Getting Started

If you are running the X window as root

Create a normal user

```
# adduser
```

OR

```
# pw user add username -m
# passwd username
# chmod 700 /home/username
```

Logout and re login as a normal user

User Administration

What is your user id ?

```
% id
```

What is your home directory and default shell

```
% echo $HOME $SHELL
% cat /etc/passwd | grep ^username
```
(the two last fields are your home directory and default shell)

Which group do you belong to ?

(as normal user)
```
% id
```

Changing the GECOS

```
% chpass
```

Disabling a User

```
# pw user add testusername
# passwd testusername
```

Login as testusername

```
# pw lock testusername
```

Login as testusername once again

Removing User

```
# rmuser username
```

Who else are logged in ?

```
% w
```

What are the times you logged in/out ?

```
% last username
```

What are the times all the system users logged in/out ?

```
% last
```

Passwords

```
% cat /etc/password
```

You should see a "*" instead of the md5 encrypted password in the password field like in the example below

```
vicky:*:1000:100::/home/vicky:/bin/csh
```

```
% ls -l /etc/master.passwd
```

You actual crypt password is stored in /etc/master.passwd

```
# cat /etc/master.passwd
```

you should be able to see the crypt password here

su access

```
% su -
su: Sorry
```

Now from root console do

```
# pw user mod username -G wheel
```

Now open a new virtual terminal using normal account
Verify you belong to wheel group

```
% id
% su -
```

You should now be able to su to root

SUDO

visudo to edit the sudo files

If you are comfortable with vi EDITOR use

```
# export EDITOR=/bin/ed
# visudo
```

```
Cmnd_Alias      WHOAMI = /usr/bin/whoami
vicky   locahost = WHOAMI
```

Now as normal user execute

```
% /usr/bin/whoami
```

Now as a normal user execute

```
% sudo /usr/bin/whoami
```

What difference did you find ?

File System

Create a new file

```
% touch new.txt
```

Check it's permission

```
% ls -l new.txt
```

What permission do it have ?

Give read and write permission to everybody
`chmod o+rw new.txt`

Find SUID and SGID files

```
# find / -type f -perm +4000
# find / -type f -perm +2000
```

File Flags

Immutable Flag

Create a file and check it's default flags

```
%touch testfile
%ls -ol
```

Assign User immutable flag to the file

```
%chflags uchg testfile
%ls -ol
% mv testfile testfile.moved
```

Remove User immutable flag from the file

```
% chflags nouchg testfile
% ls -ol
% mv testfile testfile.moved
```

Append Only Flag

Create a file and check it's default flags

```
%touch testfile
%ls -ol
```

Assign Append only flag

```
%chflags uappnd testfile
%ls -ol
```

Add some data to the file

```
%echo "test" >> testfile
Try to truncate the file
%cat /dev/null > testfile
```

Check the contents of the file again

```
%cat testfile
test
```

Remove the Append only flag

```
%chflags nouappnd testfile
```

Truncate the file now

```
%cat /dev/null > testfile
%cat testfile
```

Mount Options

Check the currently mounted file systems

```
% df
```

Check the mount options

```
% mount
```

Mount Options

If you have separate /var,/tmp or /home partition remount it with security enhanced options

```
# mount -u -o remount,noexec,nosuid,nodev /var

Update your /etc/fstab accordingly
/dev/adXsXX /var  noexec,nosuid,nodev     /var
```

Inetd Ratelimiting

 # vi /etc/inetd.conf

Change the lines as follows:

```
#echo   stream  tcp     nowait  root    internal
echo   stream  tcp     nowait/10/1     root    internal
```

Start inetd daemon from command line if not already running

```
# ps ax | grep inetd
# inetd
```

Open a new terminal window

```
% telnet 127.0.0.1 7
```

Open another terminal windows

```
% telnet 127.0.0.1 7
```

Check out the logs

```
# tail /var/log/messages | grep inetd
```

TcpWrapper

Allow ssh from local network and deny the rest

```
# vi /etc/hosts.allow
```

```
ALL : localhost : allow
sshd: 192.168.0. : allow
ALL : ALL : deny
```

chroot

Simple chroot test

```
# ldd /bin/csh
/bin/csh:
        libncurses.so.5 => /lib/libncurses.so.5 (0x280b9000)
        libcrypt.so.2 => /lib/libcrypt.so.2 (0x280f8000)
        libc.so.5 => /lib/libc.so.5 (0x28110000)
```

