Practical (Linux) Security

SANOG IV Workshop 2004

July 2004

Presented by Hervey Allen
Network Startup Resource Center

What do we mean by “practical”?

Theoretical security concepts will be introduced followed by the some “real world/practical” examples.

How often do you hear, “Backup your system ,” but just how do you do this?

Anyone have some recommendations?

We'll revisit this later.

Core security concepts

Abstract, Theoretical, or the "end result":

- Maintaining confidentiality.
- Keeping our data safe from intruders.
- **Integrity:** protect from loss or change.
- **Authentication**

  Is this person who they claim to be?
  Is this person allowed access?
- **Availability**

  Are our systems up and running?

Maintaining confidentiality

A number of pieces are involved, including:

- Correct user and file permissions.
- Strong passwords.
- Trusting your users.
- Use of good cryptographic methods.

Keeping our data safe from intruders

You could argue this is the hardest thing to do.

- Keep people out who don't belong:
  - Trust your users.
  - Strong passwords.
  - Limit services you run.
  - Protect the services you do run.
- Encrypt data as needed.
- Backup data in case of intrusion or corruption.
- Don't forget about physical security.

Integrity

Protect your data against loss or change.

- Backup, backup, backup.
- Revision control.
- Intrusion detection systems (IDS).

Authentication

How do you ensure?:

- Someone accessing your system is who they claim to be?
  - Trusted users.
  - Strong passwords.
  - Public/Private keys.
- The person is allowed access?
  - Maintain accounts properly.
  - Correct user/group/file permissions.
  - Scan and watch for SUID and SGID.

Availability

Make sure your server and services are up and detect attacks like Denial of Service (DoS).

- Log what your services do and install log "watching" software.
- Setup notifications if there are problems.
- Scan for network attacks like spoofing (ARP), syn packet dumping, general packet source address spoofing, brute force attacks (dictionary password crack attempts).

More practical Linux/UNIX security

- Run only the services you plan on using.
- Use only the services that are necessary.
- Stay up-to-date and patch services as needed.
- Use secure passwords and force your users to use them.
- Consider if you need quotas.
- Restrict root access to services.
- Restrict access to services via tcpwrappers if appropriate.

practical unix terrorism graphic

More practial Linux/UNIX security cont.

- Restrict access to your box using IP firewall services (iptables/netfilter).
- Buffer overflow attacks. What to do.
- Log events and understand your logs.
- Install intrusion detection software.
- Back up your server's data!
- Think about physical security.
- Test your security model.
- Don't forget about your clients.

Some resources

As part of the books for this class you will be given:

Practical Unix & Internet Security, 3rd Edition

http://www.oreilly.com/catalog/puis3/index.html

There are many other useful publications at:

- http://www.oreilly.com/
- http://www.aw-bc.com/catalog/academic/discipline/0,,69948,00.html

Be sure to take advantage of this book to learn about the philosophy of security, particularly in the UNIX world.

Some more resources

We've setup a repository of security resources and references with some examples here:

- http://nsrc.org/security/

In addition, for more advanced topics see:

- *Linux Server Hacks* by Rob Flickenger from O'Reilly.
- *Network Server Hacks* by Andrew Lockhart from O'Reilly.

Security from the start

Make sure a server is secure *before* you connect it to your network.

Consider that you may experience some of these types of attacks:

Passive attacks:
- e.g. Packet sniffers, traffic analysis (*ngrep*, *dsniff*).

Active attacks:
- e.g. Connection hijacking, IP source spoofing, exploitation of weaknesses in IP stack or applications.

Denial of Service attacks: e.g. synflood.

"Man in the middle" attacks: Hijacking services.

Attacks against the network itself: e.g. smurf.

Understand what you are doing

A bad security solution is worse than no security at all. False sense of security.

Know what you are doing:

- Read all the documentation.
- Read sample configurations.
- Build test machines.
- Ask questions (don't be shy!).
- And, to repeat, join the announcements and/or security mailing list for your O/S and applications.

Step-by-step practical items

We'll go through each practical security item mentioned with some pointers and a possible hack or two.

Remember, we are discussing a server-based environment. For a desktop or laptop you would adjust accordingly.

Many of these items are similar or apply directly for FreeBSD, Solaris, and other Unices (UNIXes).

Run only the services you plan on using

Remember our “Intro to Linux” session?

