/server telnet model



telnet client

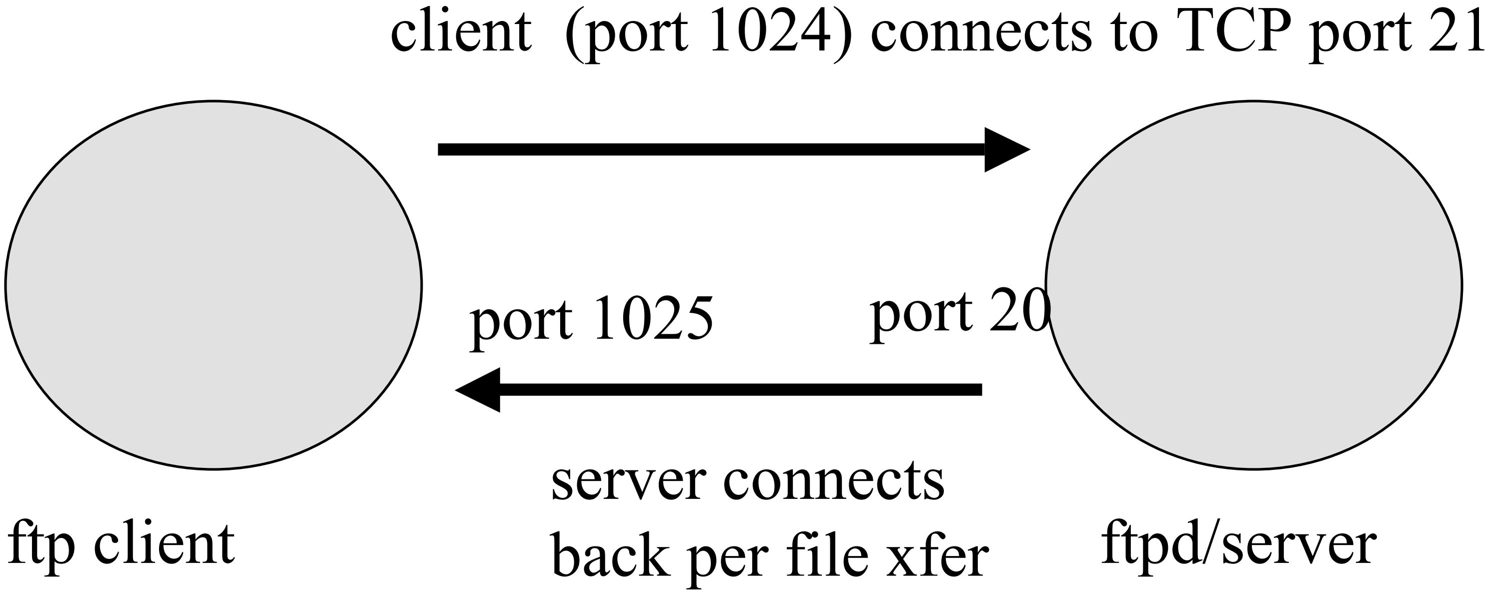
telnetd/telnet server

TCP-based

ip = 1.1.1.1
port=1025 (1024 and up)

ip=2.2.2.2
port=23 (well known)

