University of Dayton
Department of Computer Science

CPS 475/575
Secure Application Development

Lecture 1 - Course Overview

Phu Phung
1/14/2020

Welcome and pre-class survey

- Welcome 17 seniors (10 CPS, 3 CIS, 4 CPE) and 12 graduate students to this Secure Application Development-Spring 2020.

- Please finish the survey as homework and turn it back in the next class

Today's agenda

- Overview of this "Secure Application Development" course
- Introduction to Internet Application
- Demo for Peer Instruction

Syllabus and Tentative Schedule

Available on Isidore

- Syllabus: http://bit.ly/secad-s20-syllabus
- Schedule: http://bit.ly/secad-s20-schedule

What is an Application?

- A short form of “Application Software” or “Application Program”
  - A computer program that normally interacts with end-users or other programs (vs. system software)
- Types of Applications
  - Computer-based (i.e., Desktop, Laptop, Mobile…)
    - Stand-alone apps, e.g., calculator
    - Internet-based Apps, e.g., Skype…
  - Web-based
    - Web apps, e.g., www.facebook.com
      - Run within another application, i.e., web browsers

Application Development

- Based on a typical software engineering process
  - Normally developers focus on the functionalities of the application

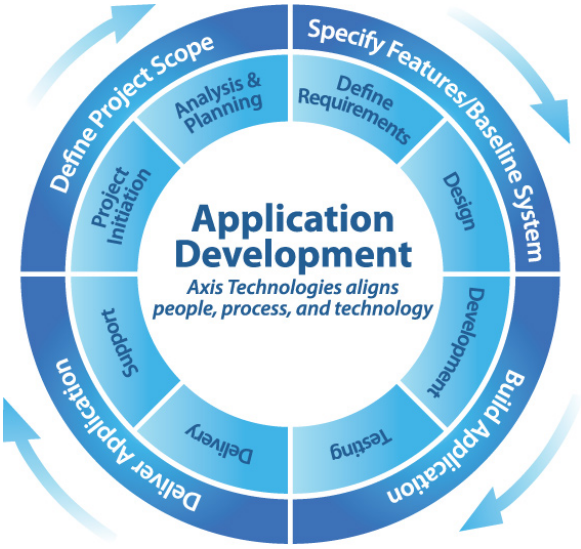


Image source: www.pinterest.com

Application/Software Security

- Vulnerabilities in software that cause cyber attacks
  - A vulnerability is a flaw in software mostly caused by software developers

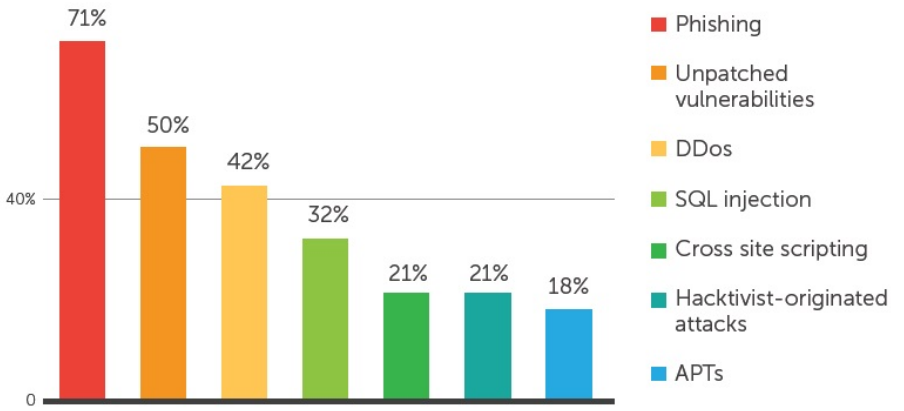
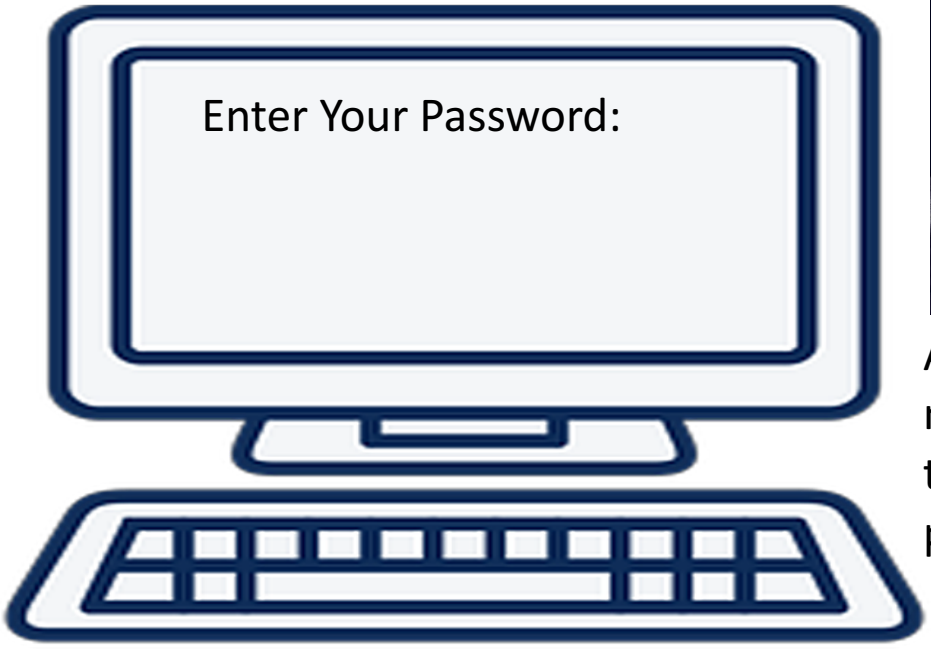


