

Lecture 9: Unix Review/Hardening

- Today's topics
 - Brief History of Unix
 - Commonly Used Unix Distributions
 - Users and file permissions
 - The system boot process
 - Logging in
 - Inetd
 - Network Daemon Processes
 - Local/Remote vulnerabilities
 - Countermeasures

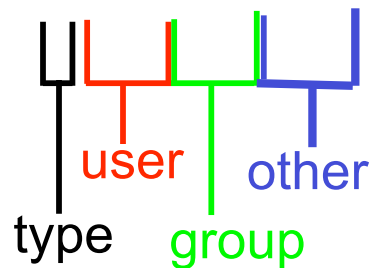
In The Beginning.....

- The UNIX “epoch” is January 1st 1970 (technically it was created in 1969.)
- System V Unix AT&T (Bell Labs)
 - System V Rev. 4 is the latest.
- BSD Unix UC Berkeley (1979) Released in 1983.
- Unix is a multi-user OS written mostly in C.

Unix Users

- “Super User” Root can view and or modify any file or process on the system.
- “users” can only view or modify files for which they have permission. They can only modify processes that they own.
- File permissions:

`-rwxrwxrwx 1 jake staff 686 Feb 20 11:56 myfile.txt`



SUID - Set user id

- When an executable is SUID, it runs with the privileges of the owner of the executable.
- Some system binaries are SUID **root** to allow “average” users the ability to perform necessary tasks.
- When an executable is SGID, it runs with the group designation specified on the file.
- For example the login program, which runs when a user logs in, is SUID root so that the process can read the shadow file.
(The shadow file is only readable by root)

```
r-sr-xr-x 1 root bin 29292 Oct 6 1998 login
```

- The SUID bit is shown as an “s” in the user’s execute portion of the file permissions when using the command “ls -al”.

Booting the system

- The process of booting a system is slightly different between BSD and System V UNIX.
- We will be focusing on System V process of booting since it is becoming the Linux standard in most distributions.

Booting Linux (simplified)

- At the first step in the boot process the system BIOS checks the system hardware.
- The BIOS then reads the instructions on the MBR (Master Boot Record) stored on the first sector of the hard disk.
- With a Linux system, the MBR typically loads the LILO (Linux Loader) program. But GRUB is becoming more popular...
- LILO proceeds to boot the kernel which initializes memory, loads device drivers, then launches the “init” process.

Booting Linux (simplified)

- The “init” process is the mother of all processes, with a process ID of 1.
- Init is responsible for starting all the system processes at boot time, and re-spawning some if they terminate while the system is running.
- The file [/etc/inittab](#) configures the behavior of the init process, and specifies the default run level.
- See the man pages for init and inittab for more details.

Kicking it old school (Sys V style)

- System V UNIX introduced the concept of runlevels.
- Different processes are started at each run level.
- Run levels
 - 0 - Halted
 - 1 - Single User
 - 2 - Linux: Not used, **SYSV: Multi-user without NFS**
 - 3 - Multi-user with NFS
 - 4 - Not Used
 - 5 - Linux: Full Multi-user with X Login, **SYSV: Halt, power off**
 - 6 - Reboot

Sys V init

- The inittab is configured to instruct init to run `/etc/rc.sysinit` first.
- It is a shell script which initializes the system, mounts local file systems and starts network parameters.
- Init then launches `/etc/rc.d/rcx` which parses the directories `/etc/rc.d/rcx.d` (where `x` is the runlevel `0-6`).
 - SysV: Init starts with `x=1` and increments till it reaches the default run level specified in inittab
 - Linux: Init uses `x=default run level` and parses `/etc/rc.d/rcx.d`

Sys V init

- The directories `/etc/rc.d/rcx.d` contain shell scripts which start and stop services and/or daemons. An example of a script name is `S99sshd`.
- As the directories are parsed, `/etc/rc.d/rc` executes the scripts in numeric and alphabetic order with the argument of `start`.
(E.g., `S99sshd start`)
- This process is continued until the last script of the default run level is executed, then `init` spawns a `getty` process to present the login prompt.
- Alternatively at shutdown, `init` parses the same directories as it decrements run levels with the argument of `stop`.

