

18732: Secure Software Systems Course Overview

Lujo Bauer

Anupam Datta

Fall 2010

Plan for Today

- Course logistics
- Software security in the news
- Overview of course topics

Course Staff

- Instructor: Lujo Bauer
 - Office: CIC 2121
 - Email: lbauer@cmu.edu
 - Office hours: Wed 2-3PM
- Instructor: Anupam Datta
 - Office: CIC 2118
 - Email: danupam@cmu.edu
 - Office hours: Tue 1:30-2:30PM
- TA: Michelle Mazurek
 - Office: CIC 2220B
 - Email: mmazurek@andrew.cmu.edu
 - Office hours: Mon 2-4PM

Logistics

- Location: WeH 4623
- Days: Monday & Wednesday
- Time: 10:30AM-12:20PM
(usually 10:30AM-11:50PM)
- Web page: <http://www.ece.cmu.edu/~ece732/>
- Course blackboard
- 732-instructors@lists.andrew.cmu.edu

Grading

- Exams: 40% (20% + 20%)
 - Three in-class, closed book exams
 - Only two highest grades count!
- Class participation: 10%
- Course projects: 50% (20% + 15% + 15%)
 - Next slide
- Re-grading policy
 - Grade could go up or *down*

Projects and Reading

- Course Projects:
 - Teams of 2 or 3 (form team by end of week)
 - All team members receive the same grade
 - 1. Implementing buffer overflow & web attacks
 - 2. Analyzing software for security vulnerabilities using automated tools
 - 3. Building a secure application using a security typed language
- Reading:
 - No textbook for the course
 - Slides will be posted after lectures
 - Research papers on which lectures are based

Prerequisites

- Familiarity with C and Java is required
- An introductory course in computer security such as 18-487 or 18-730 is recommended, but not required
- Undergraduate courses in OS, Compilers would be a bonus
- Quick class poll

Cheating

<http://www.cmu.edu/policies/documents/Cheating.html>

Some highlights:

- Cheating includes submitting solutions that are not your own, or helping another student to do so (e.g., by sharing your solutions with them)
- Examples of cheating
 - Looking up the answers on the web or elsewhere (even with citation)
 - Formulating solutions in a group, except for a class-sanctioned group
 - Allowing others to see your solutions (including “by accident”)

Questions about course logistics?

Software Bugs in the News

Software Bugs in the News

Unmanned European **rocket explodes on first flight**

Europe's newest unmanned satellite-launching rocket, the Ariane 5, intentionally was blown up Tuesday just seconds after taking off on its maiden flight. ...

[<http://edition.cnn.com/WORLD/9606/04/rocket.explode/>]



... The internal SRI software exception was caused during execution of a data conversion from 64-bit floating point to 16-bit signed integer value. The **floating point number which was converted had a value greater than what could be represented by a 16-bit signed integer**. This resulted in an Operand Error. The data conversion instructions (in Ada code) were not protected from causing an Operand Error, although other conversions of comparable variables in the same place in the code were protected. ...

[*ARIANE 5 Flight 501 Failure, Report by the Inquiry Board, Paris, Jul 19 1996*]

Software Bugs in the News

... A previously-unknown **software flaw** in a widely-deployed General Electric energy management system contributed to the **devastating scope of the August 14th northeastern U.S. blackout** ...

[*Security Focus*, Feb 11 2004]

The Northeast Blackout of August 2003, the largest in North American history, shut down 62,000 MW of generation capacity, and **cost businesses an estimated \$13 billion** in productivity. ...

[*IEEE-USA Today's Engineer*, Feb 2005]

... “There was a **couple of processes that were in contention for a common data structure**, and through a software coding error in one of the application processes, they were both able to get write access to a data structure at the same time ... And that corruption led to the alarm event application getting into an infinite loop and spinning.” ...

[*Security Focus*, Apr 7 2004]

Software Bugs in the News

- 1 bug = \$13 billion in losses
 - = 260,000 years of CMU tuition
 - = MS degrees for 130,000 people

Software Bugs in the News

E-voting vendor: Programming errors caused dropped votes

... E-voting machines from Premier Election Solutions, formerly called Diebold Election Systems, **dropped hundreds of votes** in 11 Ohio counties during the primary election, as the machine's memory cards uploaded to vote-counting servers. ...