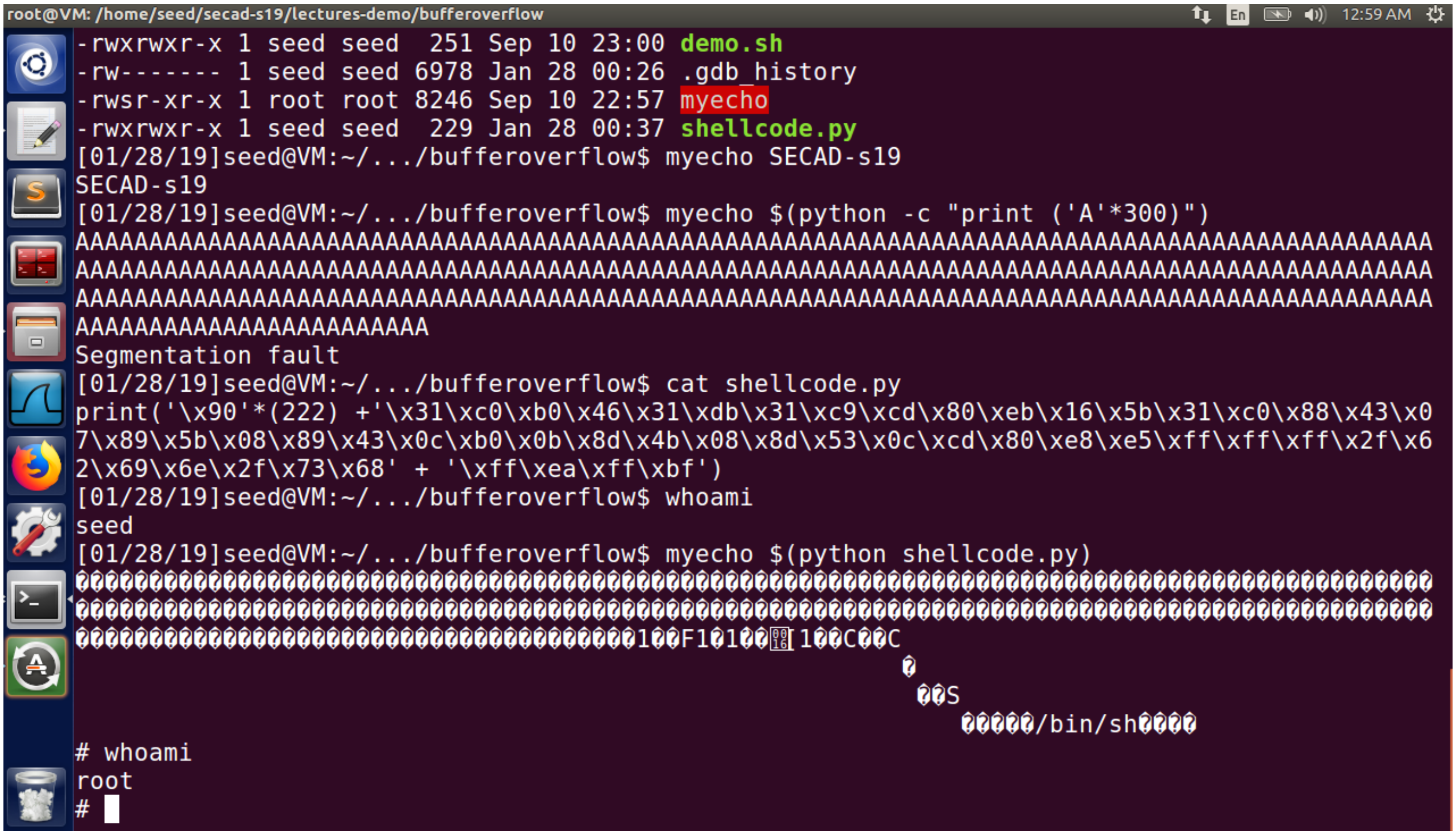
Chart source: Trend Micro and the Organization of American States (OAS)

A Vulnerability Example: Buffer-overflow



Attacker can inject malicious code from input to exploit vulnerable programs

Buffer Overflow Attack Demo



Demo video: https://youtu.be/w-yTzd7DI_s

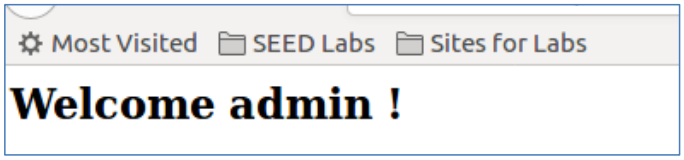
Why does Buffer-Overflow happen?

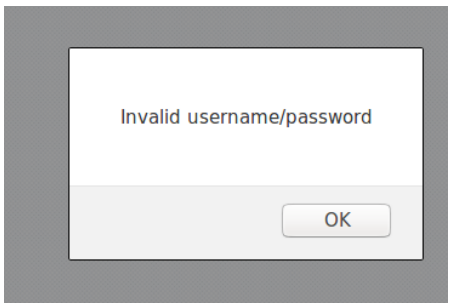
- Data is read or written outside the bounds of a buffer
  - The attacker gets the program to treat the malicious data as code
- How can this happen?
  - The developer did not check the data size or use insecure functions in the program!!!
    - The developer did not think like a hacker!!!