Logging in: The Password file

- The file `/etc/passwd` contains information about valid user's of the UNIX system.
- `/etc/passwd` has several fields separated by a ':'
 - The fields are
Username:Encrypted Password:UserID:GroupID:GCOS(Comment)
:HomeDirectory:login shell
 - Example:
`jake:x:9730:10:Jake:/home/jake:/usr/bin/bash`
- Notice the `x` in the Encrypted Password field. This indicates that shadow passwords are being used.

Logging in: The shadow file

- All UNIX systems have `/etc/passwd`, but not all have `/etc/shadow`.
- `/etc/shadow` contains a corresponding user entry from `/etc/passwd`, and additional fields separated by a `:`
- The fields are:
username:encryptedpassword:lastchg:min:max:warn:inactive:expire:
- Example:
`jake:ABC123DEF:::::::::`

Logging in: Login process (brief)

- The **login** process handles all user logins from terminal sessions. The log on each user must present their username and password.
- The **login** process verifies that an entry in `/etc/passwd` exists for the user.
- The **login** process then, using a “salt”, encrypts the entered password and compares it to the encrypted password in `/etc/passwd` or `/etc/shadow`.
(if shadowed passwords are used.)
- If the encrypted passwords match, you’re allowed in!

So, about that crypt function

- The crypt function is a one-way DES variant.
- Take a 64-bit block of \0 (NUL) characters.
- Create a 56-bit key by extracting the first 7-bits of each password character.
- Key expansion permutation is a function of the salt
 - 2^{12} possible permutations (salt is a 12-bit value)
 - To circumvent hardware passwd crackers.
- Run DES algorithm 25 times
- Convert final 64-bit block output to a 64-char alphabet
 - [A-Za-z./]
 - One-way (maps multiple different inputs to a single output)
- Can't be decrypted (see above)

Inetd

- Inetd is often referred to as the “Internet super server” .
- The inetd process oversees several other network daemons that provide specific services such as ftp, telnet, rlogin etc...
- Inetd’s functions are defined in its configuration file [/etc/inetd.conf](#):

For example: [Telnet](#)’s inetd.conf entry:

```
telnet stream tcp nowait root /usr/sbin/tcpd in.telnetd
```

The keyword “telnet” now needs a corresponding entry in [/etc/services](#) to tell the system on which port it’s listening:

```
telnet          23/tcp
```

Inetd

```
/etc/inetd.conf:
```

```
telnet stream tcp nowait root /usr/sbin/tcpd in.telnetd
```

```
/etc/services:
```

```
telnet                23/tcp
```

- In this example, `inetd` has been configured to handle incoming telnet requests.
- `Inetd` listens on tcp port 23 for incoming connections.
- When a valid connection is received, `inetd` then spawns a telnet process (`in.telnetd`) to handle the request.
- Why all this indirection?
 - `Inetd` cuts down on the number of processes running on the system.
 - Processes spawned by `inetd` will only run as long as they are needed.

Xinetd

- xinetd is a secure replacement for inetd.
 - performs the same function as inetd
 - can do access control on all services
 - Don't need the /etc/services indirection

```
/etc/xinetd.conf:  
service telnet  
{  
    flags          = REUSE  NAMEINARGS  
    protocol       = tcp  
    socket_type    = stream  
    wait           = no  
    user           = root  
    server         = /usr/sbin/tcpd  
    server_args    = /usr/sbin/in.telnetd  
}
```

Network Daemon Processes

- Some Internet service daemons are spawned by `inetd`, and others run as stand-alone daemons that listen on a particular port.
- Stand-alone daemons such as `httpd` fork() a child process to handle each connection.
- A default installation of most UNIX systems install and start several daemons by default such as
 - `portmap/rpcbind`
 - `nfsd`
 - `httpd`
 - `lpd`

Process Control

- The `ps` (process status) command can be used to view the current running processes
- The `kill` command can be used to send a signal to a process to terminate, restart or reconfigure the process.
 - See the `kill` and `signal` man pages for more details
- Commonly used signals:
 - `1 (HUP)` - Tells a process to re-read its configuration file.
 - `15 (TERM)` - Tells a process to terminate.
 - `9 (KILL)` - Forces a process to die when it doesn't respond to a TERM.