[*Network World*, Aug 22 2008]

Software Bugs in the News

... **Software bugs in a Soviet early-warning monitoring system nearly brought on nuclear war in 1983**, according to news reports in early 1999. The software was supposed to filter out false missile detections caused by Soviet satellites picking up sunlight reflections off cloud-tops, but failed to do so. Disaster was averted when a Soviet commander, based on a what he said was a '...funny feeling in my gut', decided the apparent missile attack was a false alarm. The filtering software code was rewritten. . . .

[<http://rajasriengg.wordpress.com/2008/07/16/recent-major-computer-system-failures-caused-by-software-bugs/>]

Software Bugs in the News

- Accidents
- Monetary loss
- Effect on political process?
- Military conflict?

- But is it a *computer security* problem?

Malware implicated in fatal Spanair plane crash

Computer monitoring system was infected with Trojan horse, authorities say

Authorities investigating the 2008 crash of Spanair flight 5022 have discovered a central computer system used to monitor technical problems in the aircraft was infected with malware. ... infected computer failed to detect three technical problems with the aircraft ... Flight 5022 crashed just after takeoff from Madrid-Barajas International Airport two years ago today, killing 154 and leaving only 18 survivors.

[MSNBC (<http://www.msnbc.msn.com/id/38790670>), Aug 20 2010]

Software Bugs in the News

- Accidents
 - Monetary loss
 - Effect on political process?
 - Military conflict?
-
- But is it a *computer security* problem?
 - Computer security = protecting computers against misuse and interference
 - Bugs can be (and are) purposefully exploited

Exploiting Bugs as a Nuisance

- MyDoom (2004) - \$38.5 billion
- SoBig (2003) - \$37.1 billion
- Love Bug (2000) - \$15 billion
- Code Red (2001) - \$2 billion

Exploiting Bugs for Profit

- Hacker convicted of breaking into a business' computer system, stealing confidential information and threatening disclosure if \$200,000 not paid

[U.S. Dept. of Justice Press Release, Jul 2003]

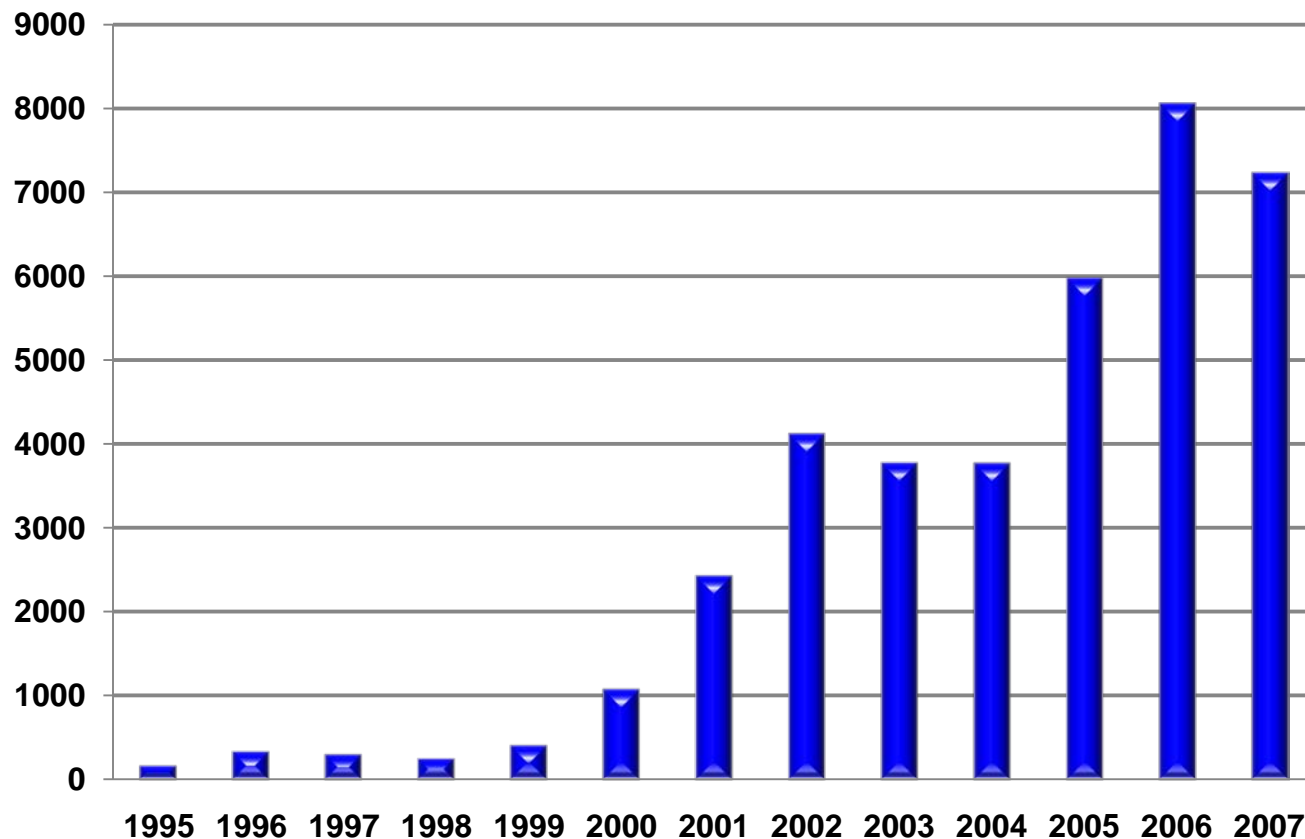
- 11 people indicted for stealing more than 40 million credit card and debit card numbers