Another example: a web login

- Users are asked to provide username/password to be able to login (authenticated)

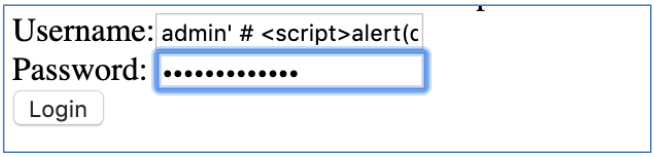
- The system will get provided data, connect to a database to verify the credentials
  - You will be logged in with correct username/password
  - Will be denied otherwise

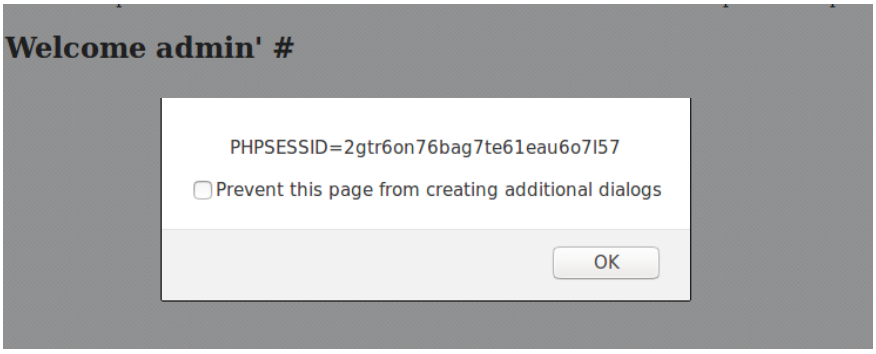




A web login attack scenario

- Let's think as a hacker
  - Is there any way that can bypass the database check?





- Another example of buffer overflow attack on the web: XSS and SQLi
  - Security issue: The inputs are used without validation

Security in Software/Application Development

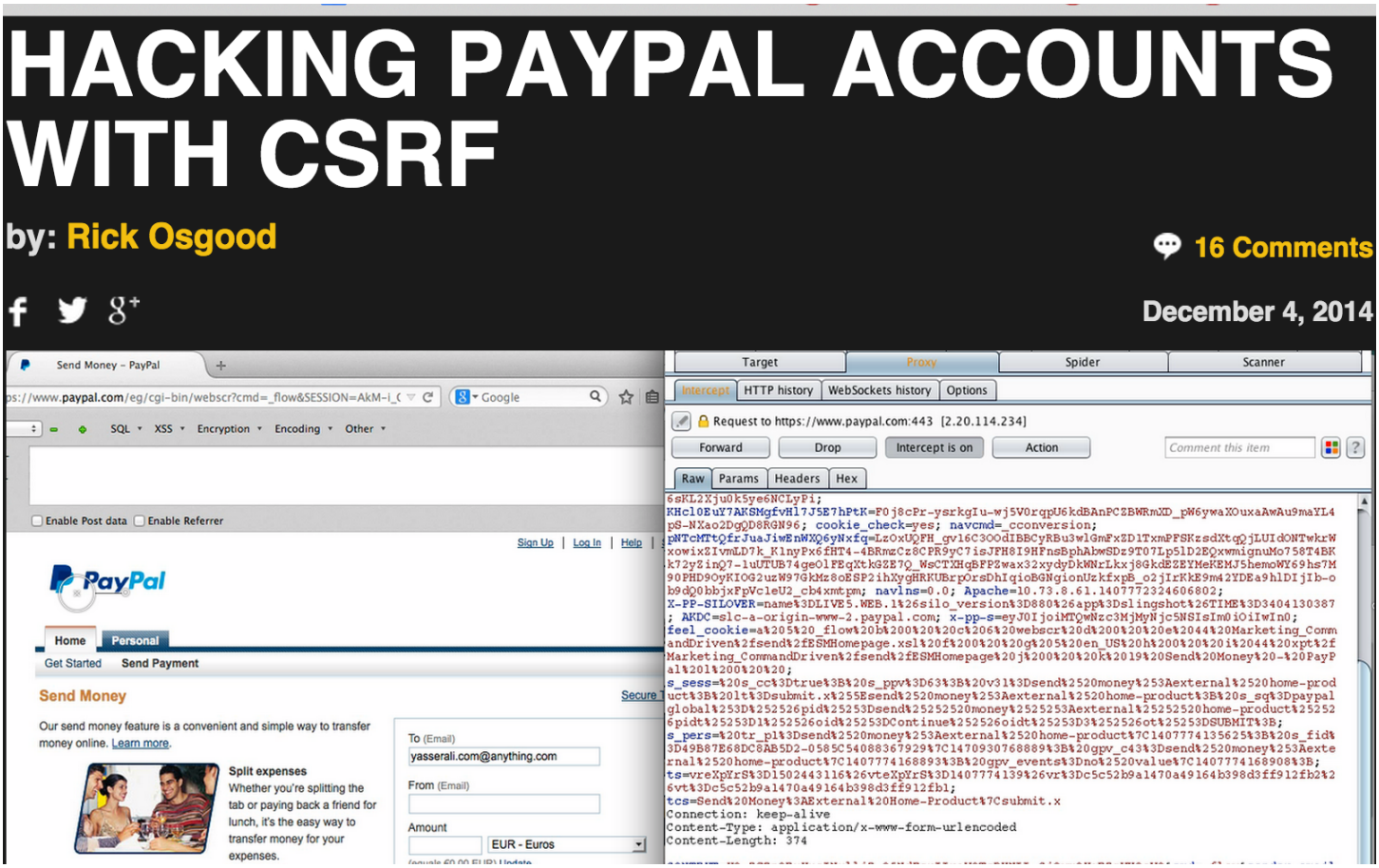
- Developers normally focus on the functionalities
  - Few developers know how to develop secure software
    - Programming books/courses do not teach it
  - Most developers do not think like a hacker
    - "How could this be attacked?" – be slightly paranoid
  - Developers do not learn from others' security mistakes
    - **Most vulnerabilities caused by same mistakes over 40+ years**
    - Focus on learning common errors… so you won't make them