- Use “`chkconfig –list`” to see what loads at system startup. Don't forget to check “xinetd”.
- Delete services you are not using.
- To see what is running use:

```
lsof -i
netstat -natup
ps -aux | more
```

Run only the services you plan on using cont.

Some services that often run that you might want to remove include:

- rpc, rpc.mountd, rpc.nfsd (*nfs support*)
- smbd and nmbd (*Samba Windows fileserver*)
- automount (*mount filesystems when accessed*)
- named (*DNS server*)
- inetd (*old-style tcpwrappers*)
- telnet rlogin, rexec, ftp (*not encrypted!*)
- finger, comsat, chargen, echo, identd (*id cmds*)

Fedora Core 2 ships with rpc, inetd and automount running.

Run only the services you need

What do we mean by this? Really this means, more or less, the following:

- Think through what it is you are providing to your clients.
- Is there some other way to provide a service that is more secure?
- Should you separate out services to more than one server?
  - NFS or Samba or shell access?

Consider if some services should run under the xinetd tcpwrapper

- Prior to xinetd Linux used inetd with the hosts.deny and hosts.allow files.
- /etc/rc.d/init.d/xinetd start (*to start*)
- /etc/xinetd.d/ (*configuration files*)
- Services are configured file-by-file under this directory.
- A service is turned off w/ “disable = yes”
- What does xinetd provide? ==>

What does xinetd provide?

- The xinetd daemon (service) listens for network packets for each service mentioned in the /etc/xinetd.d directory.
- Xinetd saves on memory and resources as a service is only started if a packet arrives for it, but it's better not to use xinetd for a loaded service like http.
- You can control how packets arrive or don't arrive on a service-by-service basis in a detailed manner using xinetd.

xinetd vs. iptables

- iptables permits full control over packets arriving for a service or server.
- iptables provides a more complete ruleset that you can apply to a service, including more fine-grained control over icmp and udp packets.
- Iptables is part of the kernel, thus it is more efficient.
- xinetd has (imho) an easier syntax to understand.

Some xinetd parameters

- **Disable:** determines if a services runs or not with xinetd.
- **Wait:** determines whether to run multiple instances of a service if there is more than one connection.
- **User:** under what user a service will run.
- **Instances:** maximum number of connections allowed per services. If “wait = nowait” then this defaults to no limit.
- **Server:** the name and location of the program to run fo rthe service being defined.
- **only_from:** Specify from where connections are accepted.
- **no_access:** Specify from where connections are not allowed.

Some more xinetd parameters

- **Interface:** you can specify on what network interface the service responds and listens.
- **cps:** if you have “cps = 10 30” this means accept up to 10 connections. If there are more, then shutdown the service for 30 seconds.
- Additional parameters include Id, Type, socket_type, Protocol, Group, server_args, log_type, log_on_success, Port.
- Xinetd allows for quite a bit of control at the application level for network services.
- The minimum parameters required in an xinetd configuration file to run a service using xinetd are “socket_type”, “user”, “server”, and “wait”.

Restrict access to your box using IP firewall services (iptables)

- Netfilter is the current firewall implementation in Linux.
- `iptables` is the command to manipulate the netfilter kernel component.
- `iptables` has three Default tables (input, ouput, forward) that act upon network packets (tcp, udp, icmp).
- `iptables` allows you to create custom tables and include flow-control statements in your filter rules.
- Fedora Core 2 ships with netfilter already configured in the kernel.

iptables cont.

- Fedora Core 2 has /etc/rc.d/init.d/iptables startup script and /etc/sysconfig/iptables ruleset predefined.
- You can NAT with iptables in a trivial (up to a very complex) manner.
  - `iptables -L`

Shows rules for all chains (tables).

  - `iptables -F`

Drop all existing filter rules.

  - `man iptables`

What you need to do to understand the example on the next page ==>

NAT with iptables

Here is an example set of commands to allow a private network to share an internet connection with one IP address (not always the best idea...):

- `echo "1" > /proc/sys/net/ipv4/ip_forward`
- `echo "1" > /proc/sys/net/ipv4/conf/all/rp_filter`
- `iptables -t nat -A POSTROUTING -o eth0 -j \ MASQUERADE`

rp_filter is a kernel rule that drops packets with source addresses that don't match corresponding routing table entries – i.e. that are IP spoofing.

