# Secure Internet Servers/Firewalls with


**Ian F. Darwin**
**http://www.darwinsys.com**
**http://www.openbsd.org**

# What you will learn today:

- How to
  - install, configure and maintain
  - a secure Internet server and/or firewall
  - using software included in OpenBSD.

# Who should be here today:

- Security Consultants who want to configure OpenBSD
- System and Network Administrators with some working knowledge of UNIX network configuration.

# What you will not learn today

- Every last detail (only a few hours)
- UNIX history, basic commands, editing
- Internet history, usage
- Configuring X11 (hint: SuperProbe, XF86Setup / xf86cfg)
- All about UNIX administration
  - See man, FAQ, books...

# Plan for the day:

- 1 About OpenBSD & Security
- 2 OpenBSD System Installation
- 3 Network Services
- 4 Mail Services
- 5 LAN services
- 6 Security services
- 7 Logging Features
- 8 Virtual Private Networks
- 9 Keeping it secure

Break mid-afternoon as per schedule

# 1 -- About OpenBSD & Security

OpenBSD Is

- Mainstream standards-conforming UNIX-like system
- Based on 4.4BSD (25+ years of continuous UNIX evolution)
- Project dedicated to code correctness & system/network security
- Versatile
  - Cryptography, VPN, networking in base
  - Firewall
  - Server
  - Desktop?
- OpenBSD is NOT
  - A Linux clone
  - A SunOS/Solaris clone (but commands close to 4.1)
  - SMP (not a goal at present)

# OpenBSD - Secure by Default

- Goal: Provide safe configuration out of the box
  - Implies: minimal services enabled by default
  - Only network services enabled by default: ssh, daytime/time services, ident
  - Sendmail and comsat (only on localhost)
- Total Code Audit: Multi-year, multi-national, ongoing
- Integrated cryptography (kernel and userland)
- Random number sources used throughout (net, pid, ...)
- Thorough documentation: man, FAQ
  - FAQ is at http://www.openbsd.org/faq/, and on the CD
  - Please learn to RTFM :-)

# OpenBSD is Free Software

- Goal: Must be usable by anybody for anything
  - Even commercial software
- Preferred licensing is standard BSD license
  - GPL acceptable for optional components
  - Unacceptable: "redistribute without modifications" clauses
- BSD versions of standard commands used where possible
- See web site, goals.html

# Code Auditing

- Three levels
  - Kernel
  - User code that ships installed ("base")
  - User code in ports/packages
- Process
  - Initial
  - Ongoing - every change looked at
    - all code viewable on the Internet via CVS
  - Process: Look for bad code, and security bugs fall out in the process
    - See Theo de Raadt's paper (on the web site) for more on auditing.

# Security Basics

- What to protect, from whom
  - External "system crackers"
    - Script kiddies, real hackers, "doorknob rattling"
  - Internal: cracker wannabes, frustrated non-sysadmins, disgruntled employees, paid spies, ...
- Attacks: local, remote...
- Buffer Overflows, ...

# Local Attacks

- Requires an account
- Escalate to system or root
  - due to vulnerability in system software
  - or careless administrator

# Remote Attacks

- May/may not need account
- IP Stack attacks
- Eavesdropping
- Daemon attacks
- RPC
- TCP Session Hijacking

# Denial of Service (DOS) Attack

- Use up some resource to prevent legit users
  - Fill filesystem
  - fork() loop - fill process table
  - Remotely if possible
  - Local DOS almost below our radar: do not give out accounts on firewall
- DDOS: 12 million Monkeys pinging your firewall...
  - from Windows 95 boxes on cable modems.

# Buffer Overflows

- Cracker deliberately overflows a fixed-length buffer, overwriting data or code beyond it with information that changes the behavior of the server
  - Extremely common form of problem - multiple recent IIS attacks
  - Any code allocating and reading into a fixed-size buffer is suspect
  - Particularly if it uses C library gets() or makes other assumptions about line length

# Paranoia is Good

- Password file stealing
  - Old hat - BSD pwdb avoids it,
  - hides passwd encryption from /etc/passwd
  - (master.passwd & pwdb only readable by root)
- Password guessing
  - hide user names (mail rewriting)
  - OpenBSD logs failed logins (by tty/pty)
- Firewall & server machines are not desktops
  - very few services
  - Not X11
- Swap File Encryption
  - enable in /etc/sysctl.conf
- Buy switches, not hubs
  - switch only sends packets to correct line via MAC address snooping
  - hub makes it easier for sysadmin (and cracker) to monitor traffic

# Security Policy

- Must state what is/is not allowed
  - Controls Firewall decisions
  - Tells employees what is/is not OK
- No policy ==> Anything goes
- Need top management backing
  - --> Office Politics
- Based in part on
  - What you are trying to protect
  - Data & Systems Integrity & availability
- Reference: Zwicky, Chapter 25
  - Sonnenreich, p 34
  - Cheswick & Bellovin

# Security Policy on the Firewall

- Firewall policy can be:
  - block everything, then pass exceptions
    - More suitable for high-security (firewall)
  - pass everything, then block exceptions
    - More suitable for high ease-of-use (notebook, home/development computer)
- Firewall may
  - forward permitted packets
    - common, efficient, needs sysctl setting
  - forward no packets, use application-level gateways
    - more overhead, can be more secure if gateways carefully written
    - no direct path for rogue packets
    - Less common!