Credit: David A. Wheeler

Yet Another Example

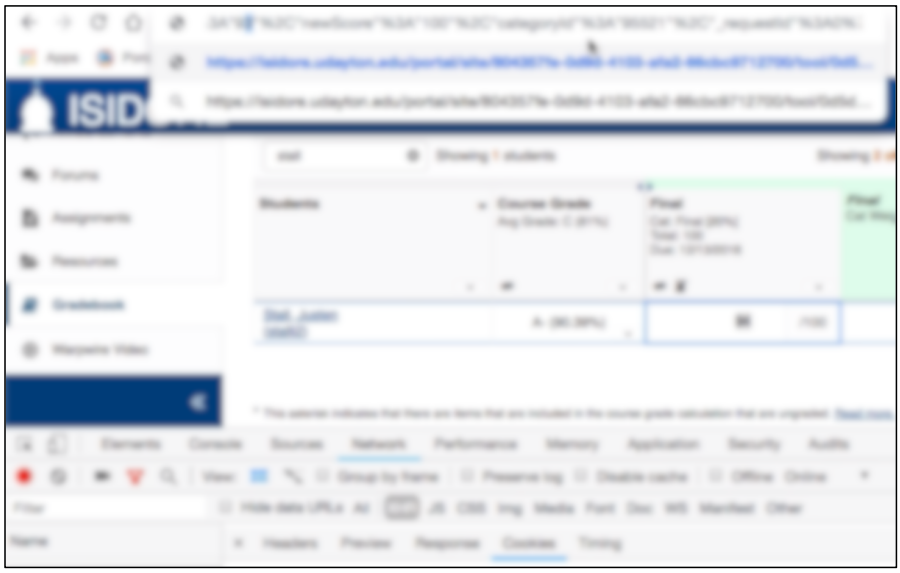
- Assume that you are a Paypal user, and assume that Paypal requires two-factor authentication, i.e., after providing username/password, you are required to confirm the login in another device
  - This mechanism prevents someone have your username/password to login to the system
- Discussion: is it safe for you to open a link like below after you logged in?
  https://www.paypal.com/eg/cgi-bin/cmd=flow&SESSION=Akl-tATMf1GOP-tQu3t3x4Vju&…

PayPal is vulnerable to CSRF



A Scenario on an open-source Web Application

- https://www.sakailms.org/ is an open-source learning management system (LMS)
  - Do you think is there any scenario on Sakai that is similar to CSRF attacks presented previously?
    - Let's see a video example



Who should be responsible for the Paypal/Sakai attack example?

- The user?
  - e.g., using anti-virus software?
- Your organization?
  - E.g., using a proxy filtering, firewalls?
- The Internet Provider?
  - e.g., installing firewalls?

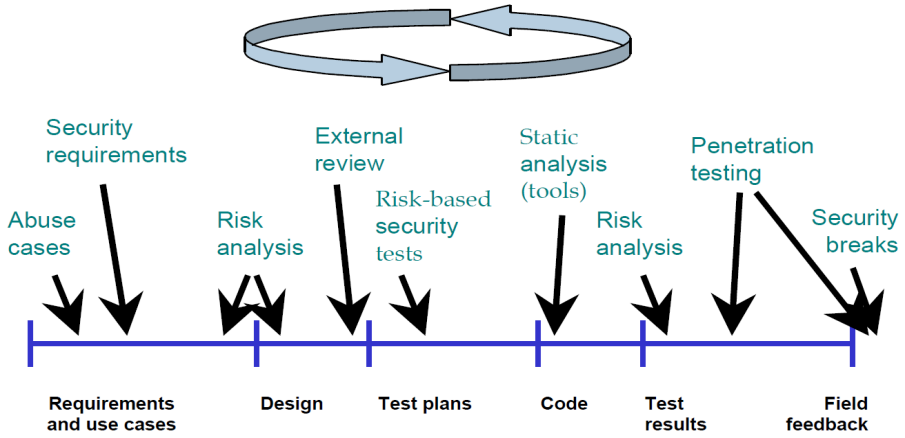
Conventional Security Solutions such as anti-virus software or firewall cannot prevent attacks caused by software vulnerabilities

Traditional Penetrate-and-Patch Software Development Approach

- Once a software vulnerability is discovered (normally by attackers), the software can be patched, but:
  - Unpatched systems remain vulnerable
  - Other vulnerabilities might still remain
  - New vulnerabilities might be discovered
- Zero-day vulnerability attacks exploit this Penetrate-and-Patch approach

Security at the source

- The developers should be responsible for security at the design and development phase
  - Secure Development Lifecycle has been proposed



Source: "Improving Security Across the Software Development Lifecycle – Task Force Report", April 1, 2004. http://www.cyberpartnership.org/init.html; based on Gary McGraw 2004, IEEE Security and Privacy. Fair use asserted.
Credit: David A. Wheeler