"eth0" is the outside network interface we've used as an example for your gateway.

Stay up-to-date and patch services as needed

Be sure that you track all the services you are running.

- If you run Bind, Apache, and Exim then subscribe to the appropriate security mailing lists for each.
- Subscribe to generic security mailing lists that pertain to your OS or Linux version.
- Subscribe to general security lists.

Where to find some security mailing lists

General security mailing lists:

- BugTraq: http://www.securityfocus.com/
- CERT: http://www.cert.org/
- Rootshell: http://www.rootshell.com/

For Apache, Bind, Exim and SSH:

- http://www.apache.org/
- http://www.isc.org/ *(Bind)*
- http://www.exim.org/
- http://www.openssh.org/

For Fedora Core 2:

- http://fedora.redhat.com/

Patching your software

As needed download patches for the services you run. You should be notified of these via the mailing lists mentioned.

For your OS the vendor will often provide specific patches or update installers.

You can apply kernel patches, but carefully as your vendor has likely patched your kernel in a significant manner.

A sample software patch

You could consider patching your kernel to improve security in several areas using the grsecurity kernel patch (http://www.grsecurity.net/).

This patch includes (but is not limited to):

- Nonexecutable stacks.
- Filesystem security enhancements.
- Restrictions on access to /proc.
- Enhanced resource limits.

Applying the patch ==>

grsecurity kernel patch

First download the patch file that corresponds to your kernel version (`cat /proc/version`).

For a 2.6.n kernel with the patch (version 1.9.13) downloaded to /tmp you would do:

- `cd /usr/src/linux-2.6.n`
- `Patch -p1 < /tmp/grsecurity-2.0 \ -2.6.5.patch`

If there were problems you will see output on the screen.

To see the new available kernel features you would run "`make xconfig`" and click on the "Grsecurity" button.

Software patches conclusion

In general you can either apply a patch directly to installed software if:

- You have the original source for the software, and
- You plan on building the software from source.

Otherwise, the software vendor may supply an “updated” version of the software in installer (RPM) form.

Or, your vendor may supply updates as well*.

*Red Hat Server 9 had 500MB+ of updates by February, 2004.

Use secure passwords and force your users to use them

First, there are some issues to consider:

- "Bad" passwords can be guessed.
- If passwords become too complex, then users tend to write them down.
- Passwords sent unencrypted can be "sniffed" from the network and reused (consider the case of dsniff).

So, enforce strong passwords and don't run services that are unencrypted.

Services you should run cryptographically

- POP/IMAP with SSL only.
- Consider TLS-Enabled SMTP.
- Remove Telnet replace with SSH.
- Remove FTP replace with SCP or SFTP.
- Anonymous FTP is OK, but be careful if you allow user uploads.
- Require HTTPS (HTTP over SSL) for sensitive information.

What is a “good” password

- Combination of upper and lower-case letters, numbers and symbols.
  - Brute force attacker has to try many more combinations.
- Not in any dictionary, including hackers dictionaries.

Two samples of creating a good password:

$40&yc4f

"Money for nothing and your chicks for free"

wsR!vst?

"workshop students aRe not very sleepy today ?"

How to enforce good passwords

This is a bit harder, but new Linux distributions come with PAM (Pluggable Authentication Modules) preconfigured to use cracklib when a regular user sets their password.

See /etc/pam.d/system-auth in Fedora Core 2 for an example.

See /usr/share/doc/pam-0.77/txts/README.pam_cracklib for more information on Fedora Core 2.

Cracklib keeps a user from creating trivial passwords.

Cracklib

From "`locate cracklib`" on a Fedora Core 2 machine:

```
/usr/lib/cracklib_dict.hwm
/usr/lib/cracklib_dict.pwd
/usr/lib/cracklib_dict.pwi
/usr/share/doc/cracklib-2.7
/usr/share/doc/cracklib-2.7/MANIFEST
/usr/share/doc/cracklib-2.7/HISTORY
/usr/share/doc/cracklib-2.7/LICENCE
/usr/share/doc/cracklib-2.7/POSTER
/usr/share/doc/cracklib-2.7/README
/usr/share/doc/cracklib-2.7
/usr/share/doc/pam-0.77/txts/README.pam_cracklib
/lib/security/pam_cracklib.so
```

As you can see cracklib is installed, a cracklib dictionary, and the PAM cracklib shared library.