ftp - non-passive-mode



client (port 1024) connects to TCP port 21

port 1025

port 20

server connects back per file xfer

ftp client

ftpd/server

in passive mode, ftp client connects to server

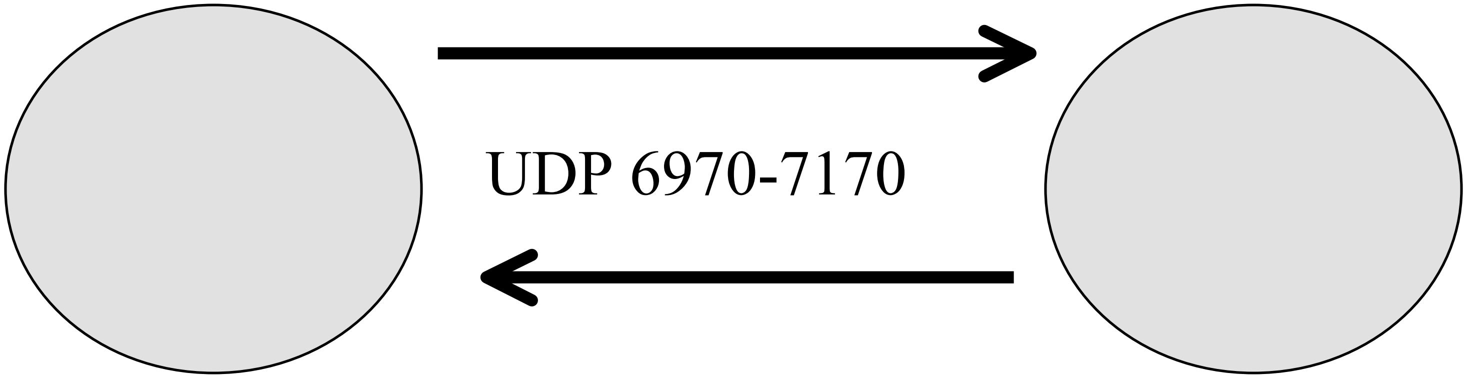
X11

client (port 1024) connects to TCP port 6000..X

xterm (or whatever) client

X/server/display

real audio



client (port 1024) connects to TCP port 554/7070

UDP 6970-7170

gui app (or whatever) client

ra server

Sun RPC

- portmapper - program #/tied to udp/tcp ports
- portmapper lives at port 111 (block ...)
- example attack: buffer overflow on rpc.statd
- NFS parts like mountd theoretically move around (they register with portmap at boot and get a port)
- NSF parts like nfsd do NOT move around (2049)
- rpc is painful and dangerous in terms of acl-firewalls
- Sun has had shadows ports > 32k (ouch)

study questions

- go thru previous 5 app slides
- and DOS attacks previously studied
  - teardrop is a good one
- use acls to alternatively
  - try to kill it (deny)
  - enable it with everything else killed
  - what problems exist?
- also ask the ?: what makes this particular app less secure? and what can we do about it?

issues for firewalls

- not too different from routers in some ways
  - e.g., redundancy, what about load balancing?
- o.s. that firewall is on should be MORE bullet proof than average
- lack of hard disk may be GOOD thing
- logging u/i is very important
- clues about how it works important too but ... may be hard to get (testing …)
- how well does it route? (maybe you don’t want it to route ...)

more issues for firewalls

- you bought an expensive firewall system that runs on a UNIX workstation
- what services if any does it allow through
  - that they didn’t tell you about?
  - how do you find out? (nmap ...)
- let’s say you let in port 111 for tcp to box X?
  - what else could go wrong? (e.g., how are application proxies in one way better than packet filters?)
  - consider the back-channel attacks or ftp on port 12345

acl cons

- port-filtering with HOLES (allow all) is hard and problematic
  - must know previous holes
  - latest bug on bugtraq - you need to know about it and fix the firewall
  - you block web access on the lower ports but user sets up proxy server outside on port 7777 and redirects their internal browser to use it
- can be tricky if rule list is complex
- con for really high-speed networking (sigh)
  - pro compared to proxy in terms of speed

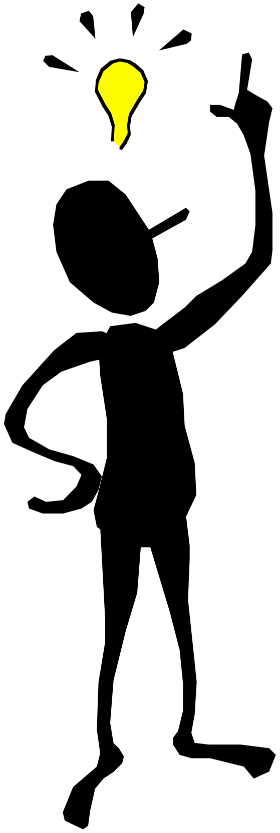
proxy services/bastion hosts

- bastion host - IDEALLY 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

please read this slide

- once more:
- NFS (rpc.statd or whatever buffer overflow of the day)
  - is a bad idea on a bastion host/proxy firewall
- so is Usoft CIFS (let’s share the password file by accident, what say?)
- does this mean that a Cisco router with ACLS is better? (than a sloppily setup bastion host?) - no NFS (fingerd though)

you must have a brain ...



