Global Information Assurance Certification Paper

Build Securely a Shadow Sensor Step-by-Step Powered by Slackware Linux

By Guy Bruneau, GCIA

This configuration process is used to deploy Shadow sensors powered by Slackware Linux operating system. This setup was developed for sensors using IDE or SCSI drives and can also be used with laptops (PCMCIA cards).

The full installation using this setup is ~80 MB in size and provides no services except for Shadow and Secure Shell for remote management.

This process doesn't address how to setup a Shadow Analysis station. More information available at: http://www.nswc.navy.mil/ISSEC/CID/

This setup can be used as well to deploy a Snort IDS sensor by replacing the Shadow package with a pre-built Snort package. However, this setup won't be discussed in this paper. A pre-built package maybe available at a later date.

The Shadow ISO image powered by the Slackware Linux OS can be downloaded at: ftp://ftp.whitehats.ca/pub/ids/shadow-slack/shadow.iso

The MD5 signature for the ISO image is available at: ftp://ftp.whitehats.ca/pub/ids/shadow-slack/shadow.md5

I invite you to read the release notes on the CD, which contains additional information not contained in this document.

Important: Before you start, make sure you are disconnected from the network until the sensor has been securely configured.

Partitioning the drive:

- Boot on the system using the Slackware CD-ROM.

Special note: If the computer doesn't support bootable CD's, you can create two boot diskettes. To build the boot diskettes, do the following:

- With a Windows workstation, go to the MS-DOS prompt and select the CD-ROM drive
- cd to bootdsks.144 directory and type rawrite and follow the instrustions
- Select either scsi.s (SCSI drive) or bare.i (IDE drive)
- cd to rootdsks directory and type rawrite and follow the instructions
- Select color.gz
- Boot the system with the boot diskette followed by the root diskette when asked

To partition the drive, login as root and run ***cfdisk /dev/hda*** (IDE drive). If this isn't a new drive, delete the old partitions before starting.

- hda1: / = 500 MB (Select new, select primary, size is 500, beginning, bootable)
- hda2: SWAP = 128 MB (Select Pri/Log Free Space, new, primary, size is 128, beginning)
- Change hda2 to swap by selecting *type 82*
- hda3 (Select Pri/Log Free Space, new, primary, remainder of disk)
- Select Write to save the new settings to disk
- Select *Quit* to exit

Now that you have partitioned the drive, and saved your setting, you are ready to setup the Operating System.

- Run setup
- Select addswap
- Continue with installation: yes
- Select Linux installation partition
    - /dev/hda1 (format, 4096)
    - /dev/hda3 (format, 4096)
    - Select mount point for /dev/hda3: /LOG
    - Select add none and continue with setup
- Select continue to go to the SOURCE section
- Select 1 to install from a Slackware CD-ROM
- Installation type is slakware

- Install Slackware 7.1 with the following trees (use spacebar to delete packages):

A, AP, D, N

- Select install everything (full)
- Make a boot disk for recovery (LILO)
- After the boot disk, choose continue with the configuration
- Skip modem configuration
- Install LILO and select expert
    - Select Begin, at the blank prompt press enter, select standard, install to MBR and none
    - Add Linux and choose the root partition (i.e. /dev/hda1)
    - Use *Linux* as a partition name
    - Install LILO
- Configure the network with your settings
- Probe the network card
- Setup the root password
- Exit setup
- Reboot (*init 6* at the prompt)
- Manually eject the CD-ROM

- Log back in the sensor as root
- Delete residual mail *rm /var/spool/mail/root*
- In order for the ssh daemon to start on the sensor, you must execute the shell script located in the /root directory called ssh_key_generator.
- To start the daemon, type /usr/sbin/sshd. Next time the sensor reboots it will start automatically through the rc scripts.
- Run **pkgtool** and **remove** the SCSI or IDE packages that are not needed (if using IDE drive - scsi and scsimods, or if using SCSI drive - ide and modules, and if not using a laptop – pcmcia)

Additional security against non-required accounts

- Lock the following accounts as follows:
  - passwd -l *ftp* (repeat for the other 4 accounts)
  - lp, - news, - uucp and - games

Configure SSH TCP Wrappers in the following way:

- vi /etc/hosts.allow
- Add in the TCP Wrappers file which host(s) are allowed to connect to the sensor
  - i.e: ALL: 192.168.3. \
    192.168.2.6 \
    .site.ca
- The /etc/hosts.deny has been configured to deny ALL (ALL: ALL) by default

Configure pre-installed firewall (rc.firewall)

Note: You need a firewall (ipchains) to allow the sensor to be as invisible as possible. An example is available at Annex B. You can create your own at: http://www.linux-firewall-tools.com/linux/firewall/index.html

O- Configured the firewall located in */etc/rc.d* directory or create your own
O- *chmod 755 /etc/rc.d/rc.firewall* to enable the firewall
O- To start the firewall now do: */etc/rc.d/rc.firewall* at the prompt

How to manually mount a CD-ROM or diskette

To manually mount the CD-ROM do:

```
mount /dev/hdc /cdrom -t iso9660    (mount the cdrom)
umount /cdrom                       (un-mount the cdrom)
```

The cdrom maybe hdb, hdc or hdd depending where it has been installed in the computer. To find out which device is the CD-ROM, do *dmesg |more*

To manually mount the floppy do:

```
mkdir /fd0                  (create floppy directory)
mount /dev/fd0 /fd0 -t vfat (mount the floppy)
umount /fd0                 (un-mount the floppy)
```

Operating System Patches

The Slackware web site should be monitored for any new patches that should be applied on the selected packages at Annex A. The site is http://www.slackware.com

The security lists are available at:

http://www.slackware.com/lists/archive/list.php?l=slackware-security&y=2000
http://www.slackware.com/lists/archive/list.php?l=slackware-security&y=2001

Patches and upgrades will be posted on this site as well. To install the patch updates as follow:

```
telinit 1
upgradepkg <patch>.tgz
telinit 3
```

Setting up the NICs if undetected during the installation

Note: A list of the NIC kernel modules is in the /etc/rc.d/rc.modules file. I recommend using an Intel EtherExpress Pro/100 PCI card for the Shadow packet collection. If the NIC card detection setup fail, the card can be manually added to this script.