# Building a Firewall

- Types of firewalls
- OpenBSD supports:
  - packet filter and NAT/redirect
    - pf (since-2.9; ipf before that)
    - ppp/pppd
  - bridge
    - covered later

# Firewall Terminology

- Bridge
  - Machine has 2 interfaces but not IP addresses
  - Originally hardware: OpenBSD has bridge driver
- Router
  - machine has 2 interfaces with IP addresses,
    - makes routing & policy decisions
  - may be unix host with IP forwarding, or dedicated hardware
- Packet Filter
  - Prevent unwanted packets from passing
  - Allow selectively
  - May redirect to inside

# Firewall Terminology II

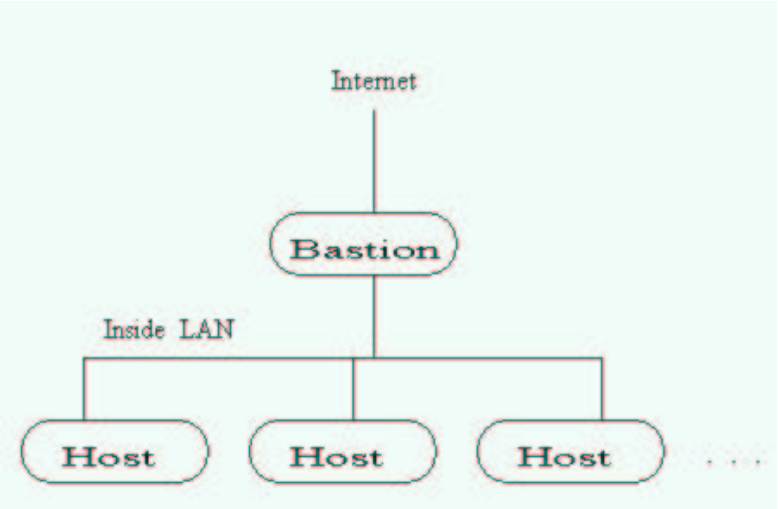
- Proxy (aka application gateway)
  - Can forward around filter
    - Listens on one interface
  - Needed if
    - forwarding off
    - NAT/masquerading for multi-connection protocols (ftp, icq, H323)
- Bastion host
  - Inside filter
  - May store & forward STMP, proxy some services
- Outside Router aka "access router"; Inside router aka "choke"
- Reference: Zwicky Chapters 6 & 11, Sonnenreich, Chap 9

# Inside Router

- Last line of defense
- Between main firewall and inside net
- Dedicated router or OpenBSD box
  - No remote logins
  - No "pc anywhere" access
  - Console access only
- If firewall compromised, this is the only protection against the firewall accessing all inside traffic

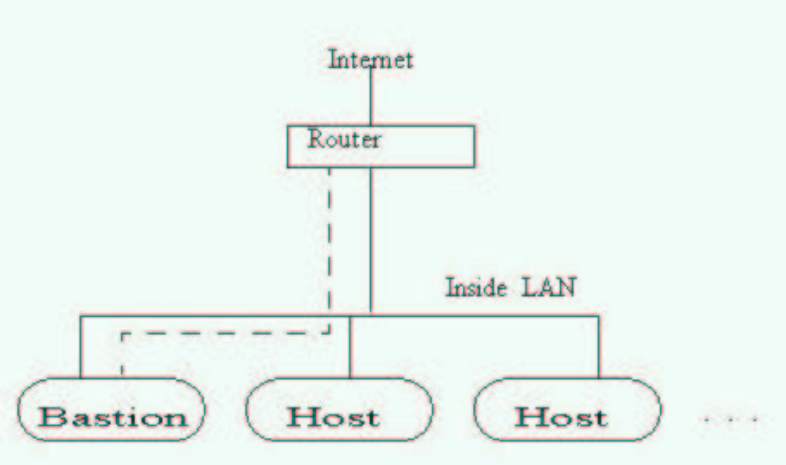
# Firewall Config - Simplest

- So-called "external router" or "dual-homed host"



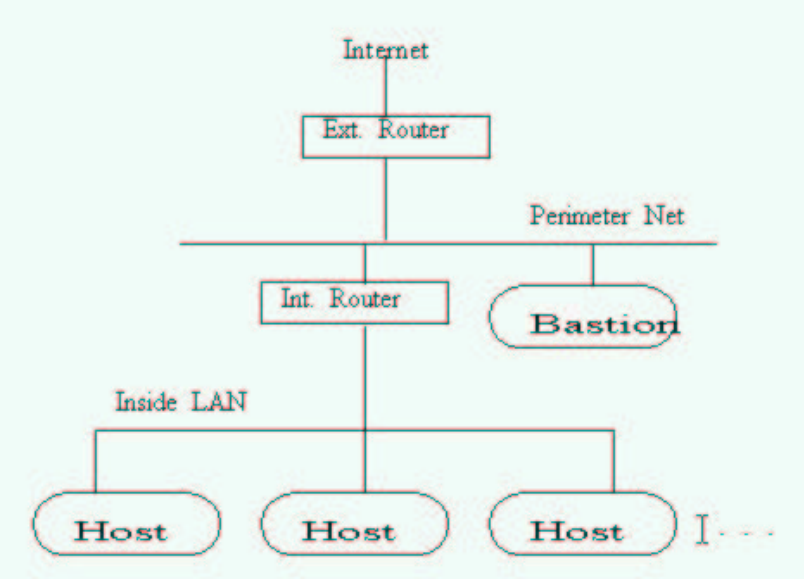
# Firewall Config - Screened Host

- External router allows access to inside bastion
- Bastion makes allowed services available to rest of inside net



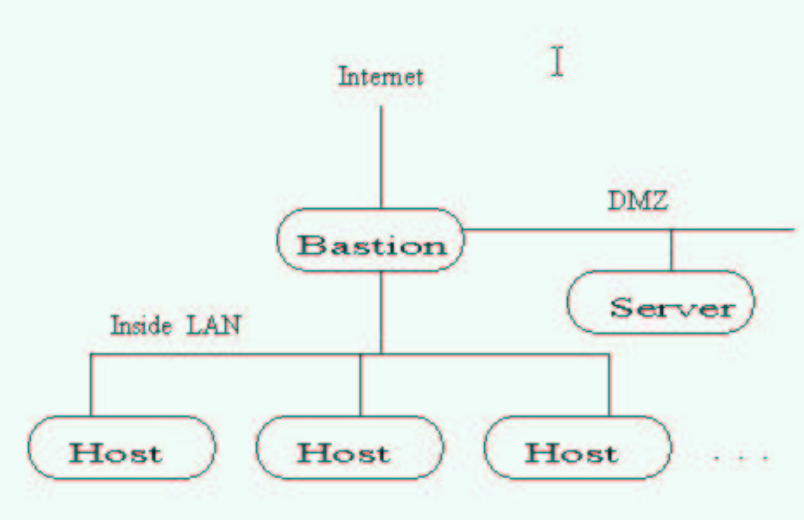

# Firewall Config - Screened Subnet

- Router provides access to subnet



# Firewall Config - "Three-legged firewall"

- Single OpenBSD host does all the work
- Suitable for many situations




# Bridge Configuration

- Allows dedicated hardware address-based routing
- Originally to join network segments.
- Can be used in conjunction with pf to hide inside hosts
- Usage: brconfig
  - Specify MAC addresses, other behavior
  - Also allows filtering (similar rules to pf)
  - brconfig brname rule block/pass etc.
- Example: brconfig bridge0 add rl0 add xl0 up
- man brconfig