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, but usually virus stuff left to virus scanners
- may require modified CLIENT software

proxy services

- pros
  - finer grain control over applications
    - » understand the protocol and harder to spoof
  - better logging
  - as deny all, more secure by definition
- 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
  - slower than packet acl mechanism

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

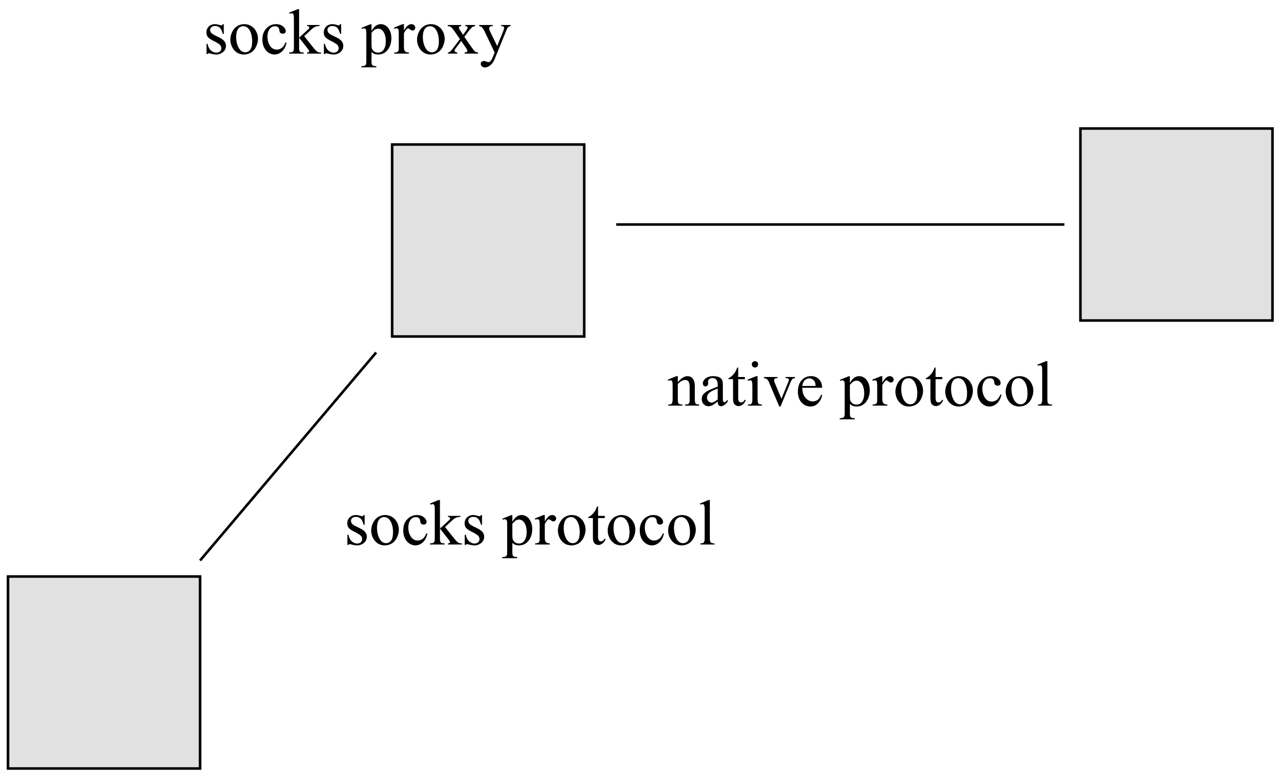
TIS, cont.

- http-gw: http/gopher proxy
- x-gw: X gateway
  - may be bad idea as X not very secure

circuit proxy - SOCKS

- originally TCP connections-only, and a redirection/circuit protocol
- need a socks server and socks-ified clients
- socks client library for UNIX boxes
- e.g., 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
- now may redirect ports at will