About this course

- Understanding of the impact of software vulnerabilities
- Develop “Building Security In” Approach
  - Secure and Sustainability Principles and Practices in Application Development (focused on Internet-based Applications)
  - Avoid security problems at the design stage

About this course (2)

- Principles and practices of Internet Application Development
  - Understand the core concepts of network programming
  - Design and implementation of real-world Internet Applications

Internet Application Security

- Data transmissions
  - Data transferred over the Internet can be stolen
    - Due to man-in-the-middle attacks
    - Due to software vulnerabilities
- Cyber attacks
  - Denial-of-Service
  - Buffer overflow
  - Data races
  - Code Injections
  - Session hijacking

Tentative Topics (Security)

- Fundamental security design principles
- Software vulnerabilities and built-in countermeasures
- Secure Programming Practices
  - Specification of Security Requirements
  - Principles of Secure Programming
  - Robust Programming
  - Defensive Programming
    - Input Validation, Type checking
  - Programming Flaws
    - Buffer Overflows, Integer Errors
  - Data Protection

Tentative topics (Programming)

- Socket programming over TCP/IP
- Client/Server architecture
- Concurrency, real-time applications
- Web application development
- WebSocket, Ajax

Course Organization

- Theoretical Background
- Hands-on exercises
  - A part of labs
  - Will be prepared from scratch
    - You do not need to have prior experience in the programming languages

"Tell me and I forget. Show me and I remember. Involve me and I understand."

Chinese proverb

Labs

- HTTP exercise
- Client/Server Applications and Packet Sniffing
- Concurrent and multi-threaded programming
- Webpage development (HTML5/CSS)
  - Build your live resume on Bitbucket page
- Web Application Development
  - Sub-labs for each vulnerabilities
- Search Hint with Ajax
- Simple Chat System with WebSocket
- Simple Chat System with Node.JS

Programming Assignments and the Final Project

- Programming assignments are based on related labs with in-class tutorials
  1. A Simple Secure Multi-threaded Chat System (GoLang and Node.js)
  2. A Secure Web Application (HTML/JavaScript/PHP)
- The final project
  - A Secure Real-time Facebook-like Web Application
    - Extension of Assignment 2 and related labs

Isidore submission policy

- Submission include:
  - A minimal report, e.g., screenshots
  - Code
- All submissions must be through the Isidore system
  - Otherwise, your submissions can not be graded
- Deadlines can be flexibly extended with advanced reasonable request only

  -5% for the first 12 hour late, and -10% for each 24 hour late after that

  -2% of the assignment grade if the submission does not follow one item of the requirements, e.g., not in PDF

Assessments (Tentative)

- 10% participation (via multiple-choice quizzes)
- 20% of labs
- 20% of programming assignments
- 15% of final project.
- 15% of mid-term test
- 20% of the final exam

Bonus

  - Up to 3% of extra credit labs

Why Study this course?

- You will be trained to become a **competent developer** with mindset of a hacker
  - Note that a hacker is not an attacker
    - Hacker: "A person who delights in having an intimate understanding of the internal workings of a system, computers and computer networks in particular" [RFC 1392]
- You will be well-prepared for **job market** with hands-on experiences and skills on the latest and most popular technologies
  - C/Java Networking, git, PHP/MySQL, JavaScript/HTML5/CSS3, GoLang, Node.js. Ajax, and WebSocket

Server-side programming languages Market Share

Source: w3techs.com (December 2019)

© W3Techs.com	usage	change since 1 December 2019
1. PHP	78.9%	
2. ASP.NET	10.6%	-0.1%
3. Java	3.6%	-0.2%
4. Ruby	3.1%	+0.2%
5. static files	1.8%	-0.1%

percentages of sites

Popular sites using PHP

- Facebook.com
- Baidu.com
- Wikipedia.org
- Qq.com
- Taobao.com
- Vk.com
- Sohu.com
- Twitter.com
- 360.cn
- Sina.com.cn

JavaScript is used by 95% of all the websites

Most popular client-side programming languages

© W3Techs.com	usage	change since 1 December 2019
1. JavaScript	95.0%	
2. Flash	2.8%	-0.1%

percentages of sites

Node.JS in Real World

- Based on JavaScript
  - For both client and server side
- Deployed by many big companies:
  - Groupon
  - IBM
  - LinkedIn
  - Microsoft
  - Netflix
  - PayPal
  - Walmart
  - Yahoo!
  - …

https://en.wikipedia.org/wiki/Node.js

GoLang in Real World

- Go is a compiled programming language designed at Google, announced in 2009, and the first version released in 2012. Notable users (except Google):
  - Docker, a set of tools for deploying Linux containers
  - Couchbase, Query and Indexing services
  - Dropbox, migrated critical components from Python to Go
  - Ethereum, a cryptocurrency
  - MongoDB
  - Netflix
  - Uber, geofence-based queries
  - . . .
- Comprehensive list: https://github.com/golang/go/wiki/GoUsers

Instructor: Phu H. Phung, Ph.D.