Setting up NIC modules (example)

vi /etc/rc.d/rc.netdevice

```
# RealTek 8129/8139 to communicate with the Management station (eth0)

/sbin/modprobe rtl8139

# Intel EtherExpress Pro/100 PCI support used to collect packets (eth1):

/sbin/modprobe eepro100
```

Open Secure Shell (openssh) is part of the installation on this CD

Note: Secure Shell is normally used to transfer data between the monitoring station and the sensor. The instructions on how to setup an analysis station are available at the NSWC site. The site is at: : http://www.nswc.navy.mil/ISSEC/

Configure Shadow in the following way (Pre-configured with this installation):

- cd /usr/local/logger/sensor and make the following changes:

    - vi gmt.ph and verify the following settings:

    * Decide whether you want to use local time or GMT (default GMT)
    * $LOGPROG = "/usr/sbin/tcpdump" (or its exact location)
    * $PROGPAR = "-i eth0" (or eth1 if using the second card as the traffic collector)
    * $GZIPPROG = "/bin/gzip" (or its exact location)
    * $LOGDIR = "/LOG/RAW/gmt" (if not using default, change it here)

The message of the day changed to reflect more proactive security (Pre-configured with this installation):

- vi /etc/motd

```
**************************************************************
This is a controlled access system.
This station is monitored at all times.
Only authorized users may connect
**************************************************************
```

- cp /etc/motd /etc/issue

Update rc.local to start local applications:

O- vi /etc/rc.d/rc.local and add the following services

```
# Starting eth1. Make sure the next two entries match the card performing traffic
# collection (eth0 or eth1) if you are using the 2 cards concept. If only one card is
# used, don't enter the next two lines
echo "Assign an IP to eth1 now..."
/sbin/ifconfig eth1 0.0.0.0 promisc
echo "Starting shadow sensor..."
/usr/local/logger/sensor/start_logger.pl gmt
echo "Starting firewall..."
/etc/rc.d/rc.firewall
```

Update cronjob to cut hourly Shadow files, update the time against a timeserver (pre-configured on Shadow CD):

crontab –e

Sync with a time server on a daily basis

17 23 * * * /usr/sbin/ntpdate time-a.nist.gov > /dev/null 2>1&
18 23 * * * /sbin/hwclock –systohc

Cut a new Shadow file on a hourly basis

0 * * * * /usr/local/logger/sensor/sensor_driver.pl gmt > /dev/null 2>1&

Note: The following has already been done on the Shadow ISO CD. It has been added to show the security configuration for this sensor.

Secure inetd.conf file in /etc directory:

O- vi /etc/inetd.conf

Note: Comment out all the applications in the file to turn off all unnecessary services.

Secure the following applications in each files in the /etc/rc.d directory by commenting them out. Use vi to edit each files. See annex C for an example of the scripts:

rc.S
- O- issue
- O- motd
- O- pcmcia (if not using a laptop)

rc.M
- O- rc.font
- O- rc.ibcs2
- O- rc.httpd
- O- rc.samba
- O- rc.gpm

rc.inet2
- O- IN-SERV "lpd"
- O- LPSPOOL="/var/spool/lpd"
- O- IPV4_FORWARD
- O- rpc.portmap
- O- All NFS services
- O- rpc.mountd
- O- rpc.nfsd

Update cronjob to start Shadow, update time, clean up dead links and cut new logs each hour:

O- crontab -e

Sync with a time server on a daily basis

17 23 * * * /usr/sbin/ntpdate time-a.nist.gov
18 23 * * * /sbin/hwclock --systohc

Cut a new Shadow log on a hourly basis
0 * * * * /usr/local/logger/sensor/sensor_driver.pl gmt > /dev/null 2>1&

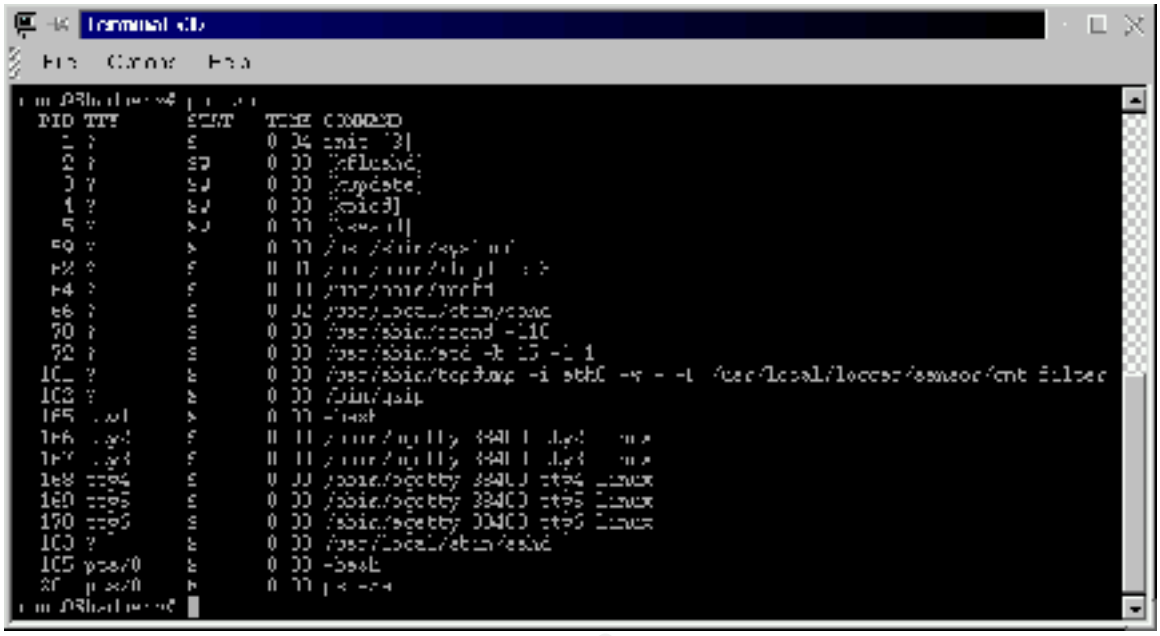
O- Reboot the sensor. It is now ready for traffic collection.

Note: During the first reboot, you will notice some errors in directory /LOG/RAW/gmt with files sniff and sensor date. This is normal as those files do not exist yet and the sensor is creating them.

O- Log in as root
O- Run *ps -xa* and verify the services running. (See picture 1)
O- Run *netstat -at* and verify the active connections (See picture 2). ssh should be the only service listening for remote login.
O- Check Annex D for a NMAP port reconnaissance probe confirming ssh is the only available service.
O- cd /LOG/RAW/gmt and verify the sensor is collecting. Do an *ls -l* and look for the hourly file that looks like this: tcp.20010312.gz with 0 bytes.
O- The sensor is ready to be connected to the network.