# 2 -- OpenBSD System Installation

- Semi-friendly install, non-GUI
- Partitioning
- Selecting software
- PostInstall configs
- Adding software

# Partitioning

- Divide hard disk
- fdisk vs disklabel
- Partitions vs DOS attacks
- Sharing with other OSes

# Selecting software

- OS load in n main pieces
  - boot floppy
  - base, etc, misc, man, comp, x*
- Avoid X on firewall
- Comp package: C/C++, headers, ...

# PostInstall configs

- Read root mail
- Read man afterboot, web site /errata.html
  - /etc/rc.conf
  - /etc/sysctl.conf
  - /etc/inetd.conf
  - /etc/rc.securelevel
  - /etc/rc.local

# /etc/rc.conf

- This file is the main enable/disable file for userland services
- On/Off lines like:
  - named_flags=NO  # for normal use: ""
  - sshd_flags=""  # for normal use: ""
- Settings flags (only if given server enabled) like
  - nfsd_flags="-tun 4"  # Crank the 4 for a busy NFS fileserver
- To keep upgrades simple, can edit /etc/rc.conf.local (read after main file)

# /etc/sysctl.conf

- This file enables/disables kernel features, e.g.,
- #net.inet.ip.forwarding=1  # 1=Permit forwarding (routing) of packets
- #net.inet6.ip6.forwarding=1  # 1=Permit forwarding (routing) of packets
- All lines are commented out to begin with.
- Details in sysctl(3) and sysctl(8)
- Try this: sysctl -a | more

# Other files

- /etc/securelevel
  - BSD kernel has "secure levels"; normal secure level does not allow:
    - time changes
    - loading modules
    - changing immutable files
  - These must be done in this sh script
- /etc/rc.local
  - A sh script run near the end of boot
  - Can start local daemons etc here.
  - Remind yourself to boot up your spouse's computer :-)

# Ian's Favorite PostInstalls

- Change root shell (use vipw)
- Customize dot files in ~root (/root)
- Configure sudo
- Remove unused accounts (uucp)
- "Insecure" console and ttyC* in /etc/ttys
  - ==> Requires root passwd to go single user
- Add "portmap=NO" in /etc/rc.conf.local; comment out rusers and rstatd from /etc/inetd.conf
  - ==> This is fixed in 3.2
- Enable minimal PF
  - Even on inside machine - see examples later
- Add packages...

# Adding your favorite UNIX software I

- /usr/ports (extract ports.tar.gz into /usr)
  - Third-party software ported for you
    - cd category/package
    - "make" downloads source, extracts, patches, builds it
    - "make install" builds package, and installs from it!
  - subdir categories archivers, audio, databases, devel, games, lang, mbone, misc, net, news, plan9, security, shells, sysutils, textproc, www, x11, ...

# Adding UNIX software II

- Packages
  - Ports that are pre-compiled: just pkg_add
    - Built by somebody having done "make package" in port directory
- Package once, install many

# Ports/Packages to know about

- emacs, bash, TeX (teTex), Python, PHP
- mysql, msql, postgresql, BerkeleyDB (newer version than in base)
- spiff, idiff, zap, magicpoint, xbill
- gimp, xv
- enlightenment, windowmaker, full KDE, full GNOME
- Java: JDK (native 1.2; 1.3/1.4 currently linux-compat), Kaffe
- Not ports:
  - Apache, SSH, groff, Perl5 (all built into base)

# Adding UNIX software III - Roll your own port

- Thousands of apps already ported
- Easy to make your own
  - Makefile, pkg/{DESCR,PLIST}
  - See web site /porting.html
- Test like mad, mail to ports@openbsd.org
  - Say what the program does!
  - If good it will get committed

# Add Your Own S/W IV - Emulation

- OpenBSD emulates Linux, other-BSD and UNIX binaries
- Fast kernel implementation (system call switch)
- Needs appropriate userland libraries
- Works: Corel WordPerfect, Netscape Communicator, Applix OfficeWare
  - (Star Office?)
- Worked: Adobe Framemaker for Linux - Beta (expired :-( )
- Difficulties: too-clever install scripts that "know" real paths
- Directory Tree:
  - /emul/linux/{etc,lib,usr,...}
  - /emul/freebsd/{etc,lib,usr,...}

# 3 -- Network Services

- httpd
- ftpd
- rlogin/telnet/SSH
- DNS

# httpd

- Default httpd is Apache 1.3.x, included in OpenBSD base
  - Just enable in rc.conf
- Installs under /var/www/{cgi-bin, conf, htdocs, icons, logs}
  - Change conf/httpd.conf, at least for DocumentRoot :-)
- DSO support most platforms
  - Ports of PHP3, JServ, etc. make DSO
- Other HTTP servers in ports/packages

# httpd and chroot

- Effective with OpenBSD 3.2, Apache HTTPD chroots into its ServerRoot by default
  - CGI must be within ServerRoot
  - May want to link statically, to avoid huge list of shared libs in ServerRoot
  - UserDir (~/public_html) cannot access e.g., /home
  - All DocumentRoot(s) must be under ServerRoot
  - NO files/dirs writable by www/www must exist under ServerRoot
- Can disable with -u, but should not for security

# HTTPD

- Adding SSL/HTTPS
  - Buy or build "certificate"
    - Used for (1) trust/identity (2) encryption
    - For #1, need to pay Verisign/Thawte near-monopoly
    - For #2, can fake it, see scripts/rsa_master on handout floppy
  - Then set http_flags to -DSSL in /etc/rc.conf

