CSCE 465: Computer &Network Security (lab overview)

Instructor: Sungmin Kevin Hong

Agenda

- Part 1: Virtual Machine
- Part 2: Linux Programming
- Part 3: Libpcap Programming
- Part 4: Raw Socket Programming

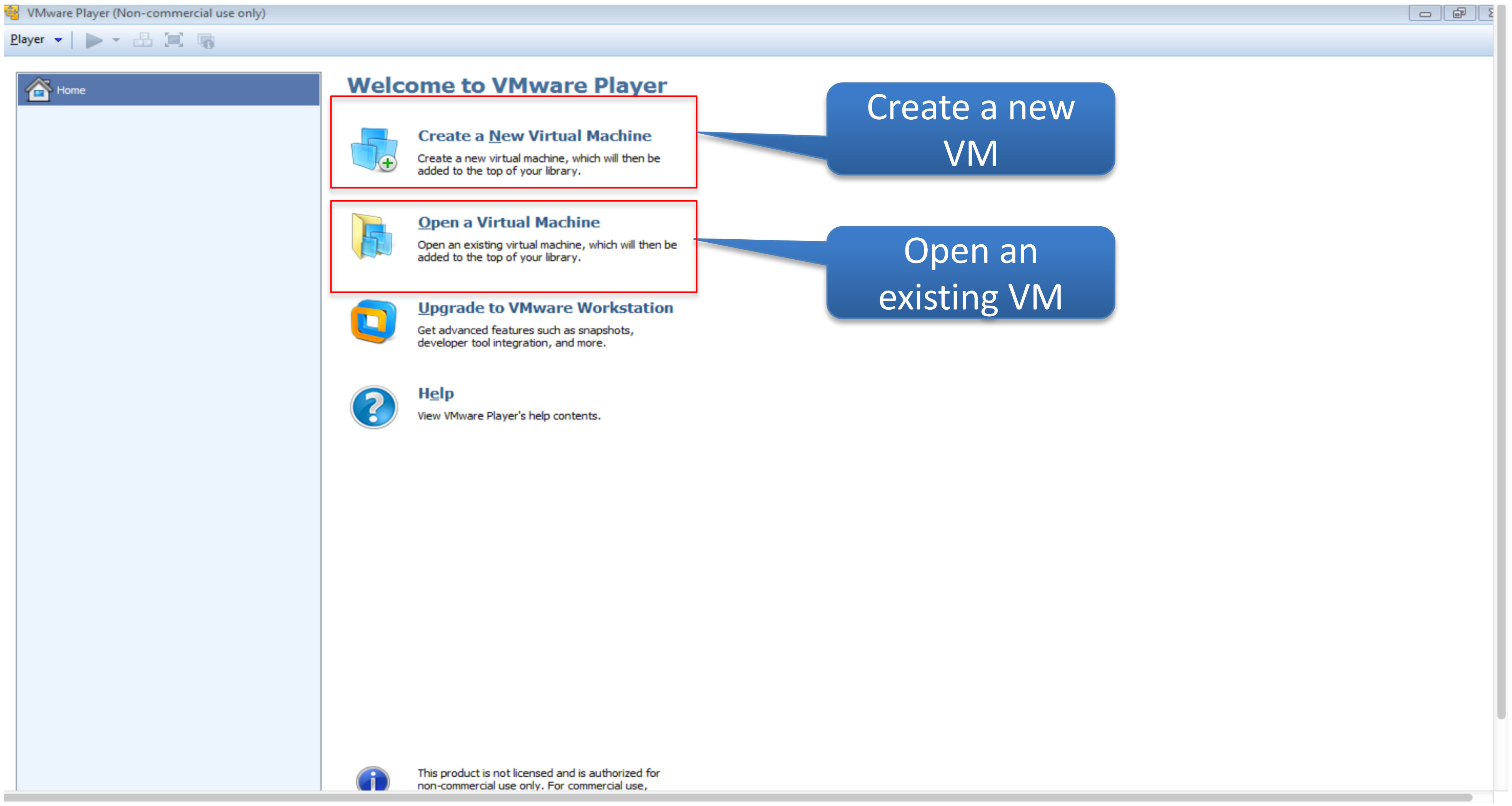
Part 1: Virtual Machine

- Definition: A virtual machine (VM) is a software implementation of a machine (for example, a computer) that executes programs like a physical machine. [wiki]
- Two Recommended Free Virtual Machines
  - VMWare Player (support Windows, Linux)
  - Virtual Box (Support Windows, Linux, Mac)