Create directories

```
# mkdir /chroot
# mkdir /chroot/bin
# mkdir /chroot/lib
# mkdir /chroot/libexec
```

Copy the necessary libraries

```
# cp /lib/libncurses.so.5 /chroot/lib/
# cp /lib/libcrypt.so.2  /chroot/lib/
# cp /lib/libc.so.5 /chroot/lib/
# cp /libexec/ld-elf.so.1 /chroot/libexec/
```

Copy the csh binary

```
# cp /bin/csh /chroot/bin/
```

Now enter the chroot environment

```
# chroot /chroot/ /bin/csh
```

Notice that none of the commands work except csh built in commands

```
% ls -l
ls: command not found
% echo *
bin lib libexec
% set
```

rc.conf

```
# vi /etc/rc.conf
```

```
nfs_server_enable=”NO”
nfs_client_enable=”NO”
rpcbind_enable=”NO”
syslogd_enable=”YES”
sshd_enable=”YES”
inetd_enable=”NO”

accounting_enable=”YES”
clear_tmp_enable=”YES”
enable_quotas=”YES”
check_quotas=”YES”
```

Don't enable kernel secure level if running X

```
kern_securelevel_enable=”NO”
kern_securelevel=”3”
log_in_vain=”YES”
tcp_drop_synfin=”YES”
tcp_extensions=”NO”
accept_sourceroute=”NO”
forward_sourceroute=”NO”
icmp_drop_redirect=”YES”
icmp_log_redirect=”YES”
icmp_bmcastecho=”NO”
```

sysctl.conf

```
# vi /etc/sysctl.conf
```

```
net.inet.tcp.blackhole=2
net.inet.udp.blackhole=1
security.bsd.see_other_uids=0
net.inet.ip.check_interface=1
net.inet.tcp.syncookies=1
net.inet.ip.fw.verbose=1
net.inet.ip.fw.verbose_limit=200
net.inet.icmp.icmplim=200
net.inet.icmp.bmcastecho=0
```

Deny direct root logins on console and single mode

```
Login as root
# vi /etc/ttys
```

Change all “secure” to “insecure”

```
Login as root on the console
```

Syslog

Enable NTP

vi /etc/rc.conf

```
ntpdate_enable=YES
ntpdate_flag=”-b 192.168.0.1”
```

Remote logging

Add the following line to /etc/syslog.conf
Use the peer computer as the remote host and test sending syslog output to each other

```
# vi /etc/syslog.conf
*.*     @remote_host
```

Restart syslog client

```
# killall -HUP syslogd
```

Verify

```
# tail -f /var/log/messages
```

Note: Make sure that on the Syslog server the syslogd daemon is running without “-ss” option

Backup

tar

Backup your /etc/ directory via tar

```
% tar czvf etc.tar.gz /etc
```

List the files you have backed up

```
% tar -tzf etc.tar.gz
```

Restore the files to a test directory

```
% mkdir etc-test
% cd etc-test
% tar xzvf ../etc.tar.gz
```

CPIO

Backup your /etc/ using cpio

```
% find /etc | cpio -o > etc.cpio
```

List the files in the archive

```
% cpio -t < etc.cpio
```

Restore the files to a test directory

```
% mkdir etc-test
% cd etc-test
% cpio -idmv --no-absolute-filenames < ../etc.cpio
```

Using Rsync

#create a file /usr/local/etc/rsyncd.conf

```
read only = false
uid = nobody
gid = nobody
auth users = bckoper
secrets file = /usr/local/etc/rsyncd.secrets
use chroot = no
hosts allow = 127.0.0.1
hosts deny = *

[test]
comment = Test
path = /tmp/rsync
```