# ftpd

- Must enable in rc.conf
- Default fpd is Berkeley FTPD; wuftp, proftpd in ports/packages
- Simpler, less flexible than wuftp
  - But more secure?
- Does use /etc/ftpusers (disallowed) and /etc/ftpchroot (chrooted)

# rlogin/telnet/SSH

- rlogin and telnet are insecure - do not enable
- ssh replaces them
  - scp and sftp replace ftp

# ssh - secure (remote) shell

- OpenSSH maintained by OpenBSD, included in base
- Does not transmit password unencrypted (telnet, ftp, rlogin do)
- RSA/DSA identity for equivalence (avoid .rhosts)
- Client is ssh, usage: ssh [user@]host: [command]
- Also scp client
  - Usage: scp [[user@]host:]file ... [[user@]host:]file_or_dir
  - e.g., same as "cp" but any file can have host (and optional user)
- Ssh can "forward" connections for other TCP services, encrypting them
  - (examples later)

# ssh - other programs

- sshd - daemon, must run on machines that will be logged into
  - one of the few services enabled by default
- ssh-agent - to avoid typing passwords all the time
  - run from our e.g., xdm startup files
- sftpd, sftp-server - FTP-like file transfer program over SSH

# User Management

- BSD uses Berkeley db password database
  - /etc/master.passwd -> /etc/passwd.db
- /etc/passwd is synthetic; doesn't include passwd encryptions
  - Do not edit /etc/passwd; must use vipw
- adduser command to add accounts
  - Interactive
  - Also useradd similar to SVR4 command

# User Management and login.conf

- Recent addition is /etc/login.conf
- Part of "BSD Auth" mechanism, analog of Linux PAM
- Login group (field 5 in master.passwd format) looks up in this printcap-format file
- Can give user groups different login policies, limits, password aging, etc.
- Authentication methods: Krb 4 or 5, password, skey, ActivCard X9.9, CRYPTOCard X9.9, Digital Pathways SecureNet, Generic X9.9, etc.
  - Ports software includes an sysutils/login_ldap, an LDAP tie-in for BSD Auth.
- auth-service lets you provide a custom program to approve/deny logins

# User Management Example - S/Key

- BSD-Auth supports variety of authentication tokens, e.g., S/Key from BellCore
- 1) On a local login or over a trusted port, use "skeyinit jo"
- Enter secret passphrase: enter some words here
- Again secret passphrase: enter some words here
- ID jo skey is otp-md5 99 serv71564
- Next login password: BOUT SAT SEEN ARM STIR VEND
- 2) Generate a list of passwords using "skey -n 100 100 serv71564"
- (Prompts for secret password)
- Keep this list safe!

## User Management Example - S/Key (cont'd)

- 3) Can now login using username:type, e.g.,
- ssh jo:skey@server
- otp-md5 98 servo29818
- S/Key Password:
- $
- - Can press enter to type S/Key password in clear

## User Management - More Security

- Your S/Key "list" makes a lucky find for the hacker!
- Better to use one of the hardware tokens (SNK, ActivCard, CrypoCard)
- Or the "generic" X9.9 program x99token
- See man pages {snk,activ,crypto}{init,adm}

# DNS

- Domain Name Service and Bind
- Base system includes BIND/named 4.x - heavily audited
- Chroot jail, /var/named/
  - dev/ etc/ named-xfer named.boot namedb/db.*
- BIND 9.x in ports/packages, some files new format

# 4 -- Mail Services

- smtpd listener
- sendmail
- postfix? qmail? exim?
- POP/IMAP

# smtpd - SMTP listener

- small, audited mail receiver
- avoids any outside contact with sendmail
  - Spools into a chroot jail
  - Offers rule-based SPAM filtering capability
- smtpfwdd de-spools and gives to real MTA

# smtpd Filtering

- smtpd receives mail and stores it (sendmail -bd replacement)
  - smtpfwdd passes it to sendmail or can relay to another machine
- Pretends not to be OpenBSD ("4.1 SMI" :-)
- Runs in chroot jail /var/spool/smtpd
- ./etc/smtpd_check_rules
- Similar to Sendmail anti-spam but easier to write :-)

# SMTPD Filter Rules

- Patterns in src, from
  - ALL
  - KNOWN|UNKNOWN
  - NS=
  - USER - in from - match in identd
- Examples
  - deny:UNKNOWN:AL::ALL
  - deny:*.spamhaus.com *.junkmail.com:ALL:ALL
  - noto:ALL:ALL:*%*@*:551 Sorry %H (%I), I don’t allow relaying to %T
  - noto:RBL.rbl.maps.vix.com:ALL:ALL:550 Mail from %I in MAPS RBL being blocked, see http%C//maps.vix.com/rbl/
- Reference
  - man smtpd, smtpfwdd
  - Commented examples in /usr/share/smtpd

# sendmail

- 3.2 includes sendmail 8.12.6
- Enable in rc.conf
- Config files in /etc/mail/

# postfix? qmail?

- Alternate MTA programs
- In ports/packages
- /etc/mailer.conf maps commands to programs (i.e., from sendmail to actual MTA)
  - mailq /usr/libexec/sendmail/sendmail

# POP

- popa3d POP implementation in base
- Other POP/IMAP software in ports tree
- Consider shipping over ssh
- Windows box logged into OpenBSD
  - Windows SSH: forward pop3 local to pop3 remote
  - Then invoke pop3 mail reader, tell it server is local
- KMail "pre-command" can do this too
  - sudo ssh -f -x -L 110:localhost:60210 server sleep 60

# 5 -- LAN services

- Interface configuration
- PPP and friends
- Routing, ARP, DHCP
- NIS, NFS
- Samba

# Interface configuration

- Standard unix commands:
  - netstat -i
  - netstat -i -f inet
  - ifconfig to set addresses
  - ifconfig ne3 234.56.78.9
  - ifconfig also has media options
- A few devices have special-purpose programs, e.g., wicontrol