VMWare Player

- Free Software
- Run multiple OSes at the same time on your PC
- Host OS: Windows 8, Windows 7, Chrome OS, Linux
- Homepage:
  - http://www.vmware.com/products/player/
- Download
  - https://my.vmware.com/web/vmware/free#desktop_end_user_computing/vmware_player/5_0

Setup VM



VMware Player (Non-commercial use only)

Player

Home

Welcome to VMware Player

Create a New Virtual Machine
Create a new virtual machine, which will then be added to the top of your library.

Open a Virtual Machine
Open an existing virtual machine, which will then be added to the top of your library.

Upgrade to VMware Workstation
Get advanced features such as snapshots, developer tool integration, and more.

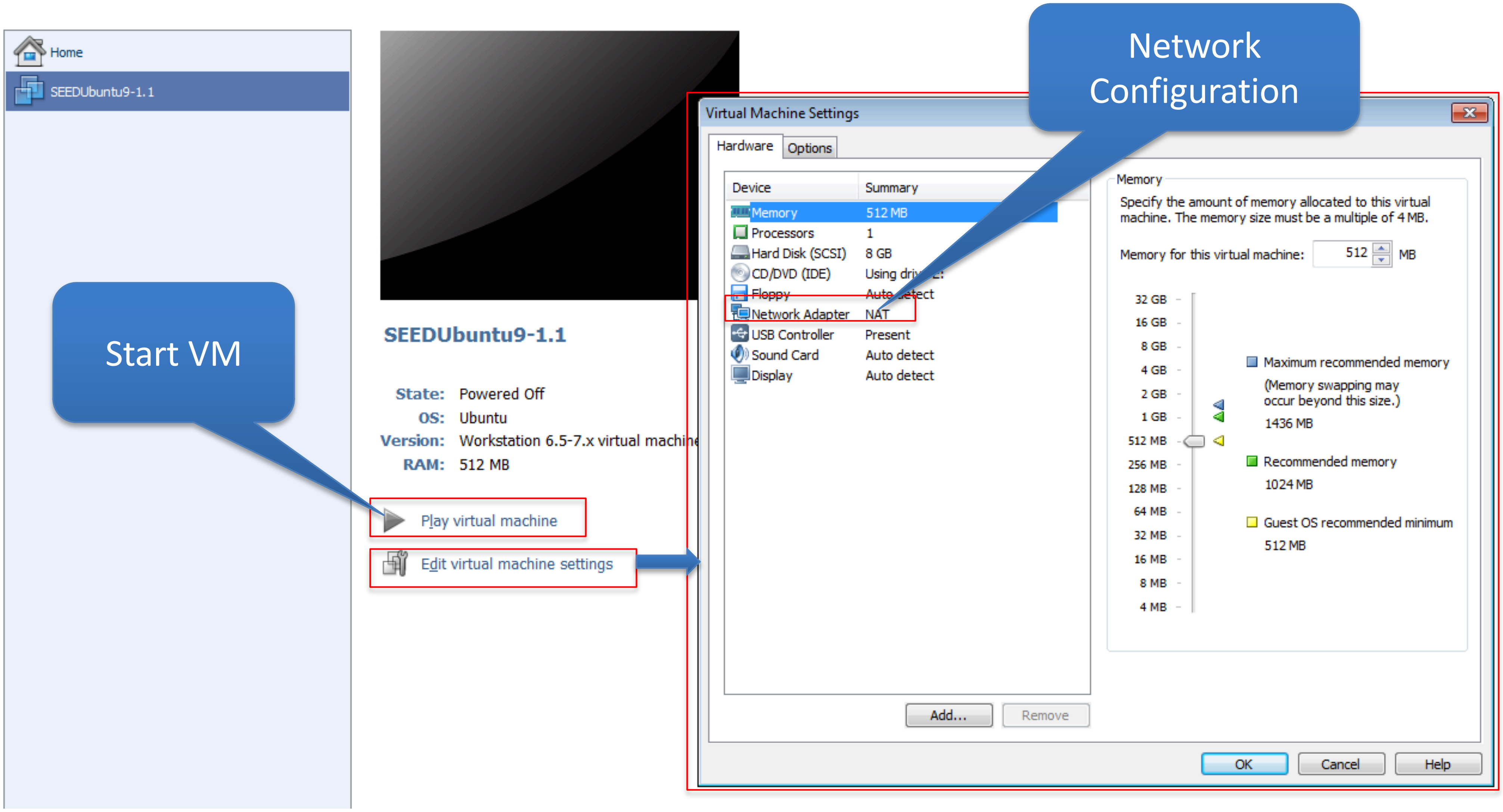
Help
View VMware Player's help contents.