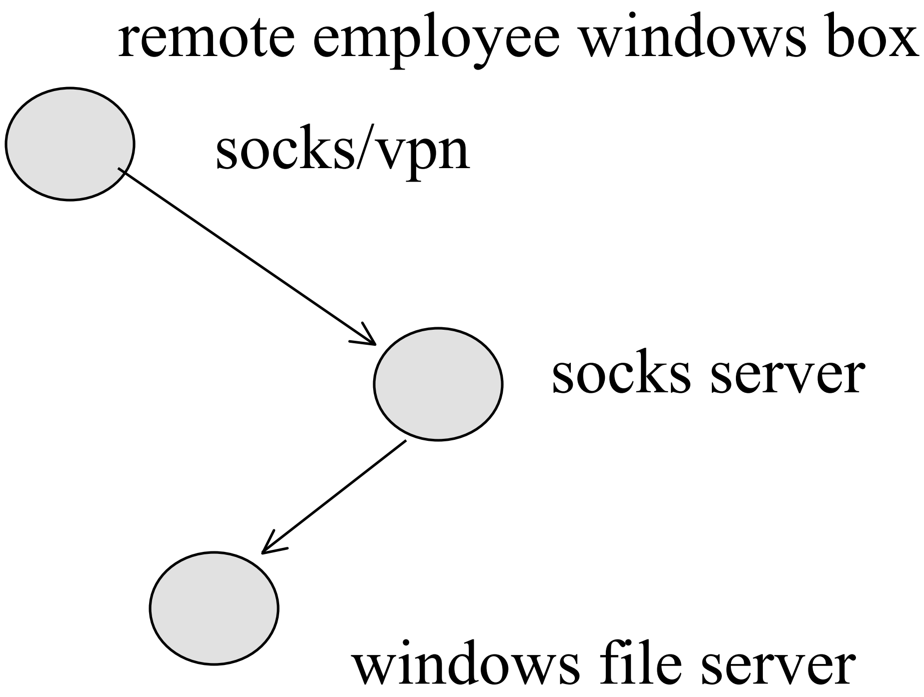
socks picture



incomplete list of proxy server functions

- web proxy - restrict outside access
  - can’t visit EVIL web pages (AUP function)
  - cache
  - fw restriction outside in as well
- socks(alike) proxy
  - turn email into encrypted http over port 80 in
  - so email in to email out (spam function)
  - possible form of remote control
  - socks may allow you to bypass the web proxy
  - may make access to rest of Inet anonymous

how about this topology though?



proxy servers may be “open” or “closed”

- closed means needs password
- open means go on through …
- question though:
  - if open, does it mean open by accident
  - if open, is it ‘watched’ (a honeypot)
  - can it just be open and be for free? (yes)
- although more complex, see TOR project: tor.eff.org (and now for the chaffing protocol)

wrappers and tcpwrappers

- basic idea: maybe we don’t have source ...
- security logic in one program encapsulates another program (which can be updated without typically breaking the paradigm)
- one wrapper may be able to deal with multiple wrappees ...
- examples: TIS smap wrapper for sendmail
- tcpwrapper by Wietse Venema
- socks ...

tcpwrapper - Wietse Venema

- ftp://ftp.win.tue.nl/pub/security or at coast
- inetd on UNIX starts tcpwrapper thus can wrap several programs (telnet/ftp e.g.,)
  - can be compiled into sendmail for that matter
- basically compares hostname/service to /etc/hosts.allow and hosts.deny files to determine if service is allowed
- logs results in syslog (you can log finger for that matter)

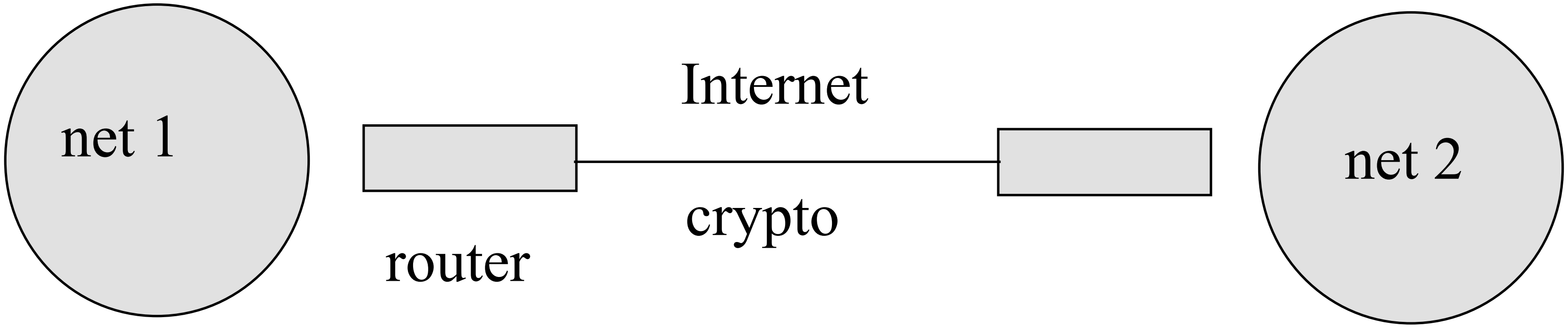
acl mechanism

- search /etc/hosts.allow 1st to see if it should be allowed
- search /etc/hosts.deny to see if it should be denied
- else allow it
- syntax:
  daemon_name: client_host_list [shell]
- e.g., all: badguys.net
- note: reliance on ip addresses here may be spoofable

Virtual Private Network notion

- firewalls may include VPNs in feature set
- glue together two secure enclaves with a virtual secure pipe; i.e., packets have crypto
- e.g., use confidentiality/authentication for all packets between routers A and routers B across the Inet
- of interest to businesses with private telco networks to connect their office
- dialup access too
- firewalls are beginning to have this feature