Cracklib password checks

Taken directly from the cracklib README file:

4) it's MIND-NUMBINGLY THOROUGH!

(is this beginning to read like a B-movie flyer, or what?)

CrackLib makes literally hundreds of tests to determine whether you've chosen a bad password.

It tries to generate words from your username and gecos entry to tries to match them against what you've chosen.

It checks for simplistic patterns.

It then tries to reverse-engineer your password into a dictionary word, and searches for it in your dictionary. *(> million entries!)*

- after all that, it's PROBABLY a safe(-ish) password. 8-)

Other password checkers

Some tools you could run against /etc/shadow after password generation for more thorough testing.

- John the Ripper: http://www.openwall.com/john/
- Crack: http://www.crypticide.org/users/alecm
- Slurpie: http://www.ussrback.com/docs/distributed/ (*URL may not work*)

You would create a cron entry to run a process against some/all user passwords once every certain period. "Cracked" passwords would generate an email warning to the user asking them to change their password or be disabled.

Consider if you need to use quotas

- Can you trust your users?
- What happens if /tmp or /home fills?
- Are these on separate disks or partitions?
- If not, you might want quotas.
- Quotas have been updated to version 2 in Fedora Core 2.

Practical quota tips ==>

Practical quota tips

If quotas are activated for groups:

- `edquota -p artc 'awk -F: '$3 >> 499 print $1' /etc/passwd'`

Useful to find and edit for all non-system (>500) users.

General Steps to Activate:

- /etc/fstab with quota/usrquota/grpquota entry for filesystem where you want quotas.
- On root of filesystem file aquota.user or aquota.group (ver 2) or quota.user/quota.group.
- Commands include quota, quotaon/quotaoff, quotastats, quotacheck.

Fedora Core 2 uses quota version 2.

Restrict root access to a minimal set of services

Check for files with setuid/setgid bits running as root. If you don't need these files, or users don't need to run them, then remove this bit (permission).

Consider running a service in a "sandboxed" environment using chroot.

Consider running a service under a different userid if possible.

Practical restriction tips ==>

Practical root restriction tips

To find all files with setuid or setgid bits set on a machine you can do:

- `find / -perm +6000 -type f -exec ls -ld {} \; > setuid.txt &`

You'll have a file listing all setuid/setgid files (but not scripts) on your machine. Fedora Core 2 had 53 files.

Use `chroot` to run services with their own root directory – i.e. in a "sandbox" or "jail".

You could consider running BIND in a chroot jail. We'll ask Joe what he thinks...

Bind in a chroot jail

From *Linux Server Hacks* by Rob Flickenger:

As root create a "named" user and group (if needed):

- `groupadd -g 25 named`
- `useradd -u 25 -g named -c "chroot BIND user" \ -d /var/named/jail -m named`

Now create the skeleton directory structure where named will run:

- `cd ~named`
- `mkdir -p var/{run,named}`
- `cp -Rav /var/named/data var/named/`

If you are acting as a slave for any zones then create a writeable directory for this:

- `mkdir var/named/slave`
- `chown named.named var/named/slave`

Bind in a chroot jail

Created dev/ and etc/ directories and copy critical system files:

- `mkdir {dev,etc}`
- `cp -av /dev/{null,random} dev/`
- `cp -av /etc/{localtime,named.conf,rndc.key} \ etc/`

Fix ownership and permission on these directories

- `chown root.root .`
- `chmod 0755 .`
- `chown named.named var/named/data/`
- `chmod 0700 var/named/data/`
- `chown named.named var/run/`

Update syslog if used for DNS logs:

- `syslogd -m 0 -a /var/named/jail/dev/log`

Bind in a chroot jail

Now see if named can run in this environment:

- `/usr/sbin/named -u named -t /var/named/jail \ -c /etc/named.conf`

If you have problems look in /var/log/syslog or /var/log/messages, and from /var/named/jail check your directories with:

- `ls -lR`

In the end BIND can still be compromised, but damage is contained (in theory) to the `/var/named/jail` directory structure.

Be careful about what you copy to a jail area as all items can be used in case of attack.

If you need to find what libraries a binary needs you can do "`ldd /dir/binary`".