Find running processes with ps

- Output of “ps -aux”

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	2.2	0.8	1324	536	?	S	21:20	0:05	init [5]
root	2	0.0	0.0	0	0	?	SW	21:20	0:00	[kflushd]
root	3	0.0	0.0	0	0	?	SW	21:20	0:00	[kupdate]
root	4	0.0	0.0	0	0	?	SW	21:20	0:00	[kpiod]
root	5	0.0	0.0	0	0	?	SW	21:20	0:00	[kswapd]
root	6	0.0	0.0	0	0	?	SW<	21:20	0:00	[mdrecoveryd]
root	334	0.2	1.3	1660	844	?	S	21:21	0:00	syslogd -m 0
root	344	0.0	1.3	1624	820	?	S	21:21	0:00	klogd
rpc	359	0.0	0.9	1468	576	?	S	21:21	0:00	portmap
root	375	0.0	0.0	0	0	?	SW	21:21	0:00	[lockd]
root	376	0.0	0.0	0	0	?	SW	21:21	0:00	[rpciod]
rpcuser	386	0.0	1.3	1572	824	?	S	21:21	0:00	rpc.statd
root	401	0.0	0.8	1308	524	?	S	21:21	0:00	/usr/sbin/apmd -p
daemon	455	0.0	0.9	1356	576	?	S	21:21	0:00	/usr/sbin/atd

Finding open ports

- Network daemons are bound to a particular port and listen for connection requests.
- A listing of service-port mappings are found in the file `/etc/services` .
- There are a few utilities in UNIX which allow a user to view which ports are “open”.
 - netstat
 - lsof
 - nmap

Using netstat to find open ports

- `netstat --inet --all` (Linux, show network ports)
- `netstat -f inet` (Solaris, Digital Unix)

Active Internet connections (servers and established)

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State
tcp	0	0	*:www	*:*	LISTEN
tcp	0	0	*:1024	*:*	LISTEN
tcp	0	0	*:sunrpc	*:*	LISTEN
udp	0	96	laptop.oit:1034	nic.umass.edu:domain	
udp	0	0	laptop.oit:1034	nic.umass.edu:domain	
udp	0	0	*:1027	*:*	
udp	0	0	*:986	*:*	
udp	0	0	*:1026	*:*	
udp	0	0	*:sunrpc	*:*	
udp	0	0	*:1025	*:*	
udp	0	0	*:sunrpc	*:*	
udp	0	0	*:1025	*:*	
udp	0	0	*:1024	*:*	

Using `lsof` to find open ports

- The utility `lsof` (“list open files”) can list network ports along with the listening process.
- `lsof -i tcp` (list all tcp ports)

COMMAND	PID	USER	FD	TYPE	DEVICE	SIZE	NODE	NAME
portmap	359	root	4u	IPv4	423		TCP	*:sunrpc (LISTEN)
rpc.statd	386	root	7u	IPv4	463		TCP	*:1024 (LISTEN)
httpd	1021	root	16u	IPv4	16531		TCP	*:www (LISTEN)
httpd	1022	root	16u	IPv4	16531		TCP	*:www (LISTEN)
httpd	1023	root	16u	IPv4	16531		TCP	*:www (LISTEN)
httpd	1024	root	16u	IPv4	16531		TCP	*:www (LISTEN)
httpd	1025	root	16u	IPv4	16531		TCP	*:www (LISTEN)
httpd	1026	root	16u	IPv4	16531		TCP	*:www (LISTEN)

Using nmap to find open ports

- nmap is an extremely versatile port scanning tool.
- <http://www.insecure.org/nmap>
- Using the command “`nmap -sT laptop.oit.umass.edu`”

Starting nmap V. 2.53 by fyodor@insecure.org

Interesting ports on laptop.oit.umass.edu (128.119.xxx.xxx):

(The 1519 ports scanned but not shown below are in state: closed)

Port	State	Service
80/tcp	open	http
111/tcp	open	sunrpc
1024/tcp	open	kdm
6000/tcp	open	X11

Unix Security

- Local access
 - Vulnerabilities, exposures, and countermeasures
 - Auditing your system
- Remote access
 - Vulnerabilities and exposures, countermeasures
 - Auditing your system

Local System V&Es

- **Physical Compromise Countermeasures**
 - Secure the host and console in a cabinet or locked room.
 - Password protect console and BIOS
 - Disable booting from removable media
 - Password protect single user mode
- **Buffer Overflows Countermeasures**
 - Disable the execution of code on the stack, via OS patch or kernel parameters.
 - Use safe programming practices:
 - `strncpy()` instead of `strcpy()`
 - Remove SUID bit from executables that aren't required for normal system operation.
 - Apply vendor OS and application patches.