- Assistant Professor of Computer Science
  - You can call me Phu, or Dr. Phung
- Director of Intelligent Systems Security Lab: http://isseclab-udayton.github.io/
  - Research areas: Software security, web security, cloud computing, mobile computing security
- Homepage: http://academic.udayton.edu/PhuPhung/
- Email: phu@udayton.edu
- Office: Anderson 146
- Office hours: TR 2:15-3:15 PM, F 3:30-4:30 PM or by appointment

TA: Praveen Hiremath

- MCS Student
- Doing master thesis in the ISSec-Lab
- Former student in SeCAD-S19

- Office hours: TR 11:00 AM - 2:00 PM
- Office: ISSec-Lab, Anderson 135

Recommended textbooks

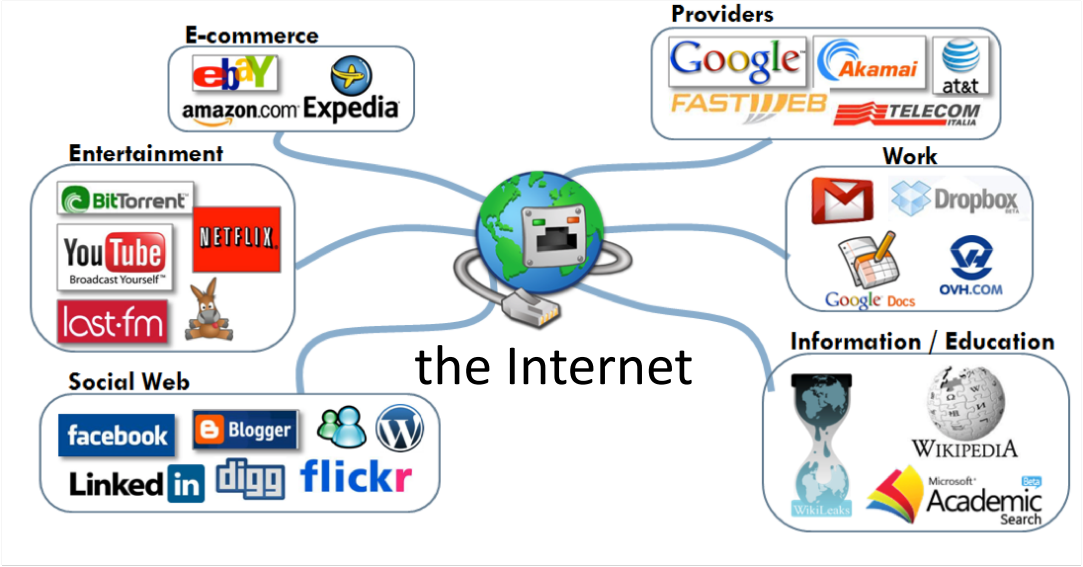
- "*Secure Programming HOWTO - Creating Secure Software*", David A. Wheeler, ebook: https://www.dwheeler.com/secure-programs/Secure-Programs-HOWTO.html
- *"Computer Security: Principles and Practice"*, Third Edition, William and Stallings Lawrie Brown.
- "*Building Secure Software – How to Avoid Security Problems the Right Away*", John Viega, Gary R. McGraw.
- "*Beej's Guide to Network Programming Using Internet Sockets*", ebook: http://beej.us/guide/bgnet/
- **Additional readings will be posted correspondingly before lectures.**

Questions?

- Note: This course will be counted as one of core courses for CPS Graduate program **(core substitute)**
- Schedule might be updated with 24-hour notice

Review: Internet Applications

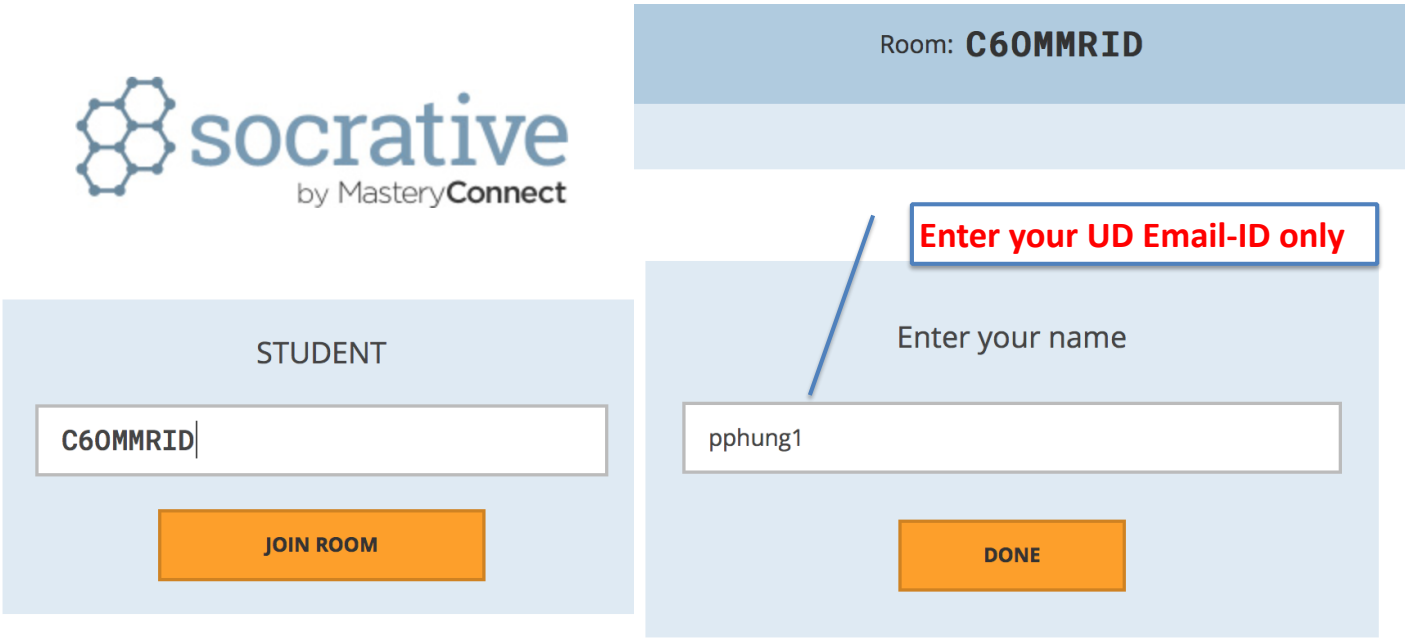
- Applications that communicate to each other over the Internet