How apache runs as user “apache”

Taken directly from */etc/httpd/conf/httpd.conf*:

```
# If you wish httpd to run as a different user or group, you
# must run httpd as root initially and it will switch.
#
# User/Group: The name (or #number) of the user/group to run
# httpd as. . On SCO (ODT 3) use "User nouser" and "Group
# nogroup". . On HPUX you may not be able to use shared memory
# as nobody, and the suggested workaround is to create a user
# www and use that user.
# NOTE that some kernels refuse to setgid(Group) or semctl
# (IPC_SET) when the value of (unsigned)Group is above 60000;
# don't use Group #-1 on these systems!
#

User apache
Group apache
```

Buffer overflow attacks

A Cracker pushes more data on to a services buffer than space provides. They can “break out” of the program space and execute arbitrary commands on your system with the privileges of the compromised service.

Many security patches deal with newly discovered buffer overflow holes.

Solutions to this include GCC compilers such as ProPolice from IBM and Stackguard from immunix.org.

Another solution is LibSafe from Avaya Labs.

LibSafe for buffer overflow attacks

LibSafe replaces several functions in the dynamic link loader under Linux. These include `gets()`, `strcpy()`, and `scanf()`.

Libsafe has three known failure points:

- Cannot accurately determine buffer size.
- Does not work on SUID binaries without static linking.
- Fails on programs compiled with the GCC compiler flag `-fomit-frame-pointer`.

You can find LibSafe at:

- http://www.research.avayalabs.com/project/libsafe/

We may install LibSafe in class in there is time.

Log events and understand your logs

This is time consuming – even with the many tools that are available.

You need to go through each service running and decide if you want to log events from this service. This has been partially done for you in /etc/syslog.conf under Fedora Core 2.

Ideally logs should be created or saved off your server. A cracker will alter your logs to cover their tracks.

Logging continued

You should consider setting up automated warning systems. Under Fedora Core 2 logwatch runs by default and sends a summary report via email to root once a day.

To configure what is logged read “man syslog.conf” for full details on how this file is formatted.

Consider using a central logging server. You can use /etc/syslog.conf to send events to another server via your network.

Logging continued

- Consider writing your logs to a CDR.
- Consider printing your logs.
- You can use `chattr` to keep people from removing your log files.
  - `chattr +a filename`
- `chattr` fails if the user has root access and realizes what you have done. Instead you can use the Linux capabilities mechanism and the lcap utility to remove the ability to remove the append-only attribute.

Logging continued

You can find lcap here:

- http://packetstormsecurity.org/linux/admin/lcap-0.0.3.tar.bz2

Download the package and as root do:

- `Tar xvfj lcap-0.0.3.tar.bz2 && \`
  `cd lcap-0.0.3 && make`
- `./lcap CAP_LINUX_IMMUTABLE`
- `./lcap CAP_SYS_RAWIO`

Add the two lcap commands to /etc/rc.local and reboot your system. You are done.

Yet more logging...

A few useful tools to monitor activity:

- **Swatch**: Simple WATCHer is available from http://swatch.sourceforge.net/ and will watch for "trigger" events in your logs and notify you immediately.
- **Process accounting**: turn this on under Linux to keep track of who has done what. Fedora Core 2 does not install this.
- **logwatch and logrotate**: read up on both of these "`man logwatch`" and "`man logrotate`" to understand how they work and what they do. Under Fedora Core 2 both are run automatically each by cron (see /etc/cron.daily).
- See http://nsrc.org/security/#logging for some more tools.

Networking monitoring/logging

A few useful network monitoring tools:

- **httptop**: can give you real time monitoring of your web traffic. Find this from http://examples.oreilly.com/.
- **Nagios**: monitors services running on hosts on your network as well as resources. Can monitor you of events via email, pager, etc. Find this at http://www.nagios.org/.
- **nmap**: network exploration tool and security scanner can identify machines and services on your network. Find this at http://www.insecure.org/nmap/.
- **ntop**: from http://www.ntop.org/ gives full featured protocol analysis of who's talking to whom on your network. Includes graphical reports and web interface.

Caveat: these tools can get you in trouble. Be sure you have permission to run them.

Install intrusion detection software

- Intrusion Detection System = IDS
- Network Intrusion Detection System = NIDS
- And, System Integrity Checking is a generic term for this.

An IDS monitors network traffic and warns if suspicious behavior is detected.

A System Integrity Checker looks for changes to files that are not expected and warns you of these.