[CNN, Aug 2008]

There Are *Lots* of Bugs!

[<http://www.cert.org/stats>]

Vulnerabilities reported to CERT/CC



[<http://www.securityfocus.com/vulnerabilities> (visited 2010-08-22)]

[SlideShowPro Director 'p.php' Directory Traversal Vulnerability](#)

2010-08-20

<http://www.securityfocus.com/bid/42566>

[MediaCoder Remote Buffer Overflow Vulnerability](#)

2010-08-20

<http://www.securityfocus.com/bid/38405>

[Adobe Flash Player and AIR \(CVE-2010-2216\) Unspecified Memory Corruption Vulnerability](#)

2010-08-20

<http://www.securityfocus.com/bid/42362>

[Adobe Flash Player and AIR \(CVE-2010-2213\) Multiple Unspecified Memory Corruption Vulnerabilities](#)

2010-08-20

<http://www.securityfocus.com/bid/42364>

[Adobe Flash Player and AIR \(CVE-2010-2215\) Unspecified Clickjacking Vulnerability](#)

2010-08-20

<http://www.securityfocus.com/bid/42361>

[Adobe Flash Player and AIR ActionScript AVM1 ActionPush Memory Corruption Vulnerability](#)

2010-08-20

<http://www.securityfocus.com/bid/42363>

[<http://www.securityfocus.com/vulnerabilities> (visited 2010-08-22)]

[Adobe Flash Player and AIR \(CVE-2010-2214\) Unspecified Memory Corruption Vulnerability](#)

2010-08-20

<http://www.securityfocus.com/bid/42358>

[Oracle MySQL 'ALTER DATABASE' Remote Denial Of Service Vulnerability](#)

2010-08-20

<http://www.securityfocus.com/bid/41198>

[Sourcefabric Campsite Multiple Cross Site Scripting Vulnerabilities](#)

2010-08-20

<http://www.securityfocus.com/bid/42107>

[Freeciv Lua Runtime Environment Remote Command Execution Vulnerability](#)

2010-08-20

<http://www.securityfocus.com/bid/40598>

[GnuPG 'GPGSM Tool' Certificate Importing Remote Code Execution Vulnerability](#)

2010-08-20

<http://www.securityfocus.com/bid/41945>

[Mozilla Firefox Plugin Parameter Reference Remote Code Execution Vulnerability](#)

2010-08-20

<http://www.securityfocus.com/bid/41933>

Page 2

[<http://www.securityfocus.com/vulnerabilities> (visited 2010-08-22)]

[Cacti Multiple Cross Site Scripting Vulnerabilities](#)

2010-08-20

<http://www.securityfocus.com/bid/40332>

[Cacti 'rra_id' Parameter SQL Injection Vulnerability](#)

2010-08-20

<http://www.securityfocus.com/bid/40149>

[Cacti 'export_item_id' Parameter SQL Injection Vulnerability](#)