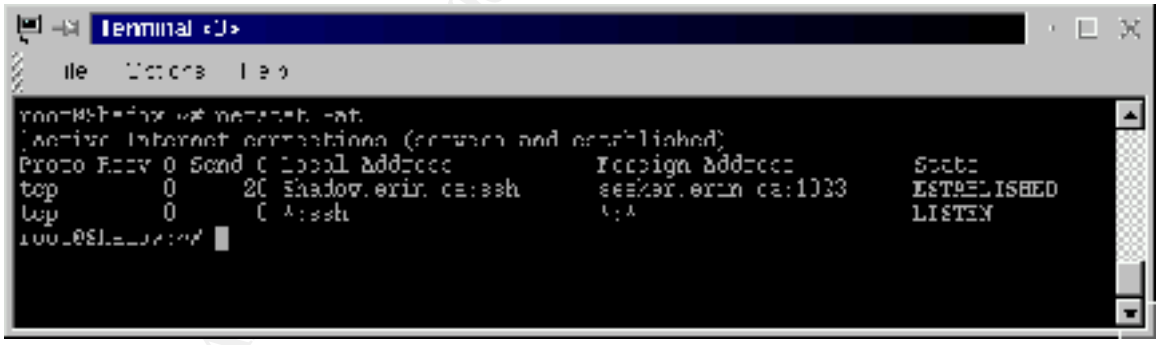
Shadow Sensor Active Services

Picture 1



Shadow Sensor Active Connections

Picture 2



Annex A:

List of package to install on the sensor

A: Slackware basic installation

Note: Select IDS or SCSI kernel depending of the hardware you are installing on.

ide: Linux IDE, and CD-ROM kernel support
scsi: Linux kernel, with SCSI support (**Only if using SCSI**)
aaa_base: Basic Linux filesystem package.
bash: GNU bash-2.04
bin: Binaries that go in /bin and /usr/bin.
cxxlibs: C++ shared libraries
devs: Device files
e2fsprog: e2fsprogs-1.18
elflibs: Assorted ELF shared libraries
elvis: Elvis is a text editor
etc: System config files and utilities
fileutls: These are the GNU file management utilities.
find: GNU findutils
fsmods: Filesystem modules for Linux 2.2.16
glibcso: glibc-2.1.3 runtime support
grep: GNU grep
gzip: GNU zip compression utilities.
hdsetup: The Slackware setup/package maintenance system
isapnp: isa pnp tools
ldso: The dynamic linker/loader
less: A text pager utility
lilo: LILO
man: man pages
modules: Linux kernel modules
modutils: module utilities
pciutils: Linux PCI utilities
pcmcia: PCMCIA card service support (**Only if using a laptop**)
procps: procps-2.0.6, psmisc-18, procinfo-17
scsimods: Linux SCSI, RAID, and CD-ROM kernel modules (**Only if using SCSI**)
sh_utils: GNU sh-utils
shadow: Shadow password suite
sysklogd: Sysklogd
sysvinit: sysvinit
tar: GNU tar
txtutils: GNU textutils
util: A huge collection of essential utilities
zoneinfo: Time zone database

AP: Additional application

groff: GNU groff 1.16 document formatting system.
manpages: Man-pages 1.24
vim: Version 5.6 of Vim: Vi IMproved

D: Development support

perl: perl-5.6.0

N: Networking

logger: A pre-built package for the Shadow software by Guy Bruneau (Shadow.iso only)
netmods: Network support modules for linux-2.2.16.
openssh: Openssh from www.openssh.org
security: A pre-built security script by Guy Bruneau (Shadow.iso only)
tcpdump: TCPdump version 3.4
tcpip1: TCP/IP networking programs and support files.
tcpip2: Extra TCP/IP programs.
xntp: Network Time Protocol

Annex B:

IPChains firewall

```
#!/bin/sh

# Script generated Sun Mar 11 15:09:42 2001


# ------------------------------------------------------------------------
# Copyright (C) 1997, 1998, 1999, 2000  Robert L. Ziegler
#
#  Permission to use, copy, modify, and distribute this software and its
#  documentation for educational, research, private and non-profit purposes,
#  without fee, and without a written agreement is hereby granted.
#  This software is provided as an example and basis for individual firewall
#  development.  This software is provided without warranty.
#
#  Any material furnished by Robert L. Ziegler is furnished on an
#  "as is" basis.  He makes no warranties of any kind, either expressed
#  or implied as to any matter including, but not limited to, warranty
#  of fitness for a particular purpose, exclusivity or results obtained
#  from use of the material.
# ------------------------------------------------------------------------

#  /etc/rc.d/rc.firewall
#  Invoked from /etc/rc.d/rc.local.

echo "Starting firewalling... "

# ------------------------------------------------------------------------
#  Some definitions for easy maintenance.
#  EDIT THESE TO SUIT YOUR SYSTEM AND ISP.

EXTERNAL_INTERFACE="eth0"             # Internet connected interface
LOOPBACK_INTERFACE="lo"               # or your local naming convention

IPADDR="192.168.30.20"                # your IP address

ANYWHERE="any/0"                      # match any IP address

NAMESERVER_1="192.168.30.1"                 # everyone must have at least one


LOOPBACK="127.0.0.0/8"                # reserved loopback address range
CLASS_A="10.0.0.0/8"                  # class A private networks
CLASS_B="172.16.0.0/12"               # class B private networks
CLASS_C="192.168.0.0/16"              # class C private networks
CLASS_D_MULTICAST="224.0.0.0/4"       # class D multicast addresses
CLASS_E_RESERVED_NET="240.0.0.0/5"    # class E reserved addresses
BROADCAST_SRC="0.0.0.0"               # broadcast source address
BROADCAST_DEST="255.255.255.255"      # broadcast destination address
PRIVPORTS="0:1023"                    # well known, privileged port range
UNPRIVPORTS="1024:65535"              # unprivileged port range
```

```
# ------------------------------------------------------------------------

NFS_PORT="2049"                    # (TCP/UDP) NFS
SOCKS_PORT="1080"                  # (TCP) Socks
OPENWINDOWS_PORT="2000"            # (TCP) openwindows

# X Windows port allocation begins at 6000 and increments to 6063
# for each additional server running.
XWINDOW_PORTS="6000:6063"          # (TCP) X windows

# The SSH client starts at 1023 and works down to 513 for each
# additional simultaneous connection originating from a privileged port.
# Clients can optionally be configured to use only unprivileged ports.
SSH_LOCAL_PORTS="1022:65535"       # port range for local clients
SSH_REMOTE_PORTS="513:65535"       # port range for remote clients

# traceroute usually uses -S 32769:65535 -D 33434:33523
TRACEROUTE_SRC_PORTS="32769:65535"
TRACEROUTE_DEST_PORTS="33434:33523"

# ------------------------------------------------------------------------
# Default policy is DENY
# Explicitly accept desired INCOMING & OUTGOING connections

   # Remove all existing rules belonging to this filter
   ipchains -F

   # Set the default policy of the filter to deny.
   ipchains -P input  DENY
   ipchains -P output REJECT
   ipchains -P forward DENY

# ------------------------------------------------------------------------

   # Enable TCP SYN Cookie Protection
   echo 1 > /proc/sys/net/ipv4/tcp_syncookies

   # Enable always defragging Protection
   echo 1 > /proc/sys/net/ipv4/ip_always_defrag

   # Enable broadcast echo  Protection
   echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts

   # Enable bad error message  Protection
   echo 1 > /proc/sys/net/ipv4/icmp_ignore_bogus_error_responses

   # Enable IP spoofing protection
   # turn on Source Address Verification
   for f in /proc/sys/net/ipv4/conf/*/rp_filter; do
      echo 1 > $f
   done

   # Disable ICMP Redirect Acceptance
   for f in /proc/sys/net/ipv4/conf/*/accept_redirects; do
      echo 0 > $f
```

```
   done

   for f in /proc/sys/net/ipv4/conf/*/send_redirects; do
      echo 0 > $f
   done

   # Disable Source Routed Packets
   for f in /proc/sys/net/ipv4/conf/*/accept_source_route; do
      echo 0 > $f
   done

   # Log Spoofed Packets, Source Routed Packets, Redirect Packets
   for f in /proc/sys/net/ipv4/conf/*/log_martians; do
      echo 1 > $f
   done


# ------------------------------------------------------------------------
# LOOPBACK

   # Unlimited traffic on the loopback interface.

   ipchains -A input  -i $LOOPBACK_INTERFACE  -j ACCEPT
   ipchains -A output -i $LOOPBACK_INTERFACE  -j ACCEPT

# ------------------------------------------------------------------------
# Network Ghouls

   # Deny access to jerks
   # --------------------
   # /etc/rc.d/rc.firewall.blocked contains a list of
   # ipchains -A input -i $EXTERNAL_INTERFACE -s address -j DENY
   # rules to block from any access.

   # Refuse any connection from problem sites
   if [ -f /etc/rc.d/rc.firewall.blocked ]; then
      . /etc/rc.d/rc.firewall.blocked
   fi

# ------------------------------------------------------------------------
# SPOOFING & BAD ADDRESSES
# Refuse spoofed packets.
# Ignore blatantly illegal source addresses.
# Protect yourself from sending to bad addresses.

   # Refuse incoming packets pretending to be from the external address.
   ipchains -A input   -s $IPADDR -j DENY -l

   # Refuse incoming packets claiming to be from a Class A, B or C private network
   ipchains -A input   -s $CLASS_A -j DENY
   ipchains -A input   -s $CLASS_B -j DENY
   ipchains -A input   -s $CLASS_C -j  ACCEPT

   # Refuse broadcast address SOURCE packets
   ipchains -A input   -s $BROADCAST_DEST -j DENY -l
   ipchains -A input   -d $BROADCAST_SRC -j DENY -l
```

```
   # Refuse Class D multicast addresses
   # Multicast is illegal as a source address.
   # Multicast uses UDP.
   ipchains -A input   -s $CLASS_D_MULTICAST -j DENY

   # Refuse Class E reserved IP  addresses
   ipchains -A input   -s $CLASS_E_RESERVED_NET -j DENY -l

   # Refuse special addresses defined as reserved by the IANA.
   # Note:  The remaining reserved addresses are not included.
   # Filtering them causes problems as reserved blocks are
   # being allocated more often now.

   # Note:  this list includes the loopback, multicast, & reserved addresses.

   # 0.*.*.*          - Can't be blocked for DHCP users.
   # 127.*.*.*        - LoopBack
   # 169.254.*.*      - Link Local Networks
   # 192.0.2.*        - TEST-NET
   # 224-255.*.*.*    - Classes D & E, plus unallocated.

   ipchains -A input   -s 0.0.0.0/8 -j DENY -l
   ipchains -A input   -s 127.0.0.0/8 -j DENY -l
   ipchains -A input   -s 169.254.0.0/16 -j DENY -l
   ipchains -A input   -s 192.0.2.0/24 -j DENY -l
   ipchains -A input   -s 224.0.0.0/3 -j DENY -l

# ------------------------------------------------------------------------
# NOTE:
#     The symbolic names used in /etc/services for the port numbers vary by
#     supplier.  Using them is less error prone and more meaningful, though.

# ------------------------------------------------------------------------
# TCP UNPRIVILEGED PORTS
# Avoid ports subject to protocol & system administration problems.

   # NFS: establishing a TCP connection
   ipchains -A input  -i $EXTERNAL_INTERFACE -p tcp -y \