This product is not licensed and is authorized for non-commercial use only. For commercial use,

Create a new VM

Open an existing VM

Network Configuration



SEEDUbuntu9-1.1

State:	Powered Off
OS:	Ubuntu
Version:	Workstation 6.5-7.x virtual machine
RAM:	512 MB

Virtual Machine Settings

Device	Summary
Memory	512 MB
Processors	1
Hard Disk (SCSI)	8 GB
CD/DVD (IDE)	Using drive E:
Floppy	Auto detect
Network Adapter	NAT
USB Controller	Present
Sound Card	Auto detect
Display	Auto detect

Memory

Specify the amount of memory allocated to this virtual machine. The memory size must be a multiple of 4 MB.

Memory for this virtual machine: 512 MB

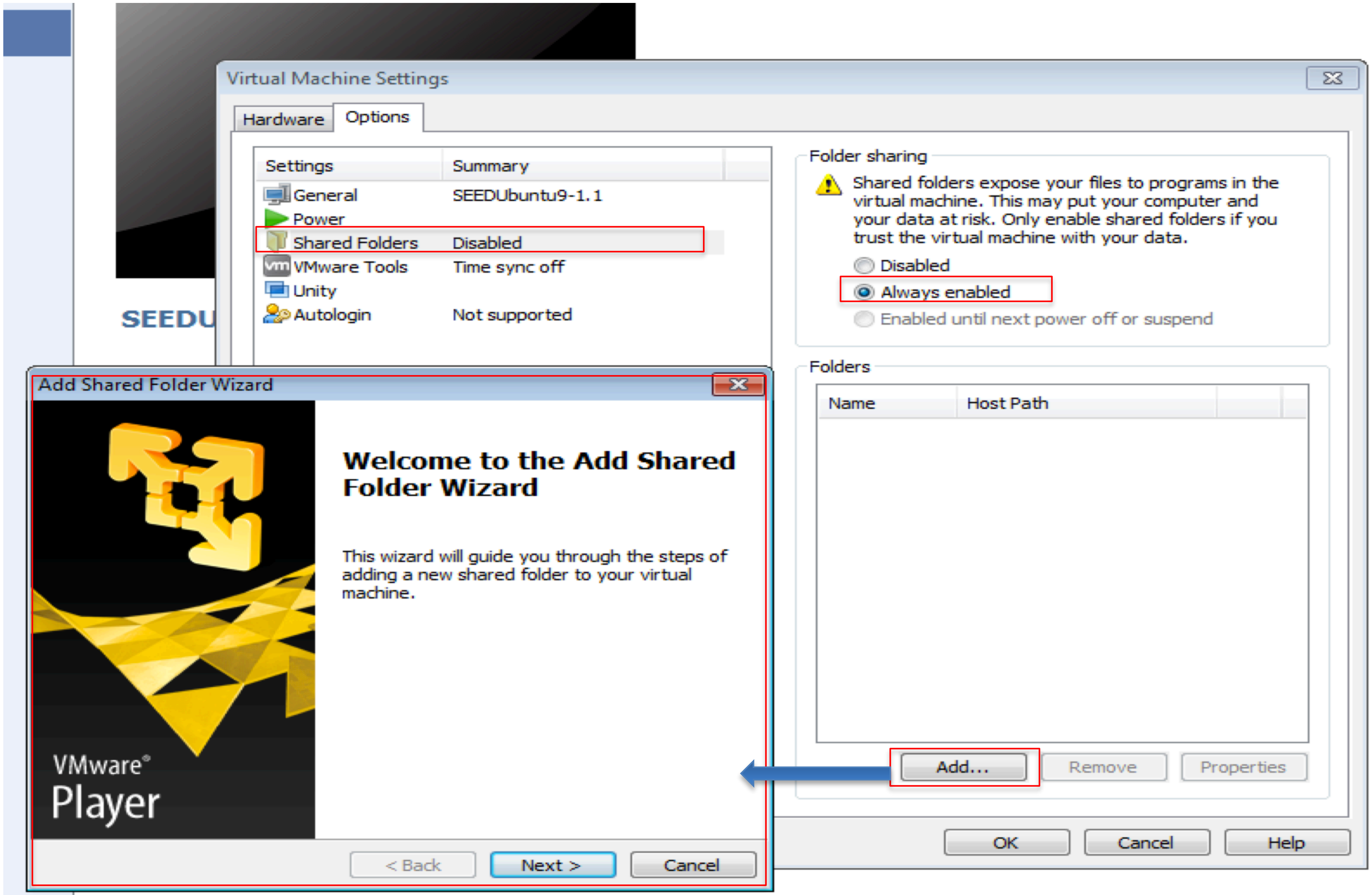
32 GB, 16 GB, 8 GB, 4 GB, 2 GB, 1 GB, 512 MB, 256 MB, 128 MB, 64 MB, 32 MB, 16 MB, 8 MB, 4 MB