Virtual Private Network

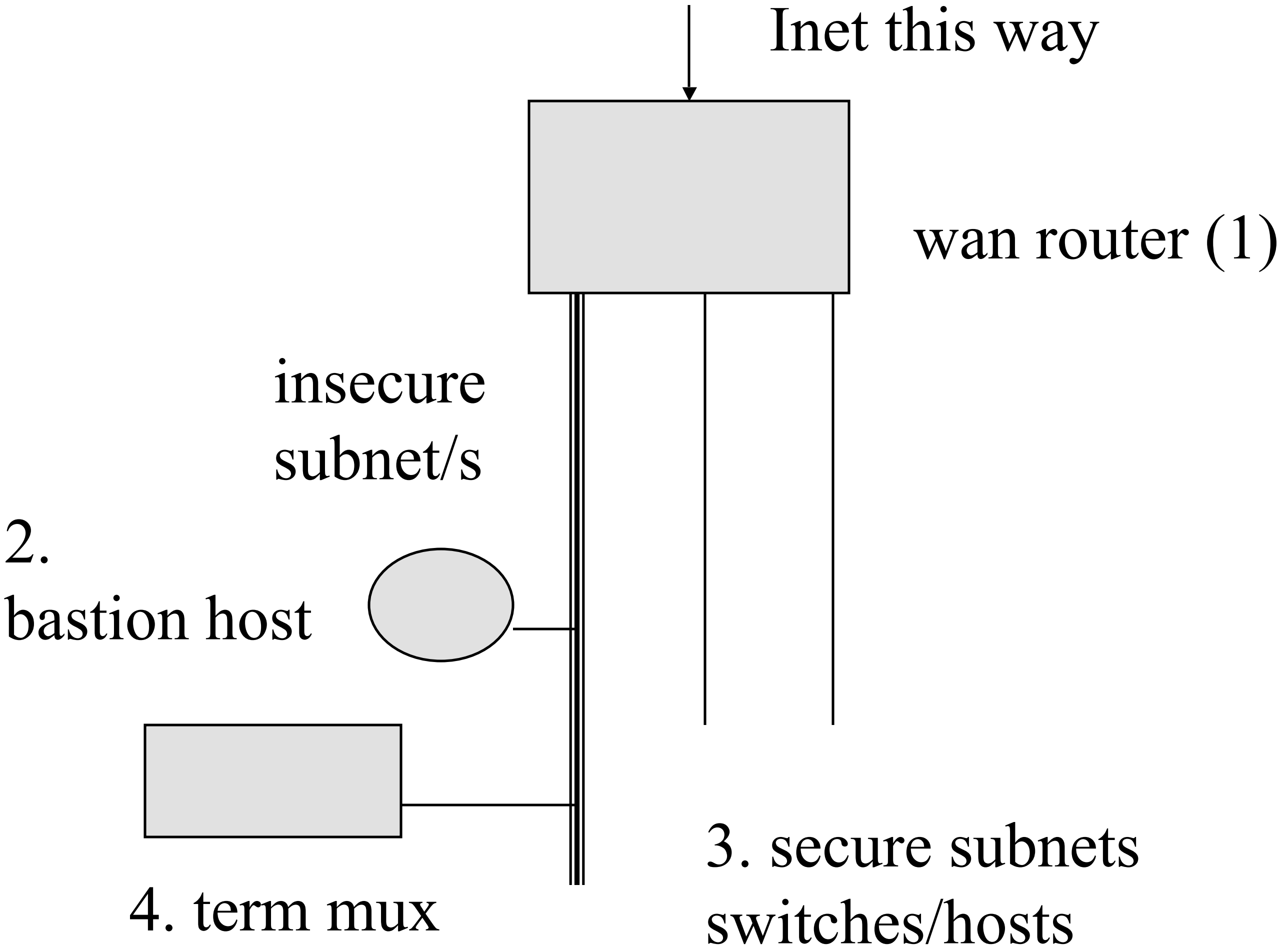


all pkts from net 1 to net 2 subject to
authentication/confidentiality
(and vice versa)

VPNs

- mechanisms extent include:
- IPSEC (we will study it)
- Microsoft PPTP, Cisco L2TP schemes
- Cisco routers have IPSEC now in some versions
- DEC Altavista tunnel is 3rd party software solution for hosts/servers including WNT/UNIX
- can be integrated into firewall rule systems
  - something like: packets from X must use IPSEC ...and either be verified on me or on bastion host Y

possible general enclave design



explained

- WAN router (1) uses ACLs to protect self/bastion host (possible app-gateway or single proxy system/s)
- one totally protected subnet (may not be allowed external access) exists for net console and switches (vlan net 1 ...)
- completely or semi-protected subnets exist for hosts, may have 2nd screening router
- dialup or wireless access point should be designed to be “outside” (possibly same ACLs ...)