         --destination-port $NFS_PORT -j DENY -l
   ipchains -A output -i $EXTERNAL_INTERFACE -p tcp -y \
         --destination-port $NFS_PORT -j REJECT

   # openwindows: establishing a connection
   ipchains -A input  -i $EXTERNAL_INTERFACE -p tcp -y \
         --destination-port $OPENWINDOWS_PORT -j DENY -l
   ipchains -A output -i $EXTERNAL_INTERFACE -p tcp -y \
         --destination-port $OPENWINDOWS_PORT -j REJECT


   # Xwindows: establishing a connection
   ipchains -A input  -i $EXTERNAL_INTERFACE -p tcp -y \
         --destination-port $XWINDOW_PORTS -j DENY -l
   ipchains -A output -i $EXTERNAL_INTERFACE -p tcp -y \
         --destination-port $XWINDOW_PORTS -j REJECT
```

```
   # SOCKS: establishing a connection
   ipchains -A input  -i $EXTERNAL_INTERFACE -p tcp -y \
         --destination-port $SOCKS_PORT -j DENY -l
   ipchains -A output -i $EXTERNAL_INTERFACE -p tcp -y \
         --destination-port $SOCKS_PORT -j REJECT

# ------------------------------------------------------------------------
# UDP UNPRIVILEGED PORTS
# Avoid ports subject to protocol & system administration problems.

   ipchains -A input  -i $EXTERNAL_INTERFACE -p udp  \
         --destination-port $NFS_PORT -j DENY -l

   # UDP INCOMING TRACEROUTE
   # traceroute usually uses -S 32769:65535 -D 33434:33523

   ipchains -A input  -i $EXTERNAL_INTERFACE -p udp  \
         --source-port $TRACEROUTE_SRC_PORTS \
         --destination-port $TRACEROUTE_DEST_PORTS -j DENY -l


   # DNS server (53)
   # ---------------

   # DNS: full server
   # ----------------

   # server/client to server query or response

   ipchains -A input  -i $EXTERNAL_INTERFACE -p udp  \
         --source-port $UNPRIVPORTS \
         -d $IPADDR 53 -j ACCEPT

   ipchains -A output -i $EXTERNAL_INTERFACE -p udp  \
         -s $IPADDR 53 \
         --destination-port $UNPRIVPORTS -j ACCEPT

   ipchains -A output -i $EXTERNAL_INTERFACE -p udp  \
         -s $IPADDR 53 \
         --destination-port 53 -j ACCEPT

   ipchains -A input  -i $EXTERNAL_INTERFACE -p udp  \
         --source-port 53 \
         -d $IPADDR 53 -j ACCEPT


   # DNS client (53)
   # ---------------
   ipchains -A output -i $EXTERNAL_INTERFACE -p udp  \
         -s $IPADDR $UNPRIVPORTS \
         -d $NAMESERVER_1 53 -j ACCEPT

   ipchains -A input  -i $EXTERNAL_INTERFACE -p udp  \
         -s $NAMESERVER_1 53 \
         -d $IPADDR $UNPRIVPORTS -j ACCEPT
```

```
   ipchains -A output -i $EXTERNAL_INTERFACE -p tcp  \
         -s $IPADDR $UNPRIVPORTS \
         -d $NAMESERVER_1 53 -j ACCEPT

   ipchains -A input  -i $EXTERNAL_INTERFACE -p tcp ! -y \
         -s $NAMESERVER_1 53 \
         -d $IPADDR $UNPRIVPORTS -j ACCEPT

   # ------------------------------------------------------------------

   # SSH server (22)
   # ---------------
   ipchains -A input  -i $EXTERNAL_INTERFACE -p tcp  \
         --source-port $SSH_REMOTE_PORTS \
         -d $IPADDR 22 -j ACCEPT

   ipchains -A output -i $EXTERNAL_INTERFACE -p tcp ! -y \
         -s $IPADDR 22 \
         --destination-port $SSH_REMOTE_PORTS -j ACCEPT


   # SSH client (22)
   # ---------------
   ipchains -A output -i $EXTERNAL_INTERFACE -p tcp  \
         -s $IPADDR $SSH_LOCAL_PORTS \
         --destination-port 22 -j ACCEPT

   ipchains -A input  -i $EXTERNAL_INTERFACE -p tcp ! -y \
         --source-port 22 \
         -d $IPADDR $SSH_LOCAL_PORTS -j ACCEPT

# ------------------------------------------------------------------
# UDP accept only on selected ports
# ---------------------------------

   # ------------------------------------------------------------------

   # NTP TIME clients (123)
   # ----------------------
   ipchains -A output -i $EXTERNAL_INTERFACE -p udp  \
         -s $IPADDR 123 \
         -d 129.6.15.28 123 -j ACCEPT

   ipchains -A input  -i $EXTERNAL_INTERFACE -p udp  \
         -s 129.6.15.28 123 \
         -d $IPADDR 123 -j ACCEPT

   # ------------------------------------------------------------------

   # OUTGOING TRACEROUTE
   # -------------------
   ipchains -A output -i $EXTERNAL_INTERFACE -p udp  \
         -s $IPADDR $TRACEROUTE_SRC_PORTS \
         --destination-port $TRACEROUTE_DEST_PORTS -j ACCEPT -l
```

```
# ------------------------------------------------------------------------
# ICMP

   #    To prevent denial of service attacks based on ICMP bombs, filter
   #    incoming Redirect (5) and outgoing Destination Unreachable (3).
   #    Note, however, disabling Destination Unreachable (3) is not
   #    advisable, as it is used to negotiate packet fragment size.

   # For bi-directional ping.
   #     Message Types:  Echo_Reply (0),  Echo_Request (8)
   #     To prevent attacks, limit the src addresses to your ISP range.
   #
   # For outgoing traceroute.
   #     Message Types:  INCOMING Dest_Unreachable (3), Time_Exceeded (11)
   #     default UDP base: 33434 to base+nhops-1
   #
   # For incoming traceroute.
   #     Message Types:  OUTGOING Dest_Unreachable (3), Time_Exceeded (11)
   #     To block this, deny OUTGOING 3 and 11

   #  0: echo-reply (pong)
   #  3: destination-unreachable, port-unreachable, fragmentation-needed, etc.
   #  4: source-quench
   #  5: redirect
   #  8: echo-request (ping)
   # 11: time-exceeded
   # 12: parameter-problem

   ipchains -A input  -i $EXTERNAL_INTERFACE -p icmp  \
          --icmp-type echo-reply \
          -d $IPADDR -j ACCEPT

   ipchains -A input  -i $EXTERNAL_INTERFACE -p icmp  \
          --icmp-type destination-unreachable \
          -d $IPADDR -j ACCEPT

   ipchains -A input  -i $EXTERNAL_INTERFACE -p icmp  \
          --icmp-type source-quench \
          -d $IPADDR -j ACCEPT

   ipchains -A input  -i $EXTERNAL_INTERFACE -p icmp  \
          --icmp-type time-exceeded \
          -d $IPADDR -j ACCEPT

   ipchains -A input  -i $EXTERNAL_INTERFACE -p icmp  \
          --icmp-type parameter-problem \
          -d $IPADDR -j ACCEPT

   ipchains -A input  -i $EXTERNAL_INTERFACE -p icmp  \
          -s 192.168.30.1 echo-request \
          -d $IPADDR -j ACCEPT

   ipchains -A output -i $EXTERNAL_INTERFACE -p icmp  \
          -s $IPADDR echo-reply \
          -d 192.168.30.1 -j ACCEPT
```

```
   ipchains -A output -i $EXTERNAL_INTERFACE -p icmp \
         -s $IPADDR fragmentation-needed -j ACCEPT

   ipchains -A output -i $EXTERNAL_INTERFACE -p icmp \
         -s $IPADDR source-quench -j ACCEPT

   ipchains -A output -i $EXTERNAL_INTERFACE -p icmp \
         -s $IPADDR echo-request -j ACCEPT

   ipchains -A output -i $EXTERNAL_INTERFACE -p icmp \
         -s $IPADDR parameter-problem -j ACCEPT

# ------------------------------------------------------------------------
# Enable logging for selected denied packets

   ipchains -A input  -i $EXTERNAL_INTERFACE -p tcp  -j DENY -l

   ipchains -A input  -i $EXTERNAL_INTERFACE -p udp  \
         --destination-port $PRIVPORTS -j DENY -l

   ipchains -A input  -i $EXTERNAL_INTERFACE -p udp  \
         --destination-port $UNPRIVPORTS -j DENY -l


   ipchains -A input  -i $EXTERNAL_INTERFACE -p icmp  \
         --icmp-type 5 -j DENY -l
   ipchains -A input  -i $EXTERNAL_INTERFACE -p icmp  \
         --icmp-type 13:255 -j DENY -l

   ipchains -A output -i $EXTERNAL_INTERFACE  -j REJECT -l

# ------------------------------------------------------------------------

echo "done"

exit 0
```