Maximum recommended memory (Memory swapping may occur beyond this size.) 1436 MB

Recommended memory 1024 MB

Guest OS recommended minimum 512 MB

File Sharing with the host



Virtual Machine Settings

Hardware | Options

Settings	Summary
General	SEEDUbuntu9-1.1
Power	
Shared Folders	Disabled
VMware Tools	Time sync off
Unity	
Autologin	Not supported

Folder sharing

Shared folders expose your files to programs in the virtual machine. This may put your computer and your data at risk. Only enable shared folders if you trust the virtual machine with your data.

( ) Disabled
[✓ Always enabled]
( ) Enabled until next power off or suspend

Folders

Name	Host Path

Add... | Remove | Properties

OK | Cancel | Help

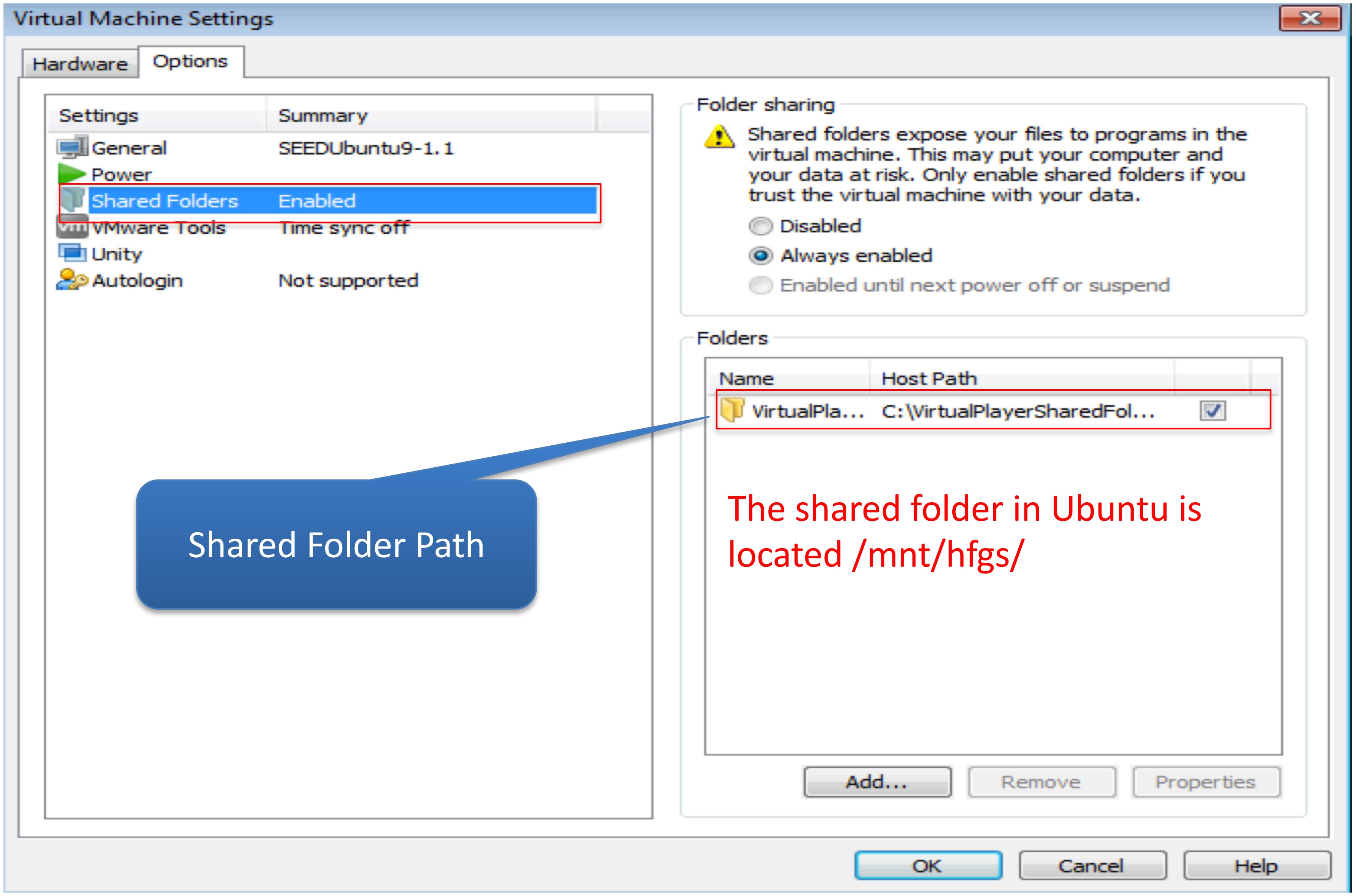