Picture credit: ict-mplane.eu

http://www.socrative.com

-> Student Login



[redacted]

A. A protocol to connect computers

B. A standard for a computer to connect to a network

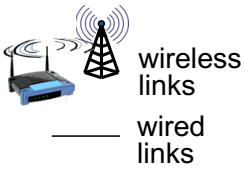
C. World Wide Web

D. Computers that are all linked together in a network

E. All of the above

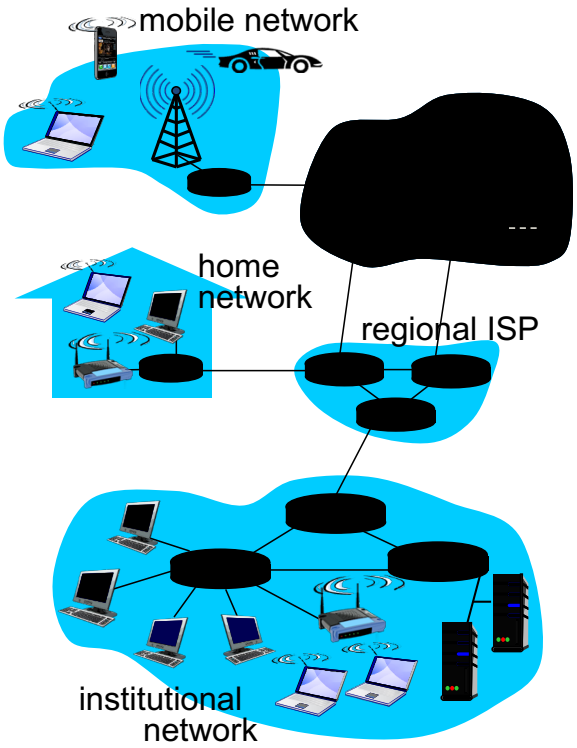
What's the Internet: "nuts and bolts" view

- millions of connected computing devices:
  - *hosts* = *end systems*
  - running *network apps*



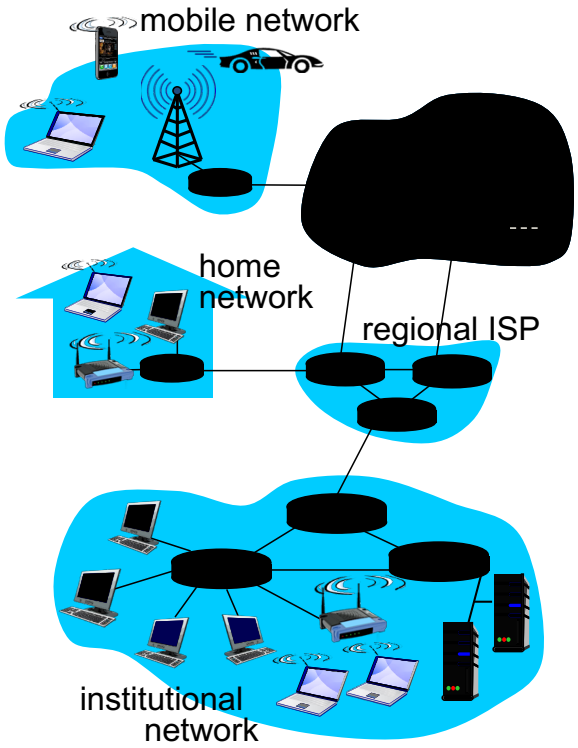
- *communication links*
  - fiber, copper, radio, satellite
  - transmission rate: *bandwidth*

- *Packet switches:* forward packets (chunks of data)
  - *routers* and *switches*



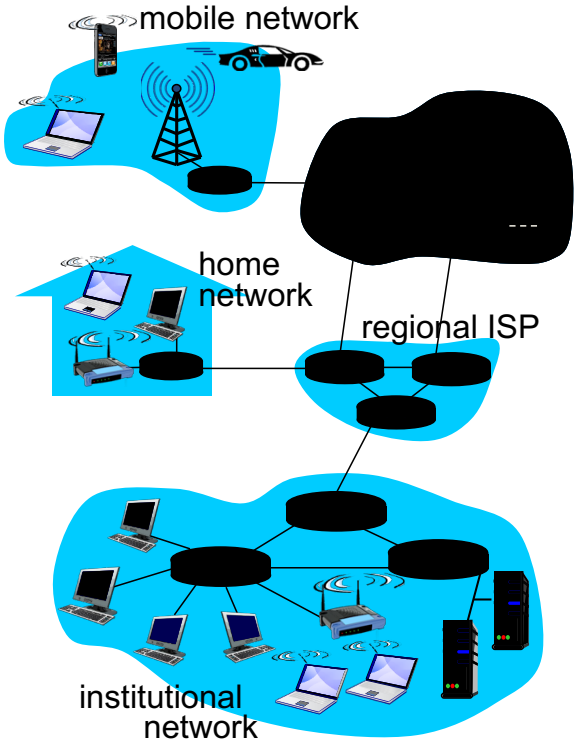
What's the Internet: "nuts and bolts" view