2010-08-20

<http://www.securityfocus.com/bid/39653>

[Cacti Multiple Cross Site Scripting and HTML Injection Vulnerabilities](#)

2010-08-20

<http://www.securityfocus.com/bid/37109>

[Cacti Multiple Input Validation Security Vulnerabilities](#)

2010-08-20

<http://www.securityfocus.com/bid/39639>

[Microsoft Word Record Parsing Length Field Remote Stack Buffer Overflow Vulnerability](#)

2010-08-20

<http://www.securityfocus.com/bid/35188>

Page 3

[<http://www.securityfocus.com/vulnerabilities> (visited 2010-08-22)]

[Red Hat VDSM Module SSL Connection Denial of Service Vulnerability](#)

2010-08-20

<http://www.securityfocus.com/bid/42580>

[QEMU KVM 'exec.c:subpage_register\(\)' Memory Corruption Vulnerability](#)

2010-08-20

<http://www.securityfocus.com/bid/42579>

[QEMU QXL Graphics Local Memory Corruption Vulnerability](#)

2010-08-20

<http://www.securityfocus.com/bid/42578>

[Novell iPrint Client Multiple Security Vulnerabilities](#)

2010-08-20

<http://www.securityfocus.com/bid/42576>

Page 4!

MediaCoder Remote Buffer Overflow Vulnerability

SlideShowPro Director 'p.php' Directory Traversal Vulnerability

GnuPG 'GPGSM Tool' Certificate Importing Remote Code Execution Vulnerability

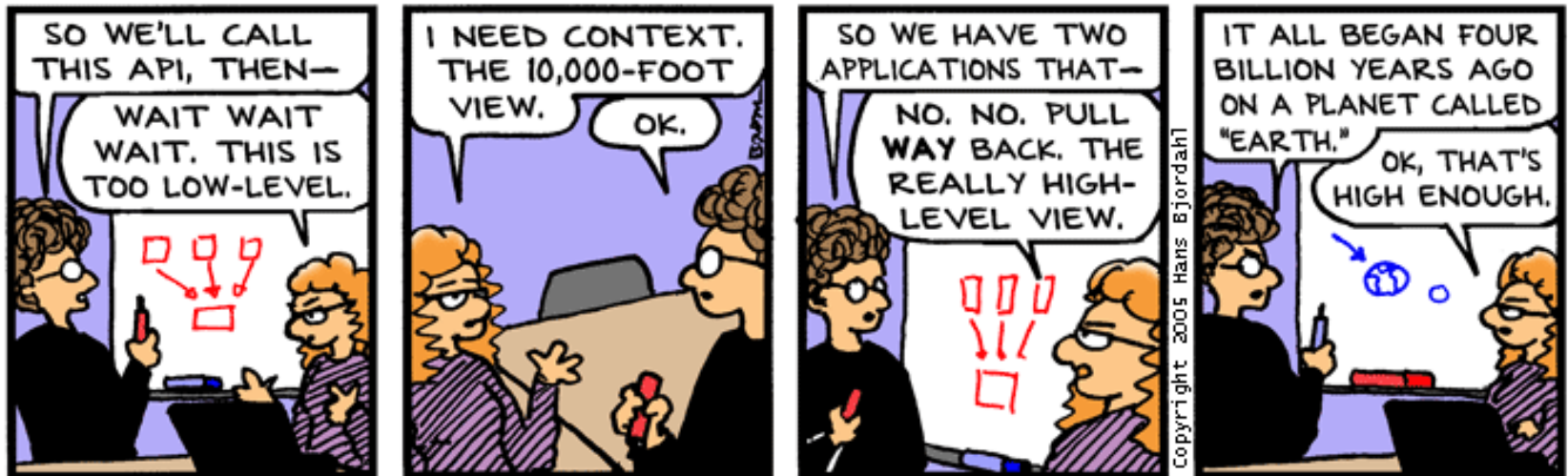
phpMyAdmin Multiple Cross Site Scripting Vulnerabilities

libHX 'HX_split()' Remote Heap-Based Buffer Overflow Vulnerability

libHX is prone to a remote heap-based buffer-overflow vulnerability because it **fails to properly validate user-supplied input**.