# PPP and friends

- PPP supported by userland ppp or pppd
  - ppp more flexible
  - pppd does more work in kernel
- Either can be used in most cases
- PPPoE supported by ppp(8) + pppoe(8) (in base)

# Routing, ARP, DHCP

- Standard UNIX route command
  - /etc/mygate names default gateway at boot time
- rarpd, bootpd, and dhcp server all included
  - Enable in /etc/rc.conf
- Configure DHCP service (listening on inside interface!) in /etc/dhcpd.conf
- DHClient in base (even on boot floppy!)
- /etc/hostname.xx0 can contain as little as "dhcp"

# XDM

- Don’t install X on firewall
- XDM allows X graphical login; enable in rc.conf
- KDE and GNOME have own GUI logins
  - See ports/packages

# NIS

- NIS (formerly Yellow Pages)
  - Standard implementation: ypclient script sets up
  - Need + line in master.passwd, /etc/group - no nsswitch.conf
  - Beware of serving blowfish passwords to proprietary unixes
- Do NOT allow NIS in/out of your security perimeter

# NFS

- Sun’s Network FileSystem spec; 4.4BSD includes BSD NFS
- Not enabled by default
  - Hard to trust: DO NOT allow in/out of firewall (2049 UDP)
- Server publishes filesystems in /etc/exports
  - Read caveats in exports(8)!

# Samba - the SMB/Netbios server for UNIX

- Samba lets UNIX serve MS-Windows boxes
  - Looks just like a Windows server to PC’s
  - In ports/packages
- Difficult to believe this can be secure
  - Do not allow SMB in/out of your security perimeter
  - Block ports 137, 138, 139

# 6 -- Security services

- sudo
- Packet Filtering
- Kerberos
- ktrace/systrace

# sudo

- Allows root access without password, or with different password
  - File /etc/sudoers controls who can do what
  - Keeps root password out of circulation!
- visudo (as root!) to edit the control file

# Tcpd - simple TCP connection filtering

- Wietse Venema’s "tcp wrappers" package in base
- replace inetd.conf entry with tcpd
  - checks if connection is allowed
  - if so, forwards to real server.
- tcpd is in base system
- See man 8 tcpd

# PF - Packet Filter

- "pf" controls what packets are allowed in/out
- Allows full packet filtering firewall functionality in kernel
- OpenBSD uses pf (packet filter), originally by Daniel Hartmeir
  - Not Darren Reed’s ipf and ipnat like some other BSDs
  - Somewhat compatible rules files, but many new features

# NAT - Network Address Translation

- What Linux calls "IP masquerading": one IP on outside, entire LAN inside
- Controlled by /etc/ipnat.rules and ipnat= line in /etc/rc.conf
  - map ppp0 10.0.0.0/8 -> ppp0/32 portmap tcp/udp 10000:20000
- Intruder may not even know the IP of the inside machines

# Packet Filtering - What to filter

- Obviously depends on your environment and firewall organization
- Can block by protocol (TCP, UDP, ICMP...), and specifics (next page)
- General idea: block any packets except what you want in
- E.g., for Mail, web server:
  - allow SMTP in
  - allow HTTP in
  - allow ICMP so users can ping you

# The pfctl program - general notes

- Must run pfctl -e to enable filtering
  - Done for you by setting "pf=YES" in /etc/rc.conf or /etc/rc.conf.local
  - This setting also causes /etc/pf.conf to be invoked
- Can test without actually changing pf rules with pfctl -n
- Can display rules, state, etc., with -s, e.g., -s nat, -s rules, etc.
  - -s info displays log statistics if logging interface (see below)
- pf does not forward packetes
  - must also enable net.inet.ip.forwarding=1 in /etc/sysctl.conf

# How to Filter - /etc/pf.conf

- This file contains NAT and packet filtering.
- Rules must be in this order: options, scrub, NAT, filter.
  - NAT occurs BEFORE filtering
- N.B. NAT is first match; packet filters are last match,
- Command syntax changes over time
  - Man page pf.conf(5) has a BNF for the parser along with more details

# pfctl - options

- Options control timeouts, logging, limits, optimization, etc.
- Timeouts: interval n frag n
  - How often to purge expired states and fragments, how long to keep packet fragments
- For stateful connections, timeouts for various modifiers, e.g.,
  - tcp.first - time from first packet, if no packets in this time, connection discarded
  - Other parameters for phases of TCP connection, UDP, and "other" - see pf.conf(8)
- Syntax: set timeout { tcp.opening 30, tcp.closing 360 }

# pfctl options (continued)

- Log interface - enables statistics on a per-interface basis
- set loginterface ne3 # set to "none" to disable
- optimization sets general parameters for one of several general types, e.g,
  - default, high-latency, aggressive or conservative
    - aggressive - quickly expire connections to reduce memory
    - conservative - keep connecions that might be still in use
- block-policy
  - sets the default for what to do with blocked packets
  - drop (drop silently) or returnn (RST for TCP, ICMP UNREACH for others)
- Limiting - maximum entries in memory pool
- set limit states 10000 - max # of entries for ’keep state’ rules

# pfctl - Scrub Rules

- Packet reassembly on other OSes can be fooled by using misaligned offsets to sneak bad things past inspection code
  - Or even crash/hack the kernel by using "interesting" offsets/sizes
- Scrub rule causes packet to be entirely re-assembled before other rules are applied
  - A form of sanity/sanitization
  - Only for IPV4 packets - IPV6 fragments are blocked
- Can scrub unconditionally, by fragment cropping, dropping overlap
- scrub in on ne3 all fragment reassemble
  - "all" could be set to src/dest address or protocols

# pfctl - NAT

- NAT rules include nat, binat and rdr
- "nat" is normal NAT (IP masquerading)
  - # my naughty client, using somebody else's real net 144 on the inside
  - # nat anything from 144.19.74 to 204.92.77.100
  - nat on $ext_if from 144.19.74.0/24 to any -> 204.92.77.100
  - nat on $ext_if from any to any -> $ext_if