Create a password file /usr/local/etc/rsyncd.secrets

```
bckoper:98kd93k
```

```
# chmod 600 /usr/local/etc/rsyncd.secrets
# chown nobody: /usr/local/etc/rsyncd.secrets
# mkdir /tmp/rsync
# chown -R nobody: /tmp/rsync
```

Enable Rsync in inetd

```
# echo “rsync stream tcp nowait nobody /usr/local/bin/rsync rsync --daemon”
>> /etc/inetd.conf
# inetd
```

Rsync test

```
% setenv RSYNC_PASSWORD 98kd93k
% rsync -a --delete /home/ bckoper@127.0.0.1::test/
```

Using SSH and Rsync

```
# echo "sshd: 127.0.0.1" >> /etc/hosts.allow
# rsync -a --delete -e 'ssh' /etc 127.0.0.1:/home/vicky
```

System Resource Monitoring

uptime

```
%uptime
```

ps

%ps -aux

top

%top

netstat

Using netstat to look at all sockets

```
% netstat -an
```

Using netstat to look at current tcp sockets

```
% netstat -n -p tcp
```

sockstat

using sockstat to see Ipv4 listening sockets

```
# sockstat -4 -l
```

Process Accounting

sa

Per process stats

```
$ sa
```

Per user stats

```
$ sa -m
```

lastcomm

Per command basis

```
$ lastcomm ls
```

Per user basis

```
$ lastcomm username
```

File Integrity

AIDE

Install AIDE from ports

```
# vi /etc/make.conf
Add:
MASTER_SITE_OVERRIDE="http://192.168.0.1/ports/”
# cd /usr/ports/security/aide/
# vi Makefile
Add Hash before BROKEN (Just for testing purpose)
# make install
```

Initialize the database

```
# cd /var/db/aide
# vi aide.conf
Add:
# For this demo only, save time
# configuration sample is in /usr/local/etc/aide.conf.sample
database=file:///var/db/aide/databases/aide.db
database_out=file:///var/db/aide/databases/aide.db.new

/etc    R

# aide -i

# cd /var/db/aide/databases
# mv aide.db.new aide.db
```

Modify some system files

```
# touch /etc/passwd
# touch /etc/fstab
# touch /etc/newfile.txt
```

Check the database for changes

```
# aide -C
```

Add it to crontab for daily updates

```
0 3 * * * /usr/bin/aide -u
```

Make sure you have alias for root , as AIDE reports via emails to root from crontab

If changes are found it will write a new file /var/lib/aide/aide.db.new.gz and report via email to root or log to /var/log/aide.log.

Updating

If changes are expected you can commit the change by copying the aide.db.new.gz to aide.db.gz

Nmap

Install nmap from ports

```
# cd /usr/ports/security/nmap
# make install
```

Nmap ping sweep

```
# nmap -sP 192.168.0.0/24
```

Scan for ports

```
# nmap -v 192.168.0.1
```

Os detection and stealth scan

```
# nmap -sS -O -v 192.168.0.1
```

XMas scan

```
# nmap -sX -v 192.168.0.1
```

Output the result to a file

```
# nmap 192.168.0.1 -oN nmap.scan
```

Try nmap using different switches and find out who is the most vulnerable

TcpDump

`tcpdump -i xl0 src net 192.168.0.0/24`
Trace packets with source Ip from 192.168.0.0/24 network

`tcpdump -i xl0 -n src net 192.168.0.0/24`
Same as above but without host name translation

`tcpdump -i xl0 -nn src net 192.168.0.0/24`
Same as above but without port name translation

`tcpdump -i xl0 -n src net 192.168.0.0/24 and dst 192.168.0.1`
Trace packets with source Ip from 192.168.0.0/24 network and destination 192.168.0.1. "-i any" means listen on all available interfaces, only supported in Linux.

`tcpdump -i xl0 tcp`
Trace tcp packets coming in/ going out of interface xl0

`tcpdump -i xl0 ! arp`
Trace all packets coming in / going out of interface xl0 that is not arp packets.