Add Shared Folder Wizard

Welcome to the Add Shared Folder Wizard

This wizard will guide you through the steps of adding a new shared folder to your virtual machine.

< Back | Next > | Cancel

Contd.


Virtual Machine Settings

Hardware | Options

Settings	Summary
General	SEEDUbuntu9-1.1
Power	
Shared Folders	Enabled
VMware Tools	Time sync off
Unity	
Autologin	Not supported

Folder sharing

Shared folders expose your files to programs in the virtual machine. This may put your computer and your data at risk. Only enable shared folders if you trust the virtual machine with your data.

- Disabled
- [✓ Always enabled]
- Enabled until next power off or suspend

Folders

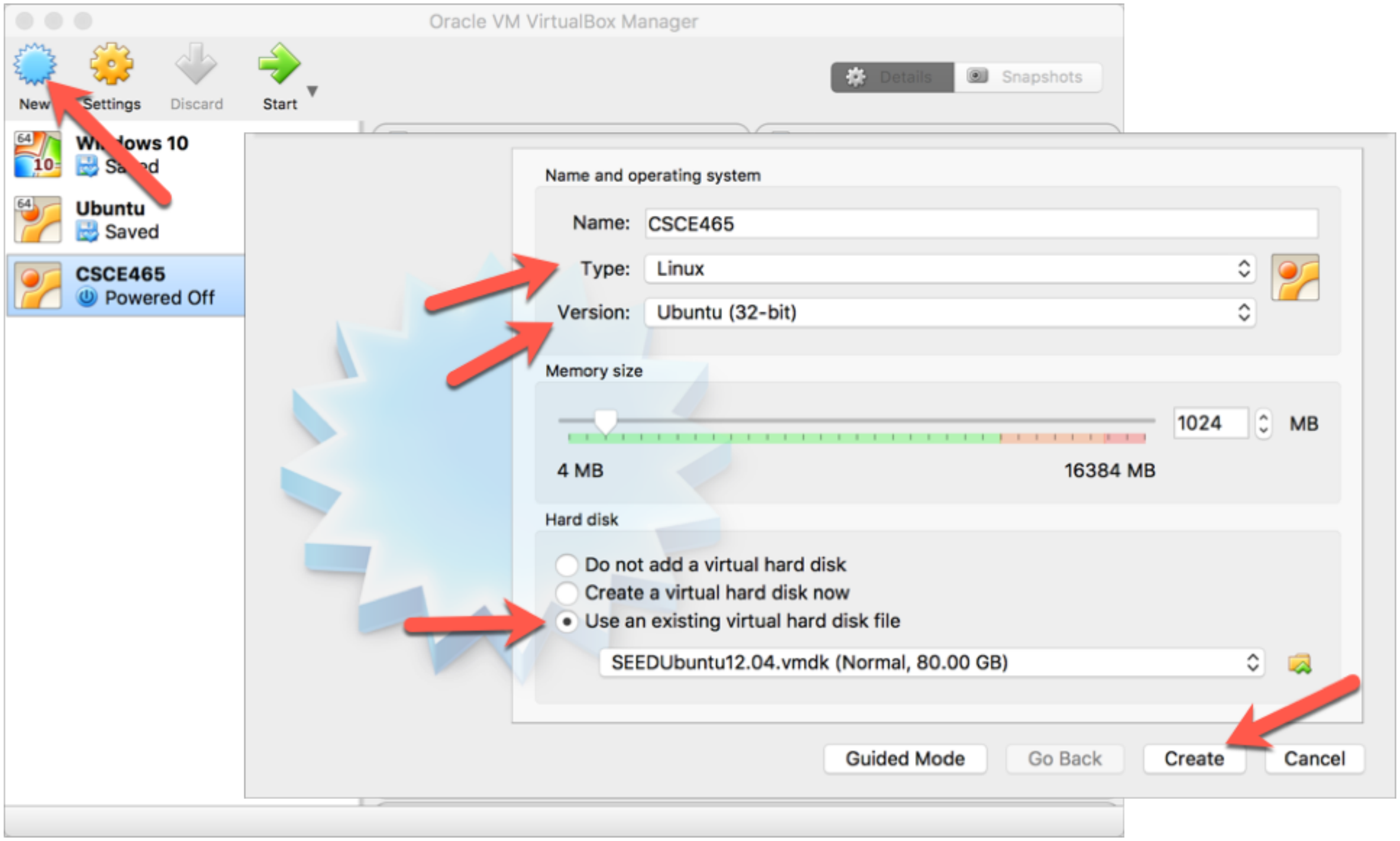
Name	Host Path	
VirtualPla...	C:\VirtualPlayerSharedFol...	✓