# pfctl - rdr rules

- "rdr" redirects incoming to another IP and/or port
  - for mapping to e.g., a NATted server
  - # Redirect "V1"'s IPalias, for 80 and 443, to machine .22 inside
  - rdr fxp1 201.31.6.100/32 port { http, https } -> 192.168.20.22

# rdr and FTP firewalling

- Outgoing FTP through a firewall is problematic due to use of multiple ports
- OpenBSD supports an FTP proxy that understands pf
- # translate outgoing ftp control connections to send them to localhost
- # for proxying with ftp-proxy(8) running on port 8081
- rdr on fxp0 proto tcp from any to any port 21 -> 127.0.0.1 port 8021
- Run ftp-proxy from inetd:
- 127.0.0.1:8021 stream tcp nowait root /usr/libexec/ftp-proxy ftp-proxy
- Also need to allowed remapped ports, either by port

# pfctl - Packet Filtration Rules

- To set a default "allow nothing" stance, first rules should be
  - block in all
  - block out all
- Rules syntax:
  - in or out - direction
  - quick - bypass all subsequent rules
  - on interface - limit to this interface (dc0, ne3 - macroizable...
  - address family - inet or inet6
  - proto - tcp, udp, icmp, ipv6-icmp

# filter rules continued

- from src-ip port src-port
- to dst-ip port dst-port
  - Addresses can be hostname, interface name, explicit IP, in CIDR notation
  - Parenthesis around interface name means to reload the IP of the interface if it changes - no explicit reload needed
- port numbers can be explicit, or relational
  - The six obvious relationals = != < <= > >=
  - >< range  <> except-range (both exclusive)
  - port 1024 >< 2048 - actually ports 1025-2047

# filter rule examples

- pass in all # don’t use this!
- pass in proto tcp fom any to any port 25
- block in log on dc0 to port 137
- block in log on dc0 from any to any port 2049 # nfs
- pass in on dc0 proto tcp from any to any port {ssh, smtp, domain}
- # traffic "from" our address should not appear on any other interface
- block in on ! dc0 inet from 200.1.1.0/24 to any
- This last is so useful it has been built-in
- antispoof for dc0 inet
- expands to
- block in on ! dc0 from 200.1.1.1/24 to any
- block in inet from 200.1.1.1 to any

# pfctl and stateful inspection

- rule with "keep state" enables this
- only check initial packets; subsequent packets are "pre-approved"
  - forged packet may have bogus sequence; will be ignored
  - faster (binary lookup)
- TCP: state ("established" or S/A)
  - Syn -- synchronize  A -- Acknowldgement, R -- RST
  - P -- Push  U -- Urgent, ...
  - Packets: 1 A=0, S=1, 2 A=1, S=1 3-n A=1, S=0
- "flags S/SA" says look at S bit out of S|A, ignoring other parts of the TCP flags
- "flags /SA" means S and A must be unset ("none out of S or A");

# Stateful Inspection (cont'd)

- # Allow inside machines to initiate connections to outside
- pass out on $ext_if proto tcp from any to any flags S/SA keep state
- # allow outside machines to initiate connections to SMTP

pass in on $ext_if proto tcp from any to any port 25 flags S/SA keep state

- pass in on dc0 proto tcp from any to any port {ssh, smtp, domain} flags S/SA keep state
- For UDP (stateless protocol!), keep state matches only host address and port
- Can use "modulate state" which also randomizes the sequence numbers - for dealing with other IP stacks that give predictable TCP sequences

# User and Group Filtering

- Can block or pass TCP/UDP by user (EUID/EGID) when socket created
  - block out proto {tcp,udp} all
  - pass out proto {tcp, udp} all user { < 100, ian, geoff} keep state
  - pass out proto tcp port 25 user { > 0, unknown }
- Example: FTP proxy runs as user "proxy"; enable remapped data ports
  - pass in on dc0 proto tcp from any to dc0 user proxy keep state

# authpf - per-user PF rules

- User shell for firewall: changes rules when you login, undoes it when you log out
  - Per-user config files
  - SSH Login
  - Begin and end are logged via syslog
- Why:
  - Let users update files in DMZ: allow ftp from inside to web server only when logged in
  - Allow inside users to access the outside (or vice versa)
  - Allow outside users selective access to inside
    - In conjunction with stronly authenticated login

# authpf (continued)

- Rules
  - Same format as normal, but defines user_ip macro
    - In /etc/authpf/users/USER_NAME/authpf.rules
    - If not found, /etc/authpf/authpf.rules (required file) used
- Flexible configuration
  - man 8 authpf for more details

# pfctl - macros

- Good to define interface name in one place
  - many rules required interface name: ne3, dc0
  - This gives only one place to change
- Usage:
  - ext_if=dc0
  - int_if=ne3
  - scrub in on $ext_if all fragment drop-ovl
- Also for IP addresses
  - remote_lan = "123.45.6.0"

# pfctl macros - dynamic

- What about notebook users? Sometimes on dc0 and sometimes on wi0?
- No "if" logic in pf.onf
- No -D option to pre-define ext_if
- Can pipe into pfctl, so use
  - sed ’s/EXTERNAL_IF/$if/’ /etc/pf.conf | pf -f -

# Other Filtering Mechanisms

- Pppd program offers simple filtering:
  - Similar syntax to tcpdump expressions
  - pass-filter "port != smtp" inbound
- User-level ppp program has filtering rules
  - Will also do NAT
  - Has in/out filter for security, and dial/alive filters for dialing
    - set filter in  0 permit tcp dst eq 113
    - set filter out 0 permit tcp src eq 113
    - set filter in  1 permit tcp src eq 25 estab
    - set filter out 1 permit tcp dst eq 25

# Kerberos

- MIT’s authentication scheme: Kerberos authentication for networking services
- E.g., "fixes" telnet, r* and other protocols by using Kerberos auth
  - ensures user is authenticated
  - prevents cleartext passwords
- Common on inside networks
- Kerberos IV implementation included in base system
  - From KTH in Sweden, not MIT implementation due to US export rules
- Kerberos V implementation based on KTH "Heimdal" in base
  - see "info heimdal" and /etc/kerberosV/README