`tcpdump -i xl0 -X`
Print the packets in both HEX and ASCII

There are lots of other options available with tcpdump

Nessus

Install Nessus and Nessus Plugins

```
# cd /usr/ports/security/nessus
# make install
# cd /usr/ports/security/nessus-plugins
# make install
```

Generate certificate and user

```
# /usr/local/sbin/nessus-mkcert
# /usr/local/sbin/nessus-adduser
```

Start the daemon

```
# /usr/local/nessus/sbin/nessusd -D
```

Regulary update the plugins

Add the following line to /etc/crontab

```
00 3 * * * root   /usr/local/sbin/nessus-update-plugins
```

Run the client

```
% /usr/local/bin/nessus
```

Enter the username and password you created above
Under "target selection" enter the ip address you want to scan for eg 192.168.0.1
Then click on "start the scan" which should take a while depending upon the number of plug ins you have enabled.

Report

You can either view the report in the report windows or save the report as various formats such as ascii, html, xml

Export it to html and it should save a directory named nessus.html. Use a browser to view the report.

Ntop

Install ntop from ports

cd /usr/ports/net/ntop
make install

Create ntop admin user

/usr/local/bin/ntop -A

Start ntop

You can leave the “-d” to see the output on the screen
/usr/local/bin/ntop -u nobody -d

Allow ntop from tcpwrapper

vi /etc/hosts.allow
Add:
ntop: 192.168.0.

View ntop in your Browser

http://ntop_server:3000
http://localhost:3000

Using GPG

Generate your Signing Key

```
% gpg --gen-key
```
* Select RSA (sign only)
* Enter key size 2048 (Sufficient)
* Enter the Expiry date for the Key 0 means no expiration set
* Enter Your Name
* Enter Your Email Address
* Type your pass phrase Don't use a Null pass phrase

Generate your Encrypt Key

% gpg --edit-key "your email id"
```
Command> addkey
```
* Select RSA (encrypt only)
* Enter keysize 2048
* Enter the Expiry date for the key 0 means no expiration set

```
% echo "Hello there" | gpg -e --default-recipient-self > example.gpg
```

Test the content of the file example.gpg
You don't want to use cat as this file is binary

```
% vi example.gpg or
% od -c example.gpg
```

Deccrypt your file

```
% cat example.gpg | gpg -d
```

Geometry Based Disk Encryption

Create directories for Testing

```
# mkdir /private
# mkdir /usr/local/disks/
```

Create a file sized 10M (will be used as virtual memory disk)

```
dd if=/dev/zero of=/usr/local/disks/cryptfs.img bs=1024K count=10
```

Create a virtual Memory disk

-a attach a memory disk
-t vnode -f cryptfs.img Use this file instead of memory
-u 9 request specific md number /dev/md9

```
# mdconfig -a -t vnode -u 9 -f /usr/local/disks/cryptfs.img
```

Initialize gbde (only required once, not needed on subsequent attach)

```
# kldload geom_bde
# gbde init /dev/md9 -i
```

Change sector_size=2048

Attach the disks to the kernel

Will prompt for a password each time when attaching an encrypted partition

```
# gbde attach /dev/md9
```

check

```
# ls -l /dev/md9.bde
```

Fill the drive with random data

```
dd if=/dev/random of=/dev/md9.bde bs=64k
```

Create New filesystem with newfs (Only required onced)

-U enable soft update
-O2 UFS2

```
# newfs -U -O2 /dev/md9.bde
```

Mount the drive

```
# mount /dev/md9.bde /private/
```

Check

```
# df
```

Create files and folders inside /private

Detach the encypted file

```
# umount /private
# gbde detach /dev/md9
```

Try Mounting the plain /dev/md9

```
# mount /dev/md9 /private
```

Detach the Memory disk

```
# mdconfig -d -u 9
```

SSH

Remote login

```
ssh username@remote
```

X forwarding

```
% xhost + local:
ssh -X username@remote
% env | grep DISPLAY
% xlogo
```

Compression

```
ssh -C username@remote
```

TCP Forwarding

-N Tunnel only mode
-f Run in background

```
% ssh -N -f -L local-port:remote-host:remote-port user@remote-host
% ssh -N -f -L 8080:192.168.0.1:80 user@192.168.0.2
```