[<http://www.securityfocus.com/bid/42592/discuss>]

10,000-foot View



Bug Bash by Hans Bjordahl

<http://www.bugbash.net/>

Course Goals

1. Understand current software attacks and defenses
2. Understand general principles of secure software system design and analysis

Topics

0. Attacks

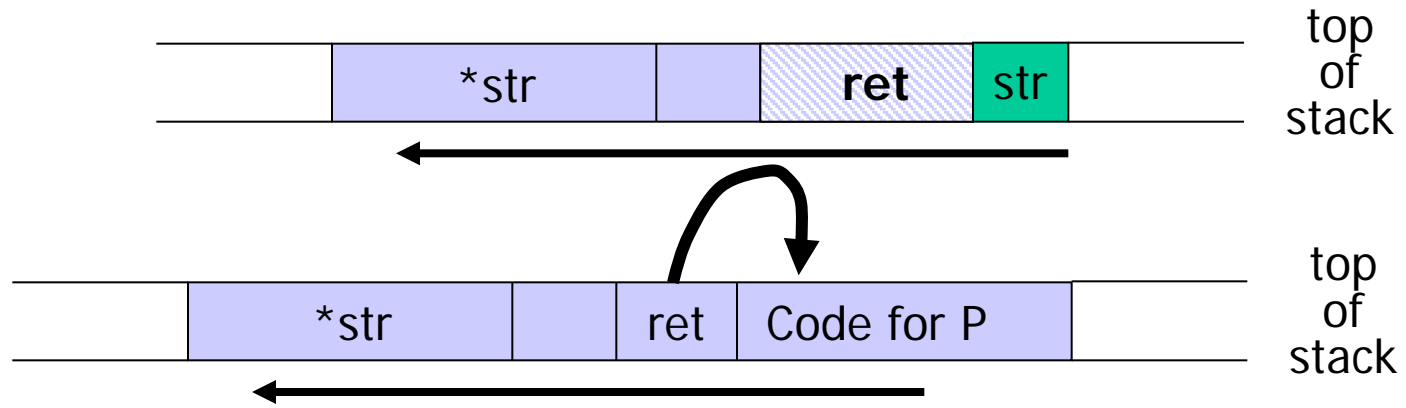
1. Software Security Architectures
2. Security Analysis of Software
3. Language-based Security
4. Run-time Security Enforcement

Control Hijacking Attacks

- Attacker causes arbitrary attack code (e.g. root shell) to execute on target machine by hijacking the control flow of an application program
- Examples
 - Buffer overflow attacks
 - Integer overflow attacks
 - Format string vulnerabilities

Buffer Overflow Attacks

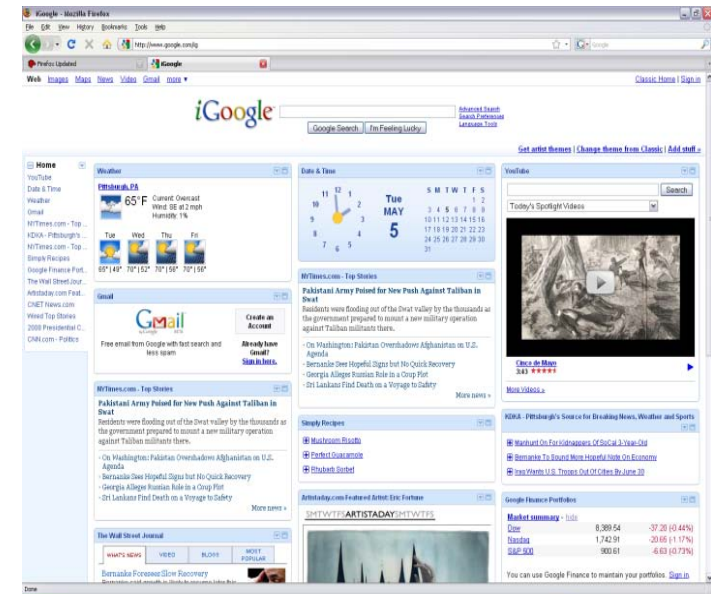
- One cause: Array abstraction not protected in C
 - Suppose array is 128 bytes
 - Possible to write 200 bytes into the array and overwrite return address of function