# NTP

- Network Time Protocol
  - Keeps internet machines time synchronized
  - Security...
- Client support in rdate -n in base
  - Userland code (ntpd, ntpdate, ...) in ports/package net/ntp

# ktrace

- A standard kernel system call trace mechanism
- By itself, lets you see what a program under trace is doing
- Very verbose:
  - $ ktrace date
  - Wed Jan  6 22:15:31 EST 2004
  - $ kdump | wc -l
    - 125

# New: systrace: a system call filter

- Run it with -A to generate profile of what a command normally does
- Then run with -a to ensure the command does not do anything it didn’t do before!
- Can prevent a compromised program from accessing files it shouldn’t
  - Since these won’t be in the systrace policy

# Example systrace

- $ systrace -A date
- $ more ~/.systrace/bin_date # date is in /bin
  - Policy: /bin/date, Emulation: native
    - native-__sysctl: permit
    - native-fsread: filename eq "/<non-existent filename>: /etc/malloc.conf" then permit
    - native-issetugid: permit
    - native-mmap: permit
    - native-break: permit
    - native-mprotect: permit
    - native-gettimeofday: permit
    - native-fsread: filename eq "/usr/share/zoneinfo/Canada/Eastern" then permit
    - native-read: permit
    - native-close: permit
    - native-fstat: permit
    - native-ioctl: permit
    - native-write: permit
    - native-munmap: permit
    - native-exit: permit

# Example Systrace continued

- For a shortened example, using "date" instead of a network server
  - (just so it fits in slide format)
  - Remove the last line, denying "exit", run command under systrace
- $ systrace -a date
- Wed Jan  6 22:20:35 EST 2004
- Memory fault (core dumped)
- $ tail -1 /var/log/messages
- Jan  6 22:20:35 daroad systrace: deny user: ian, prog: /bin/date, pid: 3288(0)[0], policy: /bin/date, filters: 14, syscall: native-exit(1), args: 4
- Result: Aggressing user sees memory fault, thinks his attack crashed the program

# Systrace - privilege escalation

- Neat feature: privilege escalation lets you run individual system calls as setuid or setgid
- Details: systrace(1) for usage; systrace(4) describes underlying kernel support
- Example: let unprivileged Tomcat bind port 80 as root
- native-bind: sockaddr eq "inet-[0.0.0.0]:80" then permit as root
- Systrace must be run as root, of course
  - And run with -c uid:gid to say who to run as
- Useful for ISPs to constrain what files virtual-hosted web server scripts (or Servlets in the case of Tomcat) have access to.

# 7 -- Logging Features

- Need logging to know who's doing what
- syslog and OpenBSD
- IP logging
- Test tools & IDS

# Syslogd

- chroot jail
- no UDP by default (DOS attack); must filter if enabled
- Multiple logs
  - newsyslog.conf controls secrecy of certain logs

# pflogd

- packet filter logging mechanism
  - reads from packets logged by pf into /dev/pflog0
  - writes to a logfile e.g. /var/log/pflog in binary tcpdump(8) format
  - Just use tcpdump to format them
- GOOD PLACE TO PAY ATTENTION if you turned on reasonable logging
- Takes part in log rotation via newsyslogd

# Testing Tools

- Tools to simulate an attack
  - "Morally neutral" (used for good and evil)
- tcpdump (in OpenBSD base system)
- netstat - standard UNIX tool, traditional syntax
- nc/netcat - in OpenBSD base system
- nmap - gather information on a site
  - in ports/packages
- nessus - detailed vulnerability scanner
- Others: see ports/net, ports/security

# Intrusion Detection

- Want to know real-time of attacks
  - Probes (nmap used by bad guys)
  - Attacks
- Intrusion Detection Software (IDS)
  - NFR - Network Flight Recorder
  - "Snort"
  - Both are in ports/packages

# Built-In Intrusion Detection?

- daily insecurity report
  - changed permissions
  - important file changes
  - device & setuid changes
- i.e., most of "tripwire" functionality is in OpenBSD base
- See Also: FreeBSD Forensics Using Ports talk tomorrow

# 8 -- Virtual Private Networks

- What & Why
- static setup
- photurisd
- isakmpd
- Conversing with the dark side

# What & Why

- A routing between two of your sites, over networks you don’t control
  - Behaves like point-to-point link
  - Encrypted for security
  - Using IPsec protocol
- Requires secret keys exchanged between both ends

# Faking it

- Use ssh to forward various protocols
- Not really a VPN, but very easy
  - Host-to-host, not to network
- This is what some books consider a VPN to be :-)
- Useful for e.g., forwarding a service or two over an encrypted tunnel
- Use -L and/or -R on UNIX SSH to forward services.
- My smtunnel script sets up to forward SMTP from desktop machine to server:
  - sudo ssh -f -x -L 25:localhost:25 ian@server sleep 60

# PPTP?

- PPTP is an outgrowth of PPP
  - encrypts ppp packets
  - encapsulates using gre driver
- "poptop" server in ports tree
- IPSec is more secure

# IPSec Protocols

- IPSec (IP Security) consists of three protocols
  - AH (authentication header)
    - verifies header: confirm message validity, incl. src and dest
  - ESP (encapsulating security payload)
    - encrypts data
  - ISAKMP (SA Key Management Protocol)
    - Framework for key exchange, needed by AH and ESP
    - IKE most common, also "photuris" and manual key exchange
- Terminology
  - SPI - security parameter index, a "conversation number"
  - SA - security association: (SPI, dest IP, and AH/ESP)
  - Flow - data transfer path

# VPN IPSec Basic Steps

- Enable protocols in /etc/sysctl.conf
  - net.inet.ip.forwarding=1 net.inet.esp.enable=1
  - net.inet.ah.enable=1
- Choose a key exchange method
  - manual, photuris, or IKE
- Either
  - Create a "security association (SA)" for each node
  - Create the IPSec "traffic flows"
- Or
  - Configure and start isakmpd
- Configure firewall rules
- Next few pages give details, then example

# Manual key setup

- ipsecadm creates SA’s ipsecadm creates flows
- See /usr/share/ipsec/rc.vpn for online example
- See handout/scripts/vpnstart for another