SCP

```
scp filename.txt server:
```

sftp

```
sftp server_name
```

Public key based SSH

In Client:

```
% ssh-keygen -t dsa
```

Make sure you put a pass phrase

```
%v scp id_dsa.pub server:.ssh/
```

In server:

```
% cd ~/.ssh
% cat id_dsa.pub >> authorized_keys
```

In client:

```
% ssh username@server
```

Enter the pass phrase for the private key instead of the password of the remote server

IPSEC

Compile Kernel with IPSEC support

```
# cd /usr/src/sys/i386/conf/
# cp GENERIC IPSEC_TEST
# vi IPSEC_TEST
Change ident = IPSEC_TEST
Add:
options IPSEC
options IPSEC_ESP
# cd /usr/src
# make buildkernel KERNCONF=IPSEC_TEST
# make installkernel KERNCON=IPSEC_TEST
```

Install Racoon

```
# cd /usr/ports/security/racoon
# setenv  DISABLE_VULNERABILITIES (Just for the LAB)
# make install
```

Configure the Pre-shared key file

```
# vi /usr/local/etc/racoon/psk.txt
192.168.0.1 verylongpassword
# chmod 600 /usr/local/etc/racoon/psk.txt
```

Racoon Configuration

```
# vi /usr/local/etc/racoon/racoon.conf
path pre_shared_key "/usr/local/etc/racoon/psk.txt";
path certificate "/usr/local/etc/racoon/certs";

remote 192.168.0.1 {
      exchange_mode main;
      proposal {
            encryption_algorithm 3des;
            hash_algorithm md5;
            authentication_method pre_shared_key;
            dh_group modp1024;
      }
}

sainfo address 192.168.0.253 any address 192.168.0.1 any {
      pfs_group modp768;
      encryption_algorithm 3des;
      authentication_algorithm hmac_md5;
      compression_algorithm deflate;
}
```

Setkey Configuration

```
# vi /usr/local/etc/racoon/ipsec.conf
```

```
#!/usr/sbin/setkey -f

# flush
flush;
spdflush;

# create policies
spdadd 192.168.0.253 192.168.0.1 any -P out ipsec
      esp/transport//require;

spdadd 192.168.0.1 192.168.0.253 any -P in ipsec
      esp/transport//require;
```

Initialize the security database

```
# setkey -f /usr/local/etc/racoon/ipsec.conf
```

SAD Security Association Database
SPD Security Policy Database

Verify the setup of the SPD entries

```
# setkey -DP
```

Verify the entries of SAD entries after successful IPSEC connection

```
# setkey -D
```

Make sure you allow the esp and ah protocol and IKE port

```
# ipfw add allow esp from any to me
# ipfw add allow esp from me to any
# ipfw add allow udp from any to me 500
```

Start racoon in foreground mode

```
# racoon -F -f /usr/local/etc/racoon/racoon.conf
```

From another terminal ping the remote host

```
# ping 192.168.0.1
```

Use tcpdump to verify

From another terminal use tcpdump to verify that you are indeed transmitting packets using IPSEC

```
# tcpdump  -nn
```

ipfw

Enable IPFW in rc.conf

```
# vi /etc/rc.conf
firewall_enable="YES"
firewall_script="/etc/rc.firewall"
firewall_logging=”YES”
```

Insert a test rule

```
# ipfw add allow ip from any to me
```

Listing Firewall Rules

```
# ipfw -a l
```

Delete the rule

```
# ipfw delete 100
```

Allow Rule for ssh

```
# ipfw add allow tcp from any to me 22
```

Deny Rule for telnet

```
# ipfw add deny tcp from any to me 23
```

Deny and Log rules

```
# ipfw add deny log logamount 200 ip from any to any
```

Flush rules

```
# ipfw flush
```

State rules

```
# ipfw add check-state
# ipfw add allow icmp from me to any keep-state
```

Listing Dynamic Rules

```
# ipfw -d l
```

Listing Dynamic expired Rules

```
# ipfw -d -e l
```

Clearing Counters

```
# ipfw zero
```

Accept all packets from interface loopback

```
# ipfw add allow ip from any to any via lo0
```

Allow outgoing packets

```
# ipfw add allow ip from me to any out
```

Allow incoming connections to port 3000

```
# ipfw add allow tcp from 192.168.0.0/24 to me 80 in
```