The shared folder in Ubuntu is located /mnt/hfgs/

Virtual Box

- Free Software
- Run multiple OS at the same time on your PC
- Host OS: Windows, Linux, Mac OS
- Homepage:
  - https://www.virtualbox.org/
- Download
  - https://www.virtualbox.org/wiki/Downloads

Setup VM



Name and operating system

Name: CSCE465

Type: Linux

Version: Ubuntu (32-bit)

Memory size

1024 MB

4 MB 16384 MB

Hard disk

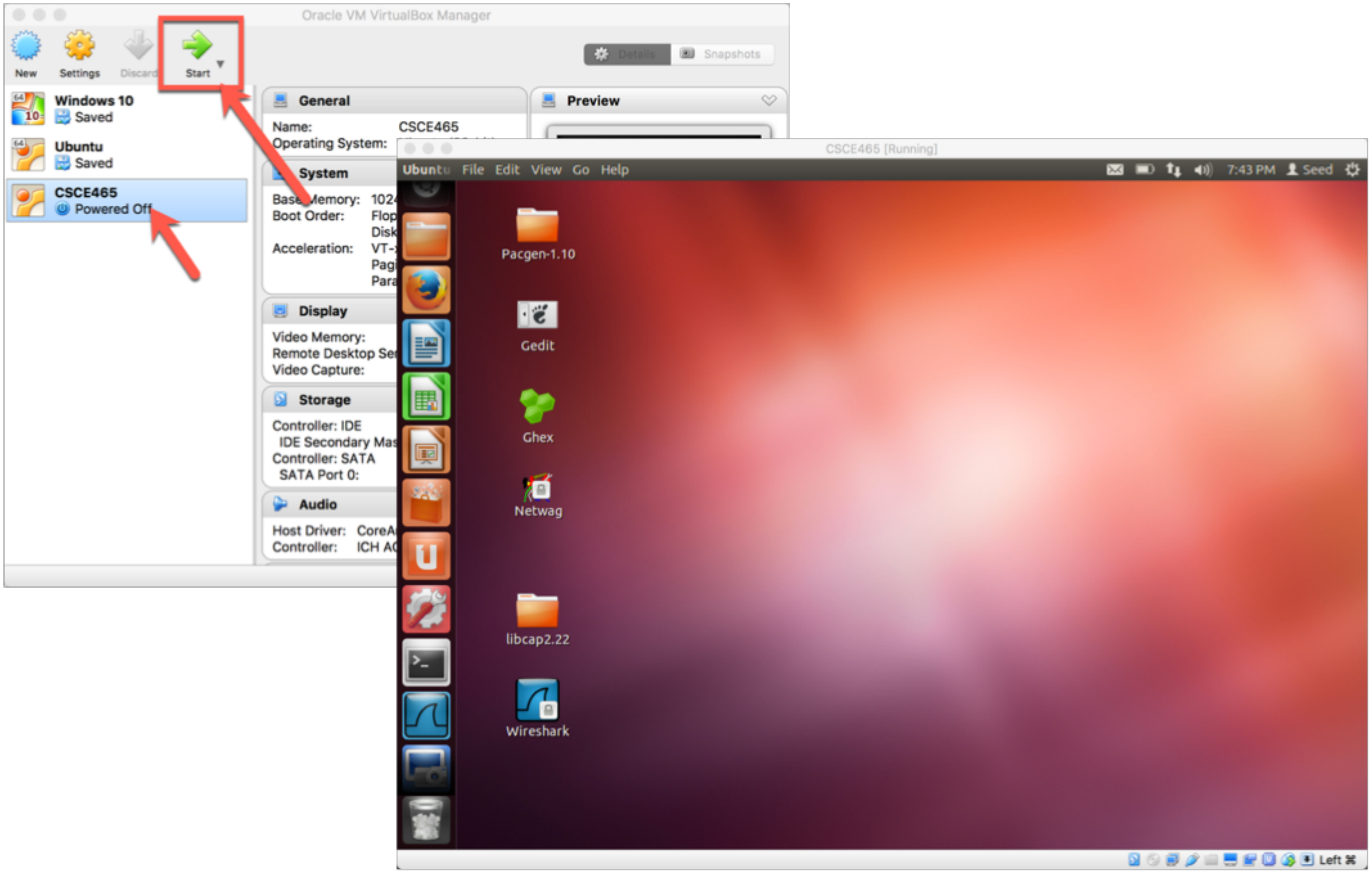
( ) Do not add a virtual hard disk

( ) Create a virtual hard disk now

(●) Use an existing virtual hard disk file

SEEDUbuntu12.04.vmdk (Normal, 80.00 GB)

Run VM



Name: CSCE465
Operating System:
Base Memory: 102
Boot Order: Flop Disk
Acceleration: VT- Pag Para
Video Memory:
Remote Desktop Se
Video Capture:
Controller: IDE
IDE Secondary Mas
Controller: SATA
SATA Port 0:
Host Driver: CoreA
Controller: ICH AC

Part 2: Linux Programming Basics

- Common Unix/Linux Commands
  - ls – list files in current directory (ignores files that are 'invisible')
  - ls -a – List all the files
  - cd *bob* – change directory to bob folder
    - cd .. (jumps one level up in directory)
  - mkdir *filename* – makes a folder of given filename
  - rm *blah* – removes file
  - rm **.ext* – removes everything in current directory of a given extension ext
  - pwd – lists the path of the current directory
- other commands can be found at https://wiki.cse.tamu.edu/index.php/Basic_UNIX_Commands

Compiling and Executing

- For C program
  - gcc *filename.c* - compiles and links c program, generating an executable file
- For C++ program
  - g++ *filename.cpp* - compiles and links c++ program, generating an executable file
- Options for both
  - '-o' –renames the executable, thus your executable no longer must go under the a.out name
  - More options can visit: https://gcc.gnu.org/onlinedocs/gcc/Option-Summary.html
- Run the program
  - ./a.out

Makefile

- Makefiles are special format files that together with the *make* utility will help you to automagically build and manage your projects

- For a simple tutorial, you can visit:
  - http://mrbook.org/blog/tutorials/make/

Tools and Useful Reference

- C/C++ program IDE:
  - CodeBlock http://www.codeblocks.org/
  - Eclipse http://www.eclipse.org/
- Linux Programming References:
  - **[**Richard Stevens**]UNIX Network Programming**
  - [Neil Matthew] **Beginning Linux Programming**