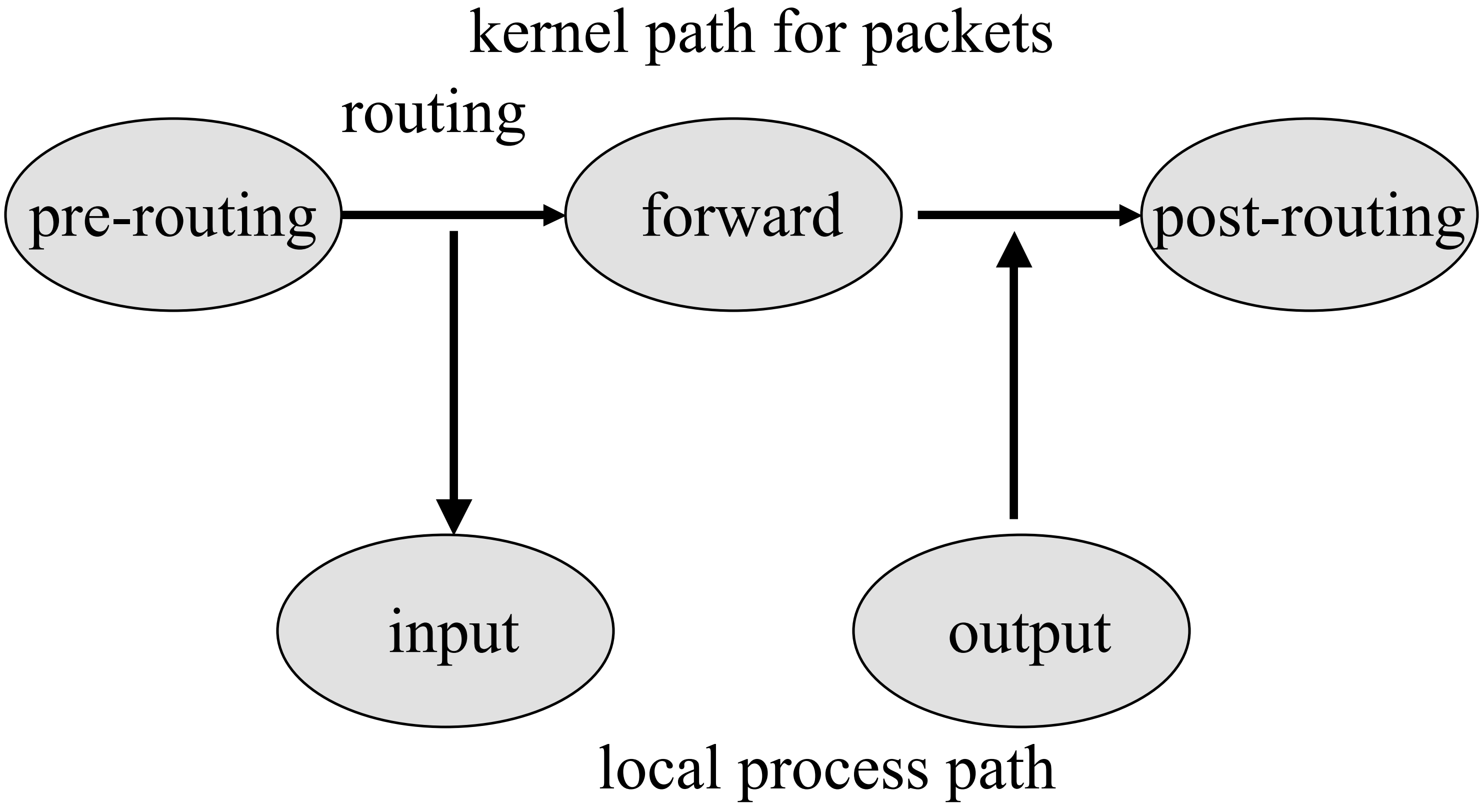
horrible generalization time

- **proxy/application systems are more secure than packet-filter firewalls**
  - can’t do telnet backchannel ...
  - you must protect your infrastructure though
- **packet-filter firewalls are faster**
  - but are they fast enough (you have a shiny new OC-12 to the Internet and a linux host as a firewall) -- oopsie

linux netfilter architecture

- goal is to provide
  - portforward
  - redirection
  - nat
  - filtering
- “netfilter” is the framework
- various form of packet filtering, plus NAT is the outcome

hook overview:



netfilter subsystems

- backwards compatible ipchains
- iptables packet classification system
- nat system
- connection tracking system (used by nat)