- *Internet:* "network of networks"
  - Interconnected ISPs
- *protocols* control sending, receiving of msgs
  - e.g., TCP, IP, HTTP, Skype, 802.11
- *Internet standards*
  - RFC: Request for comments
  - IETF: Internet Engineering Task Force



What's the Internet: a service view

- *Infrastructure that provides services to applications:*
  - Web, VoIP, email, games, e-commerce, social nets, …
- *provides programming interface to apps*
  - hooks that allow sending and receiving app programs to "connect" to Internet
  - provides service options, analogous to postal service



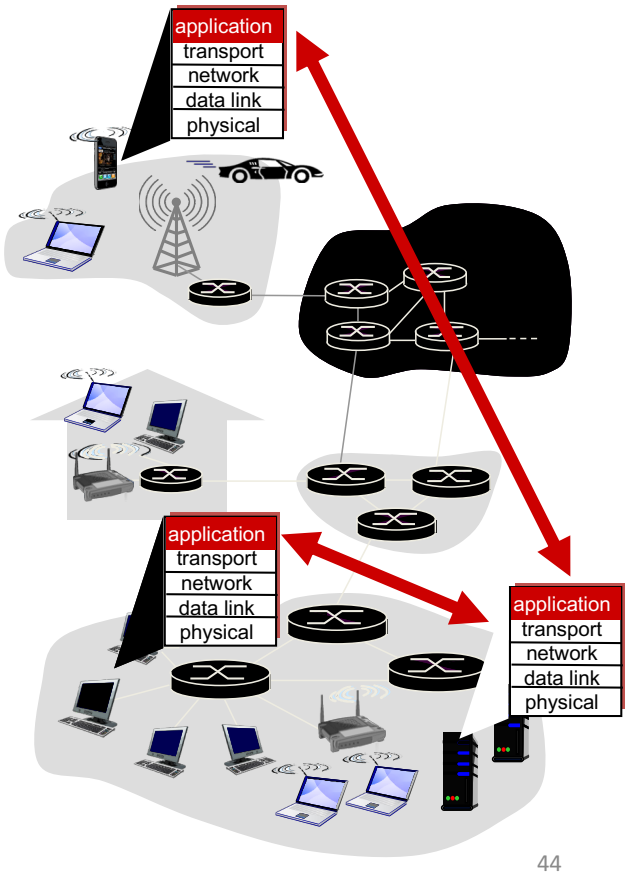
Creating an Internet application

write programs that:

- run on (different) *end systems*
- communicate over network
- e.g., web server software communicates with browser software

no need to write software for network-core devices

- network-core devices do not run user applications
- applications on end systems allows for rapid app development, propagation

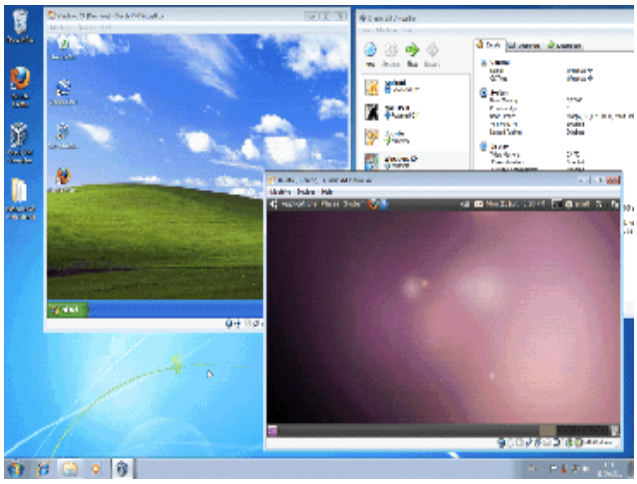


Homework – Preparation for Lab 0

- Download and install VirtualBox 6.1 in your laptop
  - https://www.virtualbox.org/
- Download the SEEDUbuntu16.04.zip file from http://www.cis.syr.edu/~wedu/seed/lab_env.html
  - Direct link: http://bit.ly/SEEDUbuntu
    - Save and unzip in your laptop in a permanent directory
- Load the unzipped to VirtualBox. Follow the instructions here: http://bit.ly/SEEDUbuntu-setup
  - After this step, you are ready to run the virtual machine, see the instruction here for login information: http://bit.ly/SEEDUbuntu-guide
- Create a bitbucket account (https://bitbucket.org), if you do not have one

Lab/Assignment Environment

- VirtualBox
  - A free virtual machine system by Oracle https://www.virtualbox.org/
  - Cross-platform



Next class

- Lab 0 - Programming environment tutorial
  - SEED Virtual machine setup
    - Virtual networking setup
  - git tutorial
    - Signup and create a repository on bitbucket.org
    - Clone a repo
    - Fun with buffer overflow
    - Update a repo
- Important Note: There is no grade for this lab, however, **you need to PASS this lab to be eligible to continue in this class**.

A Notable Student Testimonial

A few of the things that's really helped me were being able to explain what a SQL injection is and how to prevent it, the difference between a primary key and foreign key, accessing a database with different user privileges, advantages of open source software, difference between SQL and NoSQL databases (advantages and disadvantages, knowledge of Linux, and explaining the MiniBook final project.

Your class benefited me extremely and I just wanted to make sure that you knew and you so that you can tell your classes what employers might ask about. I ended up getting a job offer 15 minutes after I left the interview, they said that they were truly impressed with my range of knowledge.

Thank you again,
Kevin