Part 3: Libpcap Programming

- ***pcap*** is a user-level interface for user-level packet capture
- ***libpcap*** provides C language Application Programming Interface (API)s for network statistics collection, security monitoring, network debugging, etc.
- "Wrappers" for **pcap** have been developed to support other programming languages, such as: ***pylibpcap*** for python, ***jNetpcap*** for Java, **scapy** for python

Installing libpcap

- Linux:

  for Ubuntu user:

  Commandline Installation:

  ```
  sudo apt-get install libpcap-dev
  ```

- Compiled from source:

  http://sourceforge.net/projects/libpcap/

Working with libpcap

- Compile program using libpcap
  - *gcc sniffex.c –lpcap –o sniffer*
- When run sniffer, you need root privilege.
  - *sudo ./sniffer*
- Next, I will introduce some important methods for libpcap programming.

Sniffex: simple packet sniffer for ICMP/UDP/TCP

(Src: https://www.tcpdump.org/pcap.html)

Important libpcap methods.

- Ask pcap to *find a valid device* to sniff

  dev = ***pcap_lookupdev****(errbuf);*
- Open *live* device for sniffing w/ ***promiscuous*** mode

  desc = ***pcap_open_live****(dev,BUFSIZE,0,-1,errbuf)*
- Open *offline* pcap file

  handle = ***pcap_open_offline****(file_path, errbuf);*
- Capture a single packet

  packet = ***pcap_next****(desc, &hdr)*
- Capture multiple packets w/ *#captures*, *callback*

  pcap_loop*(descr,****-1****,callback,NULL);*
- Close the live device

  pcap_close*(desc);*

Writing a packet capture program

- Main Event Loop

```
void my_callback(u_char *useless,const struct
   pcap_pkthdr* pkthdr,const u_char*   packet) {
     //do stuff here with packets
}

int main(int argc, char **argv) {
   //open and go live

   pcap_loop(descr,-1,my_callback,NULL);
   return 0;
}
```

Filtering the Packets.

- Filter Traffic: we don't need to see every packet!
- Compile the *filter w/ expression string*

*int **pcap_compile**(pcap_t *p, struct bpf_program *fp, char ***str**, int optimize, bpf_u_int32 netmask)*

- Activate the filter

*int **pcap_setfilter**(pcap_t *p, struct bpf_program *fp)*

**See https://www.tcpdump.org/manpages/pcap-filter.7.html for more information on filters.*

Part 4: Raw Socket Programming

- Raw Socket is an internet socket that allows direct sending and receiving of IP packets **without** any protocol-specific transport layer formatting.
- Absolute control over the packet construction, allowing programmers to construct any arbitrary packet, including setting the header fields and the payload.
- The ability to craft packet headers is a powerful tool that allows hackers to do many nefarious things.
- MUST be done by a **super user!**.

Basic Steps

1. Create a raw socket
2. Set socket options
3. Construct the packet
4. Send out the packet through the raw socket

** There are many online tutorials for reference.*
Check out essential APIs for raw socket!

1. Create

- Create a raw socket with UDP protocol

*sd = **socket**(PF_INET, SOCK_RAW, IPPROTO_UDP);*

2. Fabricate network packets

- Create crafted packet (**UDP** for example)

```
struct ipheader *ip = (struct ipheader *) buffer;
struct udpheader *udp = (struct udpheader *) (buffer +
    sizeof(struct ipheader));
```

- Fabricate the **IP header**

```
ip->iph_ihl = 5;
ip->iph_ident = htons(54321);
ip->iph_ttl = 64; // hops
ip->iph_protocol = 17; // UDP
// Source IP address, can use spoofed address here!!!
ip->iph_sourceip = inet_addr(argv[1]);
// The destination IP address
ip->iph_destip = inet_addr(argv[3]);
```

Cont.

- Fabricate the **UDP** header

/ /source port number

udp->udph_srcport = ***htons****(atoi(argv[2]));*

// Destination port number

udp->udph_destport = ***htons****(atoi(argv[4]));*

// Calculate the **checksum** for integrity

ip->iph_chksum = csum((unsigned short *)buffer, sizeof(struct ipheader) + sizeof(struct udpheader));

3. Send

- Send the crafted packet with raw socket

sendto*(sd, buffer, ip->iph_len, 0, (struct sockaddr *)&sin, sizeof(sin)) < 0*

Homework submission:
Single file as **UIN-homework1.zip**

Descriptive, self-contained with your own thoughts/insights, as it requests