Linux iptables

- kernel mechanism with 3 tables and possible kickout to user process
- 3 tables are filter, nat, mangle tables:
  - 1. filter, default, hooks are local in (INPUT), FORWARD, local_out (OUTPUT). filter is for packet filtering (obvious...)
  - 2. NAT, hooks at local out, prerouting, postrouting
  - 3. mangle table (special effects), all 5 hooks now supported

some simple examples

- # iptables -A INPUT -p icmp -j DROP
  - means add an input rule to drop all icmp packets
- # iptables -D INPUT 1
  - would remove that rule
- # iptables -A INPUT -s 10.0.0.0/8 -j ACCEPT
- # iptables -A INPUT -I 3 (rule three) ...
  - rules go into the top by default
- #iptables -A INPUT -p tcp --dport 25 -j DROP (drop SMTP packets)

connection establishment

- can lead to stateful inspection
- -m flag used here (-m state --state <keyword>)
- therefore can allow ftp connection from client back out to server
- can allow udp packet out, expecting udp reply to come back in

notes on useful Linux commands

- netstat -natp - tells you which processes are using which tcp ports
  - # lsof is a pan-UNIX utility for this too
- netstat -naup - UDP version
- iptables-save and iptables-restore used to save/restore entire set of iptables commands
- KDE tool, knetfilter is GUI front-end
  - expansa.sns.it/knetfilter

one more:

- firewall builder tool
- www.fwbuilder.org
  - build firewall rules for different kinds of hosts
  - Cisco PIX/Linux iptables/BSD

IDS overview

- systems exist that look for intrusions which may be defined as
  - known attacks (you got any usoft port 80?)
  - abnormal behavior (e.g., attack not known yet)
- sys admins have looked for “abnormal” behavior for a long time
  - hmmm... I wonder what the process named “worm” does? or “scar_disk” ???

a few examples

- packet analyzers - hooked up to promiscuous mode ethernet ports
  - tcpdump to Internet Flight Recorder or snort
  - or trafshow
  - look for known attacks based on packets matched to filters (snort, IFR)
  - arpwatch
- mrtg oddly enough (or rmon, ourmon)
- log scanning (e.g., tcp wrapper can fit here)
  - automated or not (ps -ax and /var/log/messages)

a few examples

- host based - file watching
  - tripwire considered as good example
  - checksum current files, and save in secure place
  - periodically (every 24 hrs) run again, and compare results
  - what does change mean?
  - what do you do to secure tripwire?
- distributed fault finders, satan, sara, nessus, etc.
  - look for known faults on a local network
    - do you have an old sshd?

some hard questions for these systems

- lots of “false positives”
- may look for PHF (old stuff), and of course,
  - not find new stuff (reactive, not forward thinking)
- distributed and heterogeneous approach is needed
  - you have 30 switches, 5000 hosts, WNT, W98, linux, Solaris, openbsd, macintosh

jails

- emerging open source and commercial NETWORK ACCESS CONTROL world
- may use some combination of ARP/DHCP/DNS and VLANS to put host in jail
- either because it was infected and caught
- or because we assume guilty until innocent

jail #2

- roughly might go like this
- put agent on host
  - agent checks for virus checker
  - agent checks for windows update, old IE
  - agent might watch for anomalies
- server asks agent if host ok
- if not ok, stuck in evil vlan, web surfing results in message: You smell bad, get fixed then come back