Tripwire from http://www.tripwire.org is the best known of these. It monitors binary signature, size, expected change of size and more of files.

For a list of many tools see http://nsrc.org/#integrity

Snort intrusion detection system

Snort from http://www.snort.org/ is a very popular tool to detect unexpected network events using a known set of rules and patterns. This is a signature-based IDS.

Additional Snort add-ons include:

- **ACID**: Anaylsis Console for Intrusion Databases. Web front-end to IDS alert database(s). Good for large site. From http://acidlab.sourceforge.net/.
- **Sguil**: Snort GUID for Lamerz. Complex system to analyze possible IDS events with tools such as ethereal and TcpFlow as well as Snort. From http://sguil.sourceforge.net/.
- **Snort_inline**: from http://snort-inline.sf.net/. Detect intrusions and react to them.
- **SnortSam**: from http://www.snortsam.net/ to update firewalls on the fly to deal with attacks.

Back up your server's data!

Pretty hard to stress this more. If your security is compromised what will you do without a backup? A few basic items to consider are:

- What needs to be backed up.
- How often do you need to backup?
- Where will your backup media be in case of disaster (fire, flood, earthquake, theft)?
- What happens in case of total loss?
- What tools will you use? Tar, Arkeia, cpio?

Detailed considerations for backing up your server's data

- What do you want to backup?
- What do you need to backup?
- How often must you backup:?
  - User data
  - System configuration files
  - Operating system files
- What is the backup rotation? Daily, weekly, monthly, semi-annually, yearly?
- What type of backup media are you going to use?
- Will you use the same media and software for each piece of your backup process?
- Where will you backup your data?
- Where will you keep copies of your backups?
- Have you tested your backups? I.E. have you tried a restore?
- What will you do if you lose your server? Do you have a place to restore your data in this case?

Tools to use for backups

- **Arkeia**: commercial product:
  - http://www.arkeia.com/
  - http://nsrc/security/#backups
- **dd**: convert and copy a file.
  - `man dd`
  - `dd if=/dev/hda of=/dev/fd0/bootsector.bin bs=512 count=1`

Backs up a boot sector to a floppy.

  - `dd if=/dev/fd0/bootsector.bin of=/dev/hda bs=512 count=1`

Recovers from floppy to hda. Be *very* careful doing this!

Tools to use for backups cont.

- **cpio**: copy files to and from archives:
  - cpitool: http://www.nickb.org/utils/
  - `man cpio`
- **dump**: ext2/ext3 filesystem backup.
  - `man dump`
- **rsync:** remote copy.
  - `man rsync`.
- **tar**: read
  - `man tar` (scary!)

A few practical backup tricks

You can use ssh and tar together to quickly backup parts of your server. For instance, to backup all /home directories to another server as a single image:

- `root@machine1# tar xzvf - /home/ | \`
  `ssh machine2 "cat > machine1-homes.tgz"`

Or, you can use rsync over ssh if you wish to keep directories synchronized between two locations:

- `rsync -ave ssh remote:/home/docs .`

rsync with ssh and ssh keys

Later today we'll discuss ssh and the use of ssh keys to connect to a remote machine without passwords and use encryption.

Image if in /etc/cron.daily/sync-web you did the following:

- `Rsync -ae ssh /var/www/html/ \`
  `backup.machine:/var/www/html/`

This recursively copies your root web documents to a backup machine using rsync via ssh.

If you use the “--delete” option in rsync, then files removed on your local machine would be removed on the remote machine as well when you run this.

Think about physical security

All the security in the world does nothing against a disgruntled employee, server sitting out in the open, people who copy keys, and so on.

Backups: where do you physically keep your them? Who has access to them. Are they separate from your server?

Logs: are they on a separate and physically secure log server? Printed to a separate printer?

Bootloader password and encrypted files: what happens if someone walks off with your machine?! Or, how about just the hard drive(s)?

Physical access = total access

Test your security model

Once you have in place what you believe to be a secure server try connecting to it from an external machine. Verify that your security model works as expected. Try circumventing your own rules.

Run a security scanner against your server (your network as well?). A nice tool to run against your server is Nessus. You can find this product here:

http://www.nessus.org/

We'll take a look at this...

Don't forget about your clients

Make sure that your users must connect to your servers in such ways as to help ensure the integrity of their data and their user accounts.