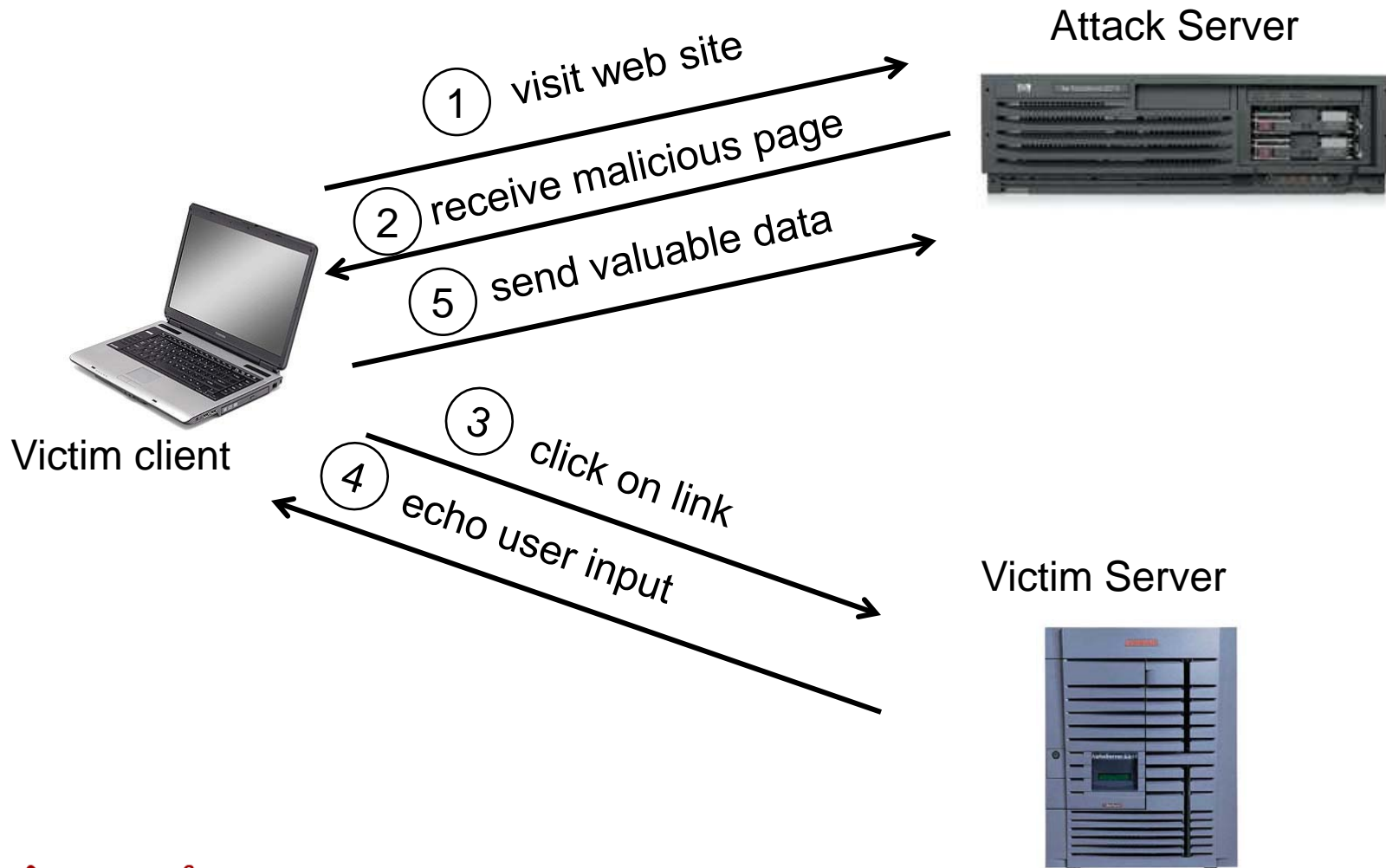
Program P: `exec("/bin/sh")`

Web Attacks

- Web-based attacks
(Top 3 web site vulnerabilities)
 1. Cross-site scripting (XSS)
 2. Cross-site request forgery
 3. SQL injection



A Reflected XSS attack



Learning Outcome

- Understanding of the state-of-the-art in control hijacking and web-based attacks and associated defenses