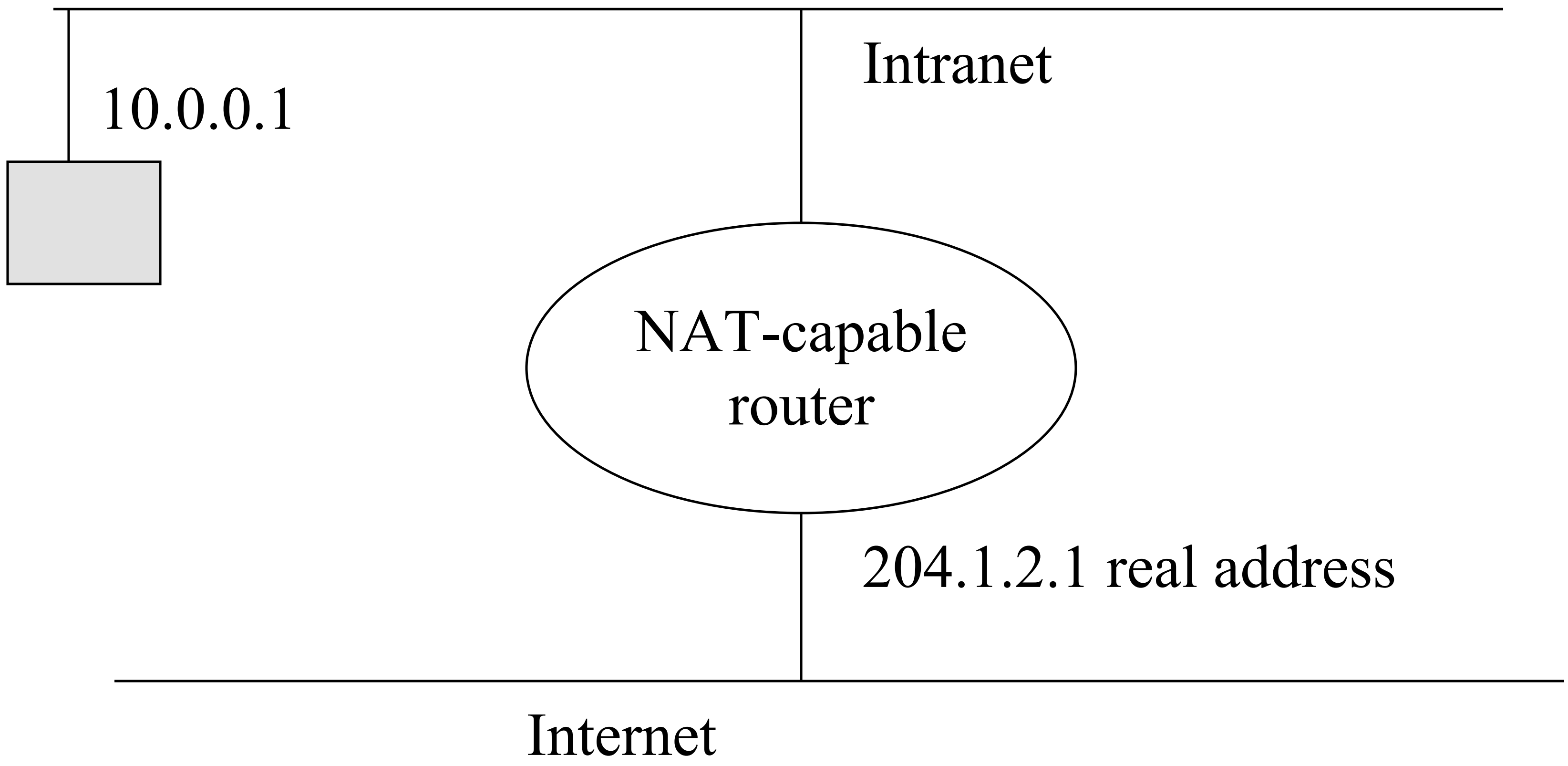
open source version

- www.packetfence.org
- how might this stuff go wrong?
- any questions?

NAT with ports seen as windows firewall

- point is we can connect out
- but they can't connect in (we hope)
- stateful - connection table needed
- packet headed out/in must be rewritten
- NAT by definition breaks end-end
  - breaks IPSEC, Mobile-IP
  - although there is an odd workaround (UDP tunnel)

NAT picture



Intranet

10.0.0.1

NAT-capable router

204.1.2.1 real address

Internet

NAT workings

- consider 10.0.0.1 and 10.0.0.2 want to send a TCP syn packet to 1.1.1.1, 1.1.1.2 at dst port 22
- 10.0.0.1, 1025 -> 1.1.1.1,22 arrives at NAT box
- rewritten to NATIP, free NATportn ->1.1.1.1,22
- 10.0.0.2,1025-> 1.1.1.2,22 becomes NATIP, NATportz->1.1.1.1,22
- this must be transparent to internet boxes
- NAT box maintains 5 tuple NAT tuples and must associate timeout with them
- note L3, L4 header munging, checksum rewrites

final conclusions

- allow all as default is a hard place to be - we know this, we don’t act on it
- security ultimately relies on human trust and human relationships
- defense in depth is good but how much is enough?
- security is not found “in a can” (weak link breaks the chain)
- new attack paradigms will occur ... firewalls will change. IDS in firewall plus anomaly detection - relatively new

in spite of end-to-end hopes

Firewalls will be necessary as long as software has flaws

corollary: principle of isolation is not going away any time soon

Jim Binkley