# Photurisd key exchange

- Designed by Phil Karn and William Simpson
  - They consider IKE flawed
- OpenBSD developers made first "photuris" implementation
- Described in photurisd.8
- Sample file /usr/share/ipsec/photuris.startup

# ISAKMP (Oakley, IKE) key exchange

- OpenBSD developers wrote own implementation
- Documented in isakmpd.8
- Config and sample in isakmpd.conf(5)
- Requirements:
  - kernel with options CRYPTO and IPSEC, and pseudo-device enc
  - enable AH and ESP with sysctl (uncomment lines in sysctl.conf)

# VPN Example using isakmpd

- 1) set up isakmpd.conf files for both machines
  - 55 lines long; see isakmpd.conf.{a,b} in handout
  - must be mode 600 (or 400)
- 2) set up isakmpd.policy files (same on both machines)
  - Keynote-version: 2
  - Authorizer: "POLICY"
  - Conditions: app_domain == "IPsec policy" &&
  - esp_present == "yes" &&
  - esp_enc_alg != "null" -> "true";
- 3) Configure firewall rules

# Firewall Rules for VPN Example - Machine A

- gatewA = "192.168.1.254/32"
- gatewB = "192.168.2.1/32"
- netA = "10.0.50.0/24"
- netB = "10.0.99.0/24"
- ext_if = ne3

- # default deny
- block in log on { enc0, $ext_if } all
- block out log on { enc0, $ext_if } all

- # Passing in encrypted traffic from security gateways
- pass in proto esp from $gatewB to $gatewA

# Isakmpd startup

- Start as root: /sbin/isakmpd
- Debugging: isakmpd -d -DA=99 # foreground, maximally verbose
  - isakmpd -l file - logs packets in tcpdump format.
- Program to spy on messages between isakmpd and kernel, analogous to tcpdump but for PF_KEY traffic, reportedly at

  http://pobox.com/~listjunkie/keydump.tar.gz
- See also VPN Using *BSD talk by Eilko Bos; OpenBSD server, FreeBSD roaming clients - full details

# Conversing with the dark side

- Windows machines can talk to OpenBSD VPN
  - Must use isakmpd (not photuris)
  - Some restrictions/limitations apply
  - See Markus Friedl's page
  - http://wwwcip.informatik.uni-erlangen.de/~msfriedl/ipsec-win2k/

# 9 -- Keeping It Secure

- System updates
- If it ain't broke, don't break it?
  - Do watch security-announce list at bare minimum (more on lists below)
- How-to?
  - FTP snapshots, install using boot floppy "upgrade"
    - cheat and untar (see my quickupgrate script)
  - Buy new CD's
    - Easiest - updated every 6 months
- CVS, anonCVS
  - Update entire source tree; build & boot kernel; make build...

# Building OpenBSD Kernel

- edit config, config, make, cp, reboot
- Config file
  - /sys/arch/{i386,sparc,...}/conf/ file SYSTEMNAME
  - GENERIC includes most everything
- config SYSTEMNAME; cd ../compile/SYSTEMNAME; make clean depend bsd
- mv /bsd /obsd; mv bsd /; reboot

# Building the rest of OpenBSD

- Once the new kernel is booted:
- cd /usr/src
- sudo make obj && sudo make build
  - N.B. This includes "make install", updating the running system!
  - Otherwise read and understand the Makefile

# Don’t break it

Don’t tinker, nor let others (few root)

- Don’t do development on firewall
- Test first pf/nat testing

# This Week

- FreeBSD VPN Case Study - Sat. 12:00
  - Uses OpenBSD as its server!
- Performance Tuning OpenBSD - Sat. 4:00
  - Philip Buhler & & Henning Brauer
- Authentication in FreeBSD 5 - Sat. 4:45
- Other OpenBSD developers are here - talk to us!

# Learn More from Books

- Design of 4.4BSD Operating System
  - McKusick, Bostic, Karels, Quarterman
  - Karels is the keynote speaker tomorrow!
- Firewalls with Linux and OpenBSD
  - Sonnenreich & Yates (2e? 1e refers to older ipf)
- Building Internet Firewalls, 2e
  - Zwicky, Chapman, et al, O'Reilly.
- Hacking Exposed (various editions)
  - McClure, Schambray, Kurtz
- UNIX System Administration, 3e
  - Nemeth et al.
- Internet Firewalls book
  - Cheswick & Bellovin - classic, bit dated, 2e in preparation
- See http://www.openbsd.org/books.html

# Help Online

- OpenBSD Site http://www.openbsd.org/
- FAQ's http://www.openbsd.org/faq/
- Man pages http://www.openbsd.org/cgi-bin/man
- User Groups http://www.openbsd.org/groups.html
- Consultants http://www.openbsd.org/support.html
- OpenBSD Journal http://www.deadly.org/
- Daemon News http://daily.daemonnews.org

# Mailing Lists

- Mailing Lists http://www.openbsd.org/mail.html
- Main lists:
  - misc - newbie, installation, device questions
  - ports - all about ports/packages
  - tech - only for hard technical questions
  - source-changes: every single commit (volume warning!)
- Lurk a month before posting
- Search the archives (see mail.html) before posting
- Read all of mail.html before posting
- Never say "please reply to me directly..."
  - If you are too busy to read the mailing lists, we have consultants for hire (support.html) who can read it to you :-)

# The One Marketing Slide

- We want OpenBSD to remain
  - free
  - non-commercial
- To do this we need money
- Please don’t buy our CD’s unless you want to :-)
  - Write code.
  - Write documents. Translate documents!
  - Donate $ or equipment (see want.html)
  - Buy CD’s, T-Shirts (via the web or here at the show)
- Thank you!

# Finale

- Questions and Answers

- Ian Darwin
  - http://www.darwinsys.com/
- Example files available (next week) from
  - http://www.darwinsys.com/training/obsd-firewalls

# About The Slides

- Presentation written by Ian F. Darwin
- Notes originally entered into Lotus Freelance
  - Quickly exported to plain text!
- This presentation edited with vi on OpenBSD, and delivered with the free software MagicPoint.

-- The End --