Trojaned LKM Countermeasures

- LKMs are **loadable kernel modules**: they're code (usually device drivers) that get loaded by the kernel when access to the particular device is needed.
- LKMs can be trojaned to modify the kernel in malicious ways; to mask a system cracker's presence for example.
- Countermeasure: compile a monolithic kernel with all device drivers included. (Only available in some OSes.)
- <http://www.redhat.com/docs/manuals/linux/RHL-7.3-Manual/custom-guide/s1-custom-kernel-monolithic.html>

Auditing your system (from inside)

- Steps you need to take (we'll go over each)
 - Verify most recent patches are applied to OS and applications
 - Disable unused services and daemons
 - Address user account and password issues
 - Protect from unauthorized remote access
 - Other general types of checks

Auditing - Verify patches

- Verify that you have installed the most recent OS and application patches
- Most Operating System vendors and applications have web pages and mailing lists to announce product updates.
- Some automated tools exist
 - `up2date` for OS and kernel updates to red hat linux.
 - Debian `aptget`
 - Also `gnorpm` is helpful, but not great.
 - www.windowsupdate.microsoft.com (now has an automatic install)

Auditing - Unused services

- Disable any unused services and daemons.
- They may be exploited if they are vulnerable and you (the administrator) aren't aware they're running.
- The commands "ps" and "netstat" are useful for determining which processes are running, and what (if any) network ports processes are listening on.
- See Unix Review lecture notes or man pages for the use of ps and netstat.

Auditing - Unused services: “ps”

- Output of “ps -aux”

```
USER      PID  %CPU  %MEM    VSZ   RSS TTY      STAT   START       TIME COMMAND
root         1   2.2   0.8   1324   536 ?        S      21:20       0:05  init [5]
root         2   0.0   0.0     0     0 ?        SW     21:20       0:00  [kflushd]
root         3   0.0   0.0     0     0 ?        SW     21:20       0:00  [kupdate]
root         4   0.0   0.0     0     0 ?        SW     21:20       0:00  [kpiod]
root         5   0.0   0.0     0     0 ?        SW     21:20       0:00  [kswapd]
root         6   0.0   0.0     0     0 ?        SW<    21:20       0:00  [mdrecoveryd]
root       334   0.2   1.3   1660   844 ?        S      21:21       0:00  syslogd -m 0
root       344   0.0   1.3   1624   820 ?        S      21:21       0:00  klogd
rpc        359   0.0   0.9   1468   576 ?        S      21:21       0:00  portmap
root       375   0.0   0.0     0     0 ?        SW     21:21       0:00  [lockd]
root       376   0.0   0.0     0     0 ?        SW     21:21       0:00  [rpciod]
rpcuser    386   0.0   1.3   1572   824 ?        S      21:21       0:00  rpc.statd
root       401   0.0   0.8   1308   524 ?        S      21:21       0:00
    /usr/sbin/apmd -p
daemon    455   0.0   0.9   1356   576 ?        S      21:21       0:00  /usr/sbin/atd
```

Auditing - Unused services: open ports

- `netstat --inet --all` (Linux, show network ports)
- `netstat -f inet` (Solaris, Digital Unix)

Active Internet connections (servers and established)

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State
tcp	0	0	*:www	*:*	LISTEN
tcp	0	0	*:1024	*:*	LISTEN
tcp	0	0	*:sunrpc	*:*	LISTEN
udp	0	96	laptop.oit:1034	nic.umass.edu:domain	
udp	0	0	laptop.oit:1034	nic.umass.edu:domain	
udp	0	0	*:1027	*:*	
udp	0	0	*:986	*:*	
udp	0	0	*:1026	*:*	
udp	0	0	*:sunrpc	*:*	
udp	0	0	*:1025	*:*	
udp	0	0	*:sunrpc	*:*	
udp	0	0	*:1025	*:*	
udp	0	0	*:1024	*:*	

Auditing- Disabling inetd services

- Disable processes spawned by inetd by editing inetd.conf and “commenting out” the relevant line with a “#”.
- For example: removing the telnet inetd.conf entry:

```
#telnet stream tcp nowait root /usr/sbin/tcpd in.telnetd
```
- Then force the inetd daemon to re-read the configuration file:

```
kill -HUP `inetd pid`
```
- **Note: xinted process for disabling is slightly different**