Annex C:

RC configuration files

rc.S - System initialization script

```
#!/bin/sh
#
# /etc/rc.d/rc.S:  System initialization script.
#
# Mostly written by:  Patrick J. Volkerding, <volkerdi@ftp.cdrom.com>
#

PATH=/sbin:/usr/sbin:/bin:/usr/bin

# enable swapping
/sbin/swapon -a

# Start update.
/sbin/update &

# Automatic module loading.  To load and unload kernel modules
# automatically as needed, uncomment the lines below to run kerneld.
# In some cases, you'll need to create aliases to load the correct
# module. For more information, see the docs in /usr/doc/modules.
# NOTE: This is commented out by default, since running kerneld has
# caused some experimental kernels to hang during boot.
#if [ -x /sbin/kerneld ]; then
#  /sbin/kerneld
#fi

# Test to see if the root partition is read-only, like it ought to be.
READWRITE=no
if echo -n >> "Testing filesystem status"; then
 rm -f "Testing filesystem status"
 READWRITE=yes
fi

# Check the integrity of all filesystems
if [ ! $READWRITE = yes ]; then
 /sbin/fsck -A -a
 # If there was a failure, drop into single-user mode.
 if [ $? -gt 1 ] ; then
  echo
  echo
  echo "***************************************"
  echo "fsck returned error code - REBOOT NOW!"
  echo "***************************************"
  echo
  echo
  /bin/login
 fi
 # Remount the root filesystem in read-write mode
```

```
 echo "Remounting root device with read-write enabled."
 /sbin/mount -w -v -n -o remount /
 if [ $? -gt 0 ] ; then
  echo
  echo "Attempt to remount root device as read-write failed!  This is going to"
  echo "cause serious problems...  "
  echo
  echo "If you're using the UMSDOS filesystem, you **MUST** mount the root partition"
  echo "read-write!  You can make sure the root filesystem is getting mounted "
  echo "read-write with the 'rw' flag to Loadlin:"
  echo
  echo "loadlin vmlinuz root=/dev/hda1 rw   (replace /dev/hda1 with your root device)"
  echo
  echo "Normal bootdisks can be made to mount a system read-write with the rdev command:"
  echo
  echo "rdev -R /dev/fd0 0"
  echo
  echo "You can also get into your system by using a bootkernel disk with a command"
  echo "like this on the LILO prompt line:  (change the root partition name as needed)"
  echo
  echo "LILO: mount root=/dev/hda1 rw"
  echo
  echo "Please press ENTER to continue, then reboot and use one of the above methods to"
  echo -n "get into your machine and start looking for the problem. "
  read junk;
 fi
else
 echo "Testing filesystem status: read-write filesystem"
 if cat /etc/fstab | grep ' / ' | grep umsdos 1> /dev/null 2> /dev/null ; then
  ROOTTYPE="umsdos"
 fi
 if [ ! "$ROOTTYPE" = "umsdos" ]; then # no warn for UMSDOS
  cat << EOF

*** ERROR: Root partition has already been mounted read-write. Cannot check!

For filesystem checking to work properly, your system must initially mount
the root partition as read only. Please modify your kernel with 'rdev' so that
it does this. If you're booting with LILO, add a line:

   read-only

to the Linux section in your /etc/lilo.conf and type 'lilo' to reinstall it.

If you boot from a kernel on a floppy disk, put it in the drive and type:
  rdev -R /dev/fd0 1

If you boot from a bootkernel disk, or with Loadlin, you can add the 'ro' flag.

This will fix the problem *AND* eliminate this annoying message. :^)

EOF
  echo -n "Press ENTER to continue. "
  read junk;
 fi
fi
```

```
# remove /etc/mtab* so that mount will create it with a root entry
/bin/rm -f /etc/mtab* /etc/nologin /etc/shutdownpid

# mount file systems in fstab (and create an entry for /)
# but not NFS because TCP/IP is not yet configured
/sbin/mount -a -v -t nonfs

# Clean up temporary files on the /var volume:
/bin/rm -f /var/run/utmp /var/run/*.pid

# Looks like we have to create this.
cat /dev/null > /var/run/utmp

# Configure the system clock.
# This can be changed if your system keeps GMT.
if [ -x /sbin/clock ]; then
  /sbin/clock -s
fi

if [ "$ROOTTYPE" = "umsdos" ]; then # we need to update any files added in DOS:
 echo "Synchronizing UMSDOS directory structure:"
 echo "  umssync -r99 -v- /"
 umssync -r99 -v- /
fi

# Setup the /etc/issue and /etc/motd to reflect the current kernel level:
# THESE WIPE ANY CHANGES YOU MAKE TO /ETC/ISSUE AND /ETC/MOTD WITH EACH
# BOOT. COMMENT THEM OUT IF YOU WANT TO MAKE CUSTOM VERSIONS.
#echo > /etc/issue
#echo Welcome to Linux `/bin/uname -a | /bin/cut -d\  -f3`. >> /etc/issue
#echo >> /etc/issue
#echo "`/bin/uname -a | /bin/cut -d\  -f1,3`." > /etc/motd

# Configure ISA Plug-and-Play devices:
if [ -r /etc/isapnp.conf ]; then
  if [ -x /sbin/isapnp ]; then
    /sbin/isapnp /etc/isapnp.conf
  fi
fi

# This loads any kernel modules that are needed.  These might be required to
# use your CD-ROM drive, bus mouse, ethernet card, or other optional hardware.
if [ -x /etc/rc.d/rc.modules ]; then
  . /etc/rc.d/rc.modules
fi

# Initialize PCMCIA devices:
#
# NOTE: This had been closer to the top of this script so that PCMCIA devices
# could be fsck'ed along with the other drives.  This had some unfortunate
# side effects, however, since root isn't yet read-write, and /var might not
# even be mounted the .pid files can't be correctly written in /var/run and
# the pcmcia system can't be correctly shut down.  If you want some PCMCIA
# partition to be mounted at boot (or when the card is inserted) then add
# the appropriate lines to /etc/pcmcia/scsi.opts.
```

```
#
#if [ -x /etc/rc.d/rc.pcmcia ] ; then
#  . /etc/rc.d/rc.pcmcia start
#fi

# Run serial port setup script:
# (CAREFUL! This can make some systems hang if the rc.serial script isn't
# set up correctly. If this happens, you may have to edit the file from a
# boot disk)
#
# . /etc/rc.d/rc.serial
```