Insist on software clients that use encryption like SSH vs. Telnet, SCP/SFTP vs. FTP, POP/IMAP over SSL.

Human clients running their OS'es... Dealing with Windows security issues such as viruses, Windows Updates, worms, spyware, etc...

Virus scanning software to defang email on your server?

Scripts as well – can rename files like .exe, .pif, .com, .scr, .vbs, .bat to fn.ft.*txt*.

Social issues. Security is inconvenient. For instance, Windows *still* does not ship with SSH – ouch!

We'll take a look at “securing” a Windows 2000 box now...

Remember these resources

CERT (Coordinated Emergency Response Team)

- http://www.cert.org/ and http://www.us-cert.gov/cas/index.html

Nice List of Security Resources for Linux/UNIX

- http://www.yolinux.com/TUTORIALS/LinuxSecurityTools.html

nmap: Network exploration tool and security scanner

- http://www.insecure.org/nmap/

O'Reilly Books

- http://www.oreilly.com/

SANS Computer Security and Mailing Lists

- http://www.sans.org/ and http://www.sans.org/newsletters/risk/

Security Documents from nsrc.org

- http://nsrc.org/security/ and http://nsrc.org/freebsd-tips.html

And, don't forget your own local help at http://www.sanog.org/!

A list of hacks

A few hacks to consider from *Network Server Hacks* by Andrew Lockhart and O'Reilly books:

- Secure Mount Points (noexec, nosuid, nodev, ro)
- Scan for SUID and SGID Programs (see presentation)
- Check for Listening Services (lsof, netstat)
- Scan Your Network for Vulnerabilities (nmap, nessus)
- Secure BIND (use chroot jail)
- Secure MySQL (use chroot jail)
- Monitor Your Logs Automatically (swatch, logwatch)
- Detect Intrusions with Snort
- Scan for Root Kits (chkrootkit)
- Find the Owner of a Network (whois – What about dig... ;-)).

A list of hacks cont.

A few hacks to consider from *Linux Server Hacks* by Rob Flickenger and O'Reilly books:

- Immutable Files in ext2/ext3 (*lsattr/chattr*)
- Cleaning up after Ex-Users (forwards, cron jobs, html, ppp...)
- Tracking Changes with rcs2log
- Backup Up with tar over ssh (tar and pipe "|" facility)
- Using rsync over ssh (use -e to for alt. shell, i.e. ssh)
- Creating a Firewalls from the Command Line of any Server
- What's Holding That Port Open? (lsof, procps)
- Checking On Open Files and Sockets with lsof (lsof /mnt)
- Using ssh-Agent Effectively (respawn shell with ssh-agent)
- Using proftpd with a MySQL Authentication Source (mod_sql)

Summary

- Be sure you understand what you are doing.
- Start with some useful books, such as:
  - *Practical Unix & Internet Security, 3rd Edition* http://www.oreilly.com/catalog/puis3/index.html
  - *Linux Server Hacks*, Rob Flickenger, O'Reilly books.
  - *Network Security Hacks*, Andrew Lockhart, O'Reilly books.
- And, as a reminder, here are the practical security items we started with:
  - Run only the services you plan on using.
  - Use only the services that are necessary.
  - Stay up-to-date and patch services as needed.
  - Use secure passwords and force your users to use them.

Summary cont.

The practical security items we started with cont:

- Consider if you need to use quotas.
- Restrict root access to a services.
- Restrict access to services via tcpwrappers if appropriate.
- Restrict access to your box using IP firewall services (iptables/netfilter).
- Buffer overflow attacks. What to do.
- Log events and understand your logs.
- Install intrusion detection software.
- Back up your server's data!
- Think about physical security.
- Test your security model.
- Don't forget about your clients.

Summary cont.

We have not discussed Windows vs. Linux/UNIX security in this presentation.

In general – With Linux/UNIX you can *see* what is happening or what is broken. Under Windows you may not have access to that which you are trying to secure or fix.

For fun/pain see:

- http://darkwing.uoregon.edu/~hervey/secure-w2k.html
- http://nsrc.org/isp.html#server

Conclusion

More security means less convenience, but a security breach can be the least convenient moment of all.

There is always a tradeoff between how much security you put in place and what services you are providing.

Your users may grumble, but they'll really grumble if their data is compromised – Remind them of this :-)