Auditing- Disabling xinetd services

- Some modern linux distributions have moved toward xinetd in favor of inetd
- /etc/xinetd.conf defines some global defaults
- Services can be started up by individual files in /etc/xinetd.d (redhat)
- Somewhat akin to how startup files are maintained in SYSV style system startup
- Still need to force the daemon to re-read the configuration files:

```
/etc/init.d/xinetd restart
```

Auditing- Disabling xinetd services

```
# default: off
# description: ftp daemon
service ftp
{
    disable      = no
    flags        = REUSE
    socket_type= stream
    wait        = no
    user        = root
    server       = /usr/sbin/ftpd
    server_args  = -A -l
}
```

- Change 'disable=no to disable=yes and HUP xinetd to disable service

Auditing - Disabling stand-alone daemons

- Disable stand-alone daemons that are started at boot time by modifying the “rc” start-up script.
- For Example: **sendmail**
In `/etc/rc.d/rc2.d` the `sendmail` startup script is `S88sendmail`.
Move `S88sendmail` to `s88sendmail` and the “rc” process will ignore it. (The “rc” process is looking for all files starting with a capital letter)
- On Linux this can also be done with the `linuxconf` and `sysconfig` utilities.

Various Daemons

- **S05kudzu** detects and configures new and/or changed hardware on a system
- **S10network** Activates/Deactivates all network interfaces configured to start at boot time.
- **S11portmap** manages RPC connections, which are used by protocols such as NFS and NIS
- **S14nfslock** Network File System (NFS) functionality
- **S16apmd** battery/power management for laptops
- **S20random** used for random number generation
- **S25netfs** mounts NFS and Samba mount points
- **S30syslog** assists programs in logging events
- **S35identd** matches users to TCP connections
- **S40atd** like cron, but for users
- **S40crond** periodically runs programs
- **S45pcmcia** for supporting cards on laptops
- **S50inet** starts and runs inetd
- **S55xntpd** Runs the Networks Time Protocol for syncing clocks

Daemons

- There are plenty of others:
 - S57ypserv
 - S58yppasswdd
 - S59ypbind
 - S60lpd
 - S60nfs
 - S75keytable
 - S79autofs
 - S80sendmail
 - S85gpm
 - S85httpd
 - S89popauth
 - S90sshd
 - S90xfs
 - S91simap
 - S91spop3
 - S99local

Services that should be firewalled

- These services don't need internet connectivity at all.
 - Anything NFS related
 - RPC services (remote procedure call)
 - Printer lpd
- Probably don't need these:
 - Sendmail (plenty of holes historically)
 - Qmail
 - BIND (DNS)
 - Replace all of these with ssh (scp, and sftp)
 - r-services: rsh, rlogin, rcp
 - telnet
 - ftp

Auditing - User accounts and passwords

- Establish a clearly defined password policy
 - frequency of forced password changes
 - enforce mixture of alpha, numeric and other characters.
 - Educate users to not share or write down their passwords
- Disable unused accounts
- Enable shadowed passwords
- Frequently do the following:
 - Check that all accounts have a password
 - Check if any accounts other than root have **UID 0**
 - Run a password cracker to ensure secure passwords
- Don't run services that use clear text passwords: IMAP, POP, telnet. (connect thru ssh if you have to.)

Auditing- Unauthorized access

- If you need to run the “r-services” (you don’t), make sure it’s done as securely as possible.
- Disallow and periodically check for users’ **.rhosts** and **.shosts** files. These files list the hosts and users that are allowed to log into the user’s account via the r-services/ssh without providing a password.
- Disallow **/etc/hosts.equiv** and **/etc/ssh/shosts.equiv** which lists hosts considered to be “trusted” (i.e. users can connect via r-services/ssh from those hosts without supplying a password)

Auditing-Unauthorized access

- Don't allow root to log directly; force user with root password to log in then "su" to root.
- Or, only allow root to log in from the console.
(Assuming the console is in a secure location)
- On linux, [/etc/securetty](#) contains a list of terminals that root can login from.
- Why are you logging in as root at all? Don't run programs or surf the web as root.