rc.M - Multi-user run levels

```
#!/bin/sh
#
# rc.M          This file is executed by init(8) when the system is being
#               initialized for one of the "multi user" run levels (i.e.
#               levels 1 through 6).  It usually does mounting of file
#               systems et al.
#
# Version:      @(#)/etc/rc.d/rc.M            2.02      02/26/93
#
# Author:       Fred N. van Kempen, <waltje@uwalt.nl.mugnet.org>
#               Heavily modified by Patrick Volkerding <volkerdi@ftp.cdrom.com>
#

# Tell the viewers what's going to happen...
echo "Going multiuser..."

# Screen blanks after 15 minutes idle time.
/bin/setterm -blank 15

# Look for a CD-ROM in a CD-ROM drive, and if one is found,
# mount it under /cdrom.  This must happen before any of the
# binaries on the CD are needed.
#
# If you don't have a CD-ROM and want to disable this, set the
# /etc/rc.d/rc.cdrom permissions to non-executable: chmod 644 /etc/rc.d/rc.cdrom
#
if [ -x /etc/rc.d/rc.cdrom ]; then
  . /etc/rc.d/rc.cdrom
fi

# If there's no /etc/HOSTNAME, fall back on this default:
if [ ! -r /etc/HOSTNAME ]; then
 echo "darkstar.example.net" > /etc/HOSTNAME
fi

# Set the hostname.  This might not work correctly if TCP/IP is not
# compiled in the kernel.
/bin/hostname `cat /etc/HOSTNAME | cut -f1 -d .`
```

```
# Initialize the NET subsystem.
if [ -x /etc/rc.d/rc.inet1 ]; then
  . /etc/rc.d/rc.inet1
  . /etc/rc.d/rc.inet2
else
  if [ -x /usr/sbin/syslogd ]; then
    /usr/sbin/syslogd
    sleep 1 # Prevents a race condition with SMP kernels
    /usr/sbin/klogd
  fi
#  if [ -x /usr/sbin/lpd ]; then
#    /usr/sbin/lpd
#  fi
fi

# Start netatalk. (a file/print server for Macs using Appletalk)
#if [ -x /etc/rc.d/rc.atalk ]; then
#  /etc/rc.d/rc.atalk
#fi

# Start crond (Dillon's crond):
# If you want cron to actually log activity to /var/adm/cron, then change
# -l10 to -l8 to increase the logging level.
/usr/sbin/crond -l10 >>/var/adm/cron 2>&1

# Remove stale locks and junk files (must be done after mount -a!)
/bin/rm -f /var/spool/locks/* /var/lock/* /var/spool/uucp/LCK..* /tmp/.X*lock /tmp/core /core 1> /dev/null
2> /dev/null

# Remove stale hunt sockets so the game can start.
if [ -r /tmp/hunt -o -r /tmp/hunt.stats ]; then
  echo "Removing your stale hunt sockets from /tmp..."
  /bin/rm -f /tmp/hunt*
fi

# Ensure basic filesystem permissions sanity.
chmod 755 /
chmod 1777 /tmp /var/tmp

# Update all the shared library links automatically
/sbin/ldconfig

# Start the sendmail daemon:
#if [ -x /usr/sbin/sendmail ]; then
#  echo "Starting sendmail daemon (/usr/sbin/sendmail -bd -q15m)..."
#  /usr/sbin/sendmail -bd -q15m
#fi

# Start the APM daemon if APM is enabled in the kernel:
if [ -x /usr/sbin/apmd ]; then
  if cat /proc/apm 1> /dev/null 2> /dev/null ; then
    echo "Starting APM daemon..."
    /usr/sbin/apmd
  fi
fi
```

```
# Load a custom screen font if the user has an rc.font script.
#if [ -x /etc/rc.d/rc.font ]; then
#  . /etc/rc.d/rc.font
#fi

# iBCS Emulation for Linux
# The Intel Binary Compatibility Specification, or iBCS, specifies the
# interfaces between application programs and the surrounding operating
# system environment for i386 based systems. There are however several
# flavours of iBCS in use - SVR4, SVR3 plus several vendor specific
# extensions to SVR3 which are slightly different and incompatible. The
# iBCS emulator for Linux supports all flavours known so far.
#if [ -x /etc/rc.d/rc.ibcs2 ]; then
#  . /etc/rc.d/rc.ibcs2
#fi

# Start Web server:
#if [ -x /etc/rc.d/rc.httpd ]; then
#  . /etc/rc.d/rc.httpd
#fi

# Start Samba (a file/print server for Win95/NT machines):
#if [ -x /etc/rc.d/rc.samba ]; then
#  . /etc/rc.d/rc.samba
#fi

# Load a custom keymap if the user has an rc.keymap script.
if [ -x /etc/rc.d/rc.keymap ]; then
  . /etc/rc.d/rc.keymap
fi

# Start the local setup procedure.
if [ -x /etc/rc.d/rc.local ]; then
  . /etc/rc.d/rc.local
fi

# All done.
```