Allow new connections from this box

```
# ipfw add allow tcp from me to any setup out keep-state
```

Ratelimiting incoming requests to port 80 on the basis of source address

Only allow 10 connections from one host

```
# ipfw add allow tcp from any to me 80 limit src-addr 10
```

Build a firewall

```
# mv /etc/rc.firewall /etc/rc.firewall.old
# vi /etc/rc.firewall
```

```
#!/bin/sh
## vickysh@wlink.com.np
## 18 July 2005

FW="/sbin/ipfw -q"

#Flush the rules
${FW} -f flush

#Allow from Interface loopback
${FW} add allow all from any to any via lo0

# Check-state
${FW} add check-state

# Allow setup of outgoing TCP connections
${FW} add allow tcp from me to any setup out keep-state

# Allow udp from this host
${FW} add allow udp from me to any out keep-state

# Allow ssh from local network
${FW} add allow tcp from 192.168.0.0/24 to me 22

# Allow icmp
${FW} add allow icmp from me to any keep-state

# Deny echo-request and allow the rest
${FW} add deny icmp from any to me icmptypes 8
${FW} add allow icmp from any to me
```

Make it executable

```
# chmod 755 /etc/rc.firewall
```

Run the firewall

```
# /etc/rc.firewall
```

chkrootkit

Install chkrootkit from ports

```
# cd /usr/ports/security/chkrootkit
# make install
```

Run Check Root kit

```
# chkrootkit
```

Snort

Prerequisites

```
mysql server
apache server with php
```

Install Mysql Server

```
# cd /usr/ports/databases/mysql40-server
# make install
```

Install snort from ports

```
# cd /usr/ports/security/snort
# make install
```

Select MySQL Support

Install Apache httpd with php

```
# cd /usr/ports/www/apache13
# make install
# cd /usr/ports/databases/php4-mysql
# make install
De select IPV6
# cd cd /usr/ports/www/php4-session
# make install
```

Start apache

```
# vi /etc/rc.conf
Add:
apache_enable=YES
# /usr/local/etc/rc.d/apache.sh start
```

Start Mysql server

```
#vi /etc/rc.conf
Add:
mysql_enable=YES
# /usr/local/etc/rc.d/mysql-server.sh start
```

Secure Mysql Installation

```
# mysql -u root
drop database test;
use mysql;
delete from columns_priv;
delete from db;
delete from func;
delete from host;
delete from tables_priv;
delete from user where User='';
update user set Password=password('1mysql90admin') where User='root';
flush privileges;
exit
```

Create Database for Snort

```
# cp /usr/ports/security/snort/work/snort-2.3.2/schemas/create_mysql /tmp
# mysql -u root -p
create database snort ;
grant all on snort.* to snort@localhost identified by '98kd97';
```

Import snort schema into mysql

```
% mysql -u snort -p
> use snort;
> \. /tmp/create_mysql
> show tables;
```

Add this to /usr/local/etc/snort.conf

```
output database: log,mysql,user=snort password=98kd97 dbname=snort
host=localhost
```

Starting snort

```
# vi /etc/rc.conf
ADD:
snort_enable=YES
snort_flags=”-D -u nobody”
# mkdir /var/log/snort
# chown nobody: /var/log/snort
# /usr/local/etc/rc.d/snort.sh start
# tail -f /var/log/snort/alert
```

Now try to nmap to your machine from other machine and you should see snort
detect these attacks.

Analysis Console for Intrusion Databases (ACID)

Install ACID from ports

```
# cd /usr/ports/security/acid
# make install
# vi /usr/local/www/acid/acid_conf.php
```

Update the following fields

```
$DBlib_path = "/usr/local/share/adodb";

$alert_dbname   = "snort";
$alert_host     = "localhost";
$alert_port     = "";
$alert_user     = "snort";
$alert_password = "98kd97";
```

For testing make symlink to main apache directory root

```
# cd /usr/local/www/data
# ln -s ../acid acid
```

Fireup a browser

```
http://localhost/acid/
```

Go to Setup page
Create ACID AG
Then go to the Main page