Auditing- Other things to check

- Make sure root's \$PATH does NOT contain a "."
 - \$PATH is set to a list of directories where commands (binaries) can be found and is searched each time a command is executed.
`$PATH=/usr/bin:/usr/sbin:/usr/local/bin`
 - Attack: malicious user Mallory looks at roots .profile and realizes root has "." as part of the \$PATH. Mallory creates a shell script in her home directory called "ls" which copies /usr/bin/bash to a specified location and make it SUID root.
 - Mallory then tells the admin . that there's something wrong in her home directory, the admin goes to look in her home dir. and types "ls". If the "." is in the path before the directory that contains the real "ls", Mallory's "ls" will get executed, and Malloy will have access to an SUID root shell.

Auditing - Other things to check (cont'd)

- Periodically check the filesystem for files and directories with permissions of 777 or 666. (-rwxrwxrwx, and -rw-rw-rw-)

```
find / -perm 777 -print
```

```
find / -perm 666 -print
```

- Check the filesystem for SUID binaries. Determine if the SUID bits are necessary for normal system operation, and remove if not.

```
find / -perms -4000 -print (SUID)
```

```
find / -perms -2000 -print (SGID)
```

Local auditing Tools

- **Crack** - Password cracking program for UNIX
- **netstat** - Can be used to show “open” network ports.
- **lsof** - Used to show network ports associated with processes.
- **Tripwire** - Application to hash system files and periodically check to see if files have changed.
- **Bastille Linux** - Linux “hardening scripts”
- **Amap** - A port scanning tool which identifies applications and services even if they are not listening on the default port.

Other various suggestions

- Why stay connected to the Internet all the time?
 - If you aren't running a server, consider disconnecting network services at night, or while at work.
- Look at SELinux
 - Security enhanced (NSA)
 - strong, flexible mandatory access control architecture incorporated into the major subsystems of the kernel.
 - Type Enforcement®, Role-based Access Control, and Multi-level Security.

Remote Vulnerabilities and Exposures

- Over the course of the semester we will be studying several remote vulnerabilities.
- Some include:
 - Remote Buffer overflow
 - ARP spoofing
 - Session Hijacking
 - Unauthorized access to network services

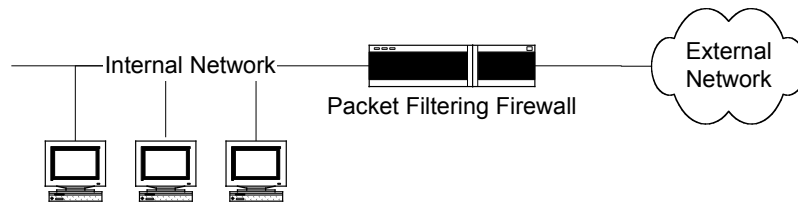
Remote Buffer Overflow - Countermeasures

- Disable the execution of code on the stack, via OS patch or kernel parameter
- Use safe programming practices.
- Apply vendor OS and application patches
 - Something you should do anyway for other types of exploits.

Unauthorized Access to Network services - Countermeasures

- IP based access control
 - host based firewalls
 - iptables
 - Snort, an IDS system www.snort.org
 - Zone alarm (windows) www.zonealarm.org
 - TCP Wrappers
 - <ftp://ftp.porcupine.org/unix/security>
 - Secure portmapper/rpcbind
 - <ftp://ftp.porcupine.org>
 - NFS shares (/etc/exports, /etc/dfs/dfstab)

Host-based firewalls



- An additional layer of defense to prevent unauthorized network access is a firewall, which operates on transport and network layers of the TCP/IP stack.
- Can be a separate box or software.
 - Beware of “NAT firewalls”, not quite the same.
 - “The Linksys EtherFast® Cable/DSL Router with 4-Port Switch ... Allowing up to 253 users, the built-in NAT technology acts as a firewall protecting your internal network.”
- Real firewalls determine the fate of a packet based upon any of the following criteria:
 - Transport protocol (TCP,UDP,ICMP)
 - Source and destination IP address
 - The source and destination port.

Host-based firewalls

- We have an entire class on firewalls
- Open Source Firewall/Packet Filter distributions:
 - iptables [ipchains, ipfwadm] (Linux)
 - IPFilter (Solaris, BSD, HP-UX, Irix)
 - ipfw (BSD)
- Commercial Firewall/Packet Filter distributions:
 - Raptor
 - Firewall-1
 - Cisco PIX

tcp_wrappers

- IP based access control to services spawned by `inetd` can be configured using the `tcp_wrappers` package developed by Weitse Venema.
- Stand alone daemons such as `ssh`, `sendmail` and `secure portmapper/rpcbind` can be compiled using `libwrap.a`, a shared library which provides `tcp_wrappers` access control.
- `tcp_wrappers` uses the configuration files `/etc/hosts.allow` and `/etc/hosts.deny` to determine whether or not a specific host can gain access to a service.

tcp_wrappers

- Example:

Check out the line for telnet in /etc/inetd.conf:

```
telnet stream tcp nowait root /usr/sbin/tcpd in.telnetd
```

"tcpd" is the wrapper program, and it calls `in.telnetd` (the telnet daemon) after it "authenticates" the remote host for a service.