rc.inet2 - Start INET system

```
#!/bin/sh
#
# rc.inet2      This shell script boots up the entire INET system.
#               Note, that when this script is used to also fire
#               up any important remote NFS disks (like the /usr
#               distribution), care must be taken to actually
#               have all the needed binaries online _now_ ...
#
# Author:       Fred N. van Kempen, <waltje@uwalt.nl.mugnet.org>
# Modified for Slackware by Patrick Volkerding <volkerdi@slackware.com>
#

# Some constants:
```

```
NET="/usr/sbin"
#IN_SERV="lpd"
#LPSPOOL="/var/spool/lpd"

# If we see IPv4 packet forwarding support in the kernel, we will turn it on.
# This was the default for 2.0.x kernels, but with recent kernels it must be
# activated through a file in /proc.  IPv4 packet forwarding support is
# required if you plan to use your Linux machine as a router or firewall.
# If you don't want your Linux machine to forward packets, change the 1 below
# to a 0.
#IPV4_FORWARD=1
#if [ -f /proc/sys/net/ipv4/ip_forward ]; then
#  if [ "$IPV4_FORWARD" = "1" ]; then
#   echo "Activating IPv4 packet forwarding..."
#   echo 1 > /proc/sys/net/ipv4/ip_forward
#   else
#     echo "Disabling IPv4 packet forwarding..."
#     echo 0 > /proc/sys/net/ipv4/ip_forward
#  fi
#fi

# When using IPv4 packet forwarding, you will also get the rp_filter, which
# automatically rejects incoming packets if the routing table entry for their
# source address doesn't match the network interface they're arriving on.  This
# has security advantages because it prevents the so-called IP spoofing,
# however it can pose problems if you use asymmetric routing (packets from you
# to a host take a different path than packets from that host to you) or if
# you operate a non-routing host which has several IP addresses on different
# interfaces.  To turn rp_filter off, uncomment the lines below:
# if [ -r /proc/sys/net/ipv4/conf/all/rp_filter ]; then
#   echo "Disabling rp_filter..."
#   echo 0 > /proc/sys/net/ipv4/conf/all/rp_filter
# fi

# Start the SUN RPC Portmapper:
#if [ -f /sbin/rpc.portmap ]; then
#  echo "Starting /sbin/rpc.portmap..."
#  /sbin/rpc.portmap
#fi

# At this point, we are ready to talk to The World...

# Mount NFS filesystems:
#echo "Mounting remote file systems..."
#/sbin/mount -a -t nfs         # This may be our /usr runtime!!!
# Show the mounted volumes:
#/sbin/mount -v -t nfs

# Begin a list of started daemons:
echo -n "Starting daemons: "

# Start the SYSLOGD/KLOGD daemons:
if [ -f ${NET}/syslogd ]; then
  echo -n " syslogd"
  ${NET}/syslogd
  sleep 1 # prevent syslogd/klogd race condition on SMP kernels
```

```
  echo -n " klogd"
  ${NET}/klogd
fi

# Start the INET SuperServer:
if [ -f ${NET}/inetd ]; then
  echo -n " inetd"
  ${NET}/inetd
else
  echo
  echo "WARNING:  ${NET}/inetd not found."
  echo -n "Continuing daemon loading: "
fi

# Look for sshd in the two most common locations (compiled with --prefix=/usr
# or with --prefix=/usr/local) and if we find it, start it up
if [ -x /usr/local/sbin/sshd ]; then
  echo -n " sshd"
  /usr/local/sbin/sshd
elif [ -x /usr/sbin/sshd ]; then
  echo -n " sshd"
  /usr/sbin/sshd
fi

# # Start the NAMED/BIND name server:
# if [ -f ${NET}/named ]; then
#   echo -n " named"
#   ${NET}/named -u daemon -g daemon
# fi

# # Start the ROUTEd server:
# if [ -f ${NET}/routed ]; then
#   echo -n " routed"
#   ${NET}/routed -g -s
# fi

# # Start the RWHO server:
# if [ -f ${NET}/rwhod ]; then
#   echo -n " rwhod"
#   ${NET}/rwhod -t -s
# fi

# Start the various INET servers:
for server in ${IN_SERV} ; do
  if [ -f ${NET}/${server} ]; then
    echo -n " ${server}"
    ${NET}/${server}
  fi
done

# # Setting up NIS:
# # (NOTE: For detailed information about setting up NIS, see the documentation
# # in /usr/doc/yp-tools, /usr/doc/ypbind, and /usr/doc/ypserv)
# #
# # First, we must set the NIS domainname.  NOTE: this is not
# # necessarily the same as your DNS domainname, set in
```

```
# # /etc/resolv.conf!  The NIS domainname is the name of a domain
# # served by your NIS server.
#
# if [ -r /etc/defaultdomain ]; then
#   nisdomainname `cat /etc/defaultdomain`
# fi
#
# # Then, we start up ypbind.  It will use broadcast to find a server.
#
# if [ -d /var/yp ] ; then
#   echo -n " ypbind"
#   ${NET}/ypbind
# fi
#
# # If you are the NIS master server for the NIS domain, then
# # you must run rpc.yppasswdd, which is the RPC server that
# # lets users change their passwords.
#
# if [ -x ${NET}/rpc.yppasswdd ]; then
#   echo -n " yppasswdd"
#   ${NET}/rpc.yppasswdd
# fi

# # Start the various SUN RPC servers:
#if [ -f /sbin/rpc.portmap ]; then
#   Start the NFS server daemons.
#  if [ -f ${NET}/rpc.mountd ]; then
  #  echo -n " mountd"
   # ${NET}/rpc.mountd
#  fi
#  if [ -f ${NET}/rpc.nfsd ]; then
#    echo -n " nfsd"
#    ${NET}/rpc.nfsd
#  fi
#  # Fire up the PC-NFS daemon(s):
#  if [ -f ${NET}/rpc.pcnfsd ]; then
#    echo -n " pcnfsd"
#    ${NET}/rpc.pcnfsd ${LPSPOOL}
#  fi
#  if [ -f ${NET}/rpc.bwnfsd ]; then
#    echo -n " bwnfsd"
#    ${NET}/rpc.bwnfsd ${LPSPOOL}
#  fi
#fi # Done starting various SUN RPC servers.

# The 'echo' below will put a carriage return at the end
# of the list of started servers.
echo

# Done!
```

Annex D:

Sensor audit with NMAP Port Scanner

Starting nmap V. 2.53 by fyodor@insecure.org (www.insecure.org/nmap/)

Interesting ports on shadow.erin.ca (192.168.30.20):
(The 1522 ports scanned but not shown below are in state: filtered)

Port	State	Service
22/tcp	open	ssh

TCP Sequence Prediction: Class=random positive increments
Difficulty=3271560 (Good luck!)
Remote operating system guess: Linux 2.1.122 - 2.2.14

Nmap run completed -- 1 IP address (1 host up) scanned in 1055 seconds

Note: The ssh service is of limited access since libwrap (TCP Wrapper equivalent) is enable and configured to only allow access to the monitoring station.

Slackware Linux security files

/etc/inetd.conf	Daemon configuration file
/etc/rc.d/rc.S	Start up script for single user mode
/etc/rc.d/rc.M	Start up script for Multi user mode
/etc/issue	Change to reflect something other than Linux version
/etc/motd	Change Message of the Day
/etc/rc.d/rc.firewall	Setup firewall according to example in this book.
/var/adm/messages	General log file
/var/adm/syslog	Syslog file
/usr/local/logger	Shadow directory

References

Securing Linux step-by-step, Version 1.0
SANS Institute

Linux Firewall
by Robert L. Ziegler, New Riders 2000

Intrusion Detection: Shadow Style step-by-step guide, Version 1.2.2
SANS Institute 1998

Installing Shadow
http://www.nswc.navy.mil/ISSEC/CID/part3.html

Shadow step-by-step Intrusion Detection using TCPdump
http://www.nswc.navy.mil/ISSEC/CID/shadow.ppt

SANS UNIX Practicum 6.5
By Lee Brotzman, SANS Institute