- The best and most paranoid method of configuring is denying all services to all hosts;
- then give explicit permissions to those you want to be able to connect on an individual service basis.
- Why not disallow access to your home machine from all hosts but a specific machine at UMass?

tcp_wrappers

- In this example, I want to explicitly give telnet access to
 - 192.168.10.1 and 192.168.3.0\24
- `/etc/hosts.deny` would contain (`#` is a comment):

```
# service : host
# Deny all hosts and all services
ALL: ALL
```

`/etc/hosts.allow` would contain:

- ```
#service : host
Allow hosts
telnet : 192.168.10.1, 192.168.3.
```

# tcp\_wrappers

- tcp\_wrappers attempts to confirm hostname -> IP and IP->hostname mappings.
- Can be configured to drop IP source routed packets.
  - We'll talk in a later class about the exploits possible with source routing.
- Logs via the **syslog** utility.
  - A denied connection attempt gets logged as follows:

```
Mar 24 14:15:22 server ftpd[29291]: refused
connect from evilhost.crackerz.com
```

# Secure portmap/rpcbind

- `portmap` (BSD) and `rpcbind` (SysV) are used by remote procedure call (RPC) programs. They provide a mapping of program to RPC number.
- secure `portmap` and `rpcbind` provides `tcp_wrappers` support to control who can receive information from the `rpcbind/portmap` daemons.
- Although it's one level of defense, it doesn't prevent an attacker from figuring out RPC numbers and attacking RPC daemons directly.
- (these services should be firewalled!)

## Auditing your system from the outside

- NFS shares
- rpcinfo
- nmap
- Remote Auditing tools
- try exploits against specific services.

# Auditing - NFS

- **Network File Service** (NFS) allows a host to access a shared filesystem on another host as if it was a locally connected filesystem.
- NFS shares on the NFS server have an access control list to specify which hosts and users have access to a particular shared resource.
- Check that the NFS access lists are properly configured such that it's not sharing file systems to "the world"
- The **showmount** command can be used to view NFS exported filesystems.

# Aditing - nmap

- `nmap` is an extremely versatile port scanning tool which can be used to determine which services are running.
- It can also be used as a rudimentary packet filter or `tcp_wrappers` test.
- <http://www.insecure.org/nmap>
- Using the command “`nmap -sT laptop.oit.umass.edu`”

```
Starting nmap V. 2.53 by fyodor@insecure.org
Interesting ports on laptop.oit.umass.edu (128.119.xxx.xxx):
(The 1519 ports scanned but not shown below are in state: closed)
Port State Service
80/tcp open http
111/tcp open sunrpc
1024/tcp open kdm
6000/tcp open X11
```

# Auditing - Remote auditing tools.

- Several utilities are available to “attack” or gather information about services/daemons on a system.
  - SAINT - Based on SATAN utility
  - SARA - Also based on SATAN
  - Nessus - Excellent open source vulnerability scanner
    - <http://www.nessus.org>
- Commercial:
  - ISS scanner
  - Cybercop

# Backing up

- Once your system is installed, configured, and secured, **but not on the net** make a backup
- It's the only time you know you haven't been hacked yet.
- Attacks often have signatures:
  - Snort tries to stop attacks as they occur by filtering out specific packets. E.g., ones that contain "exec /bin/sh"
  - After an attack is complete, file time stamps, sizes, and contents will change.
- Checking your computer against what you installed is key;
- Restoring your computer to a previous state requires a backup.

# Tripwire

- Tripwire is a tool originally developed at Purdue as an academic project.
- The basic idea is keep a hash of each file on your system.
  - Anytime a single bit is altered in a file, the resulting hash will change.
  - Store the hashes of your files on readonly media (CD-ROM); compare results of the current system to the saved results: find files that have changed.
- Can be used in /etc, or even for your web pages.
- Source code available still, and there is a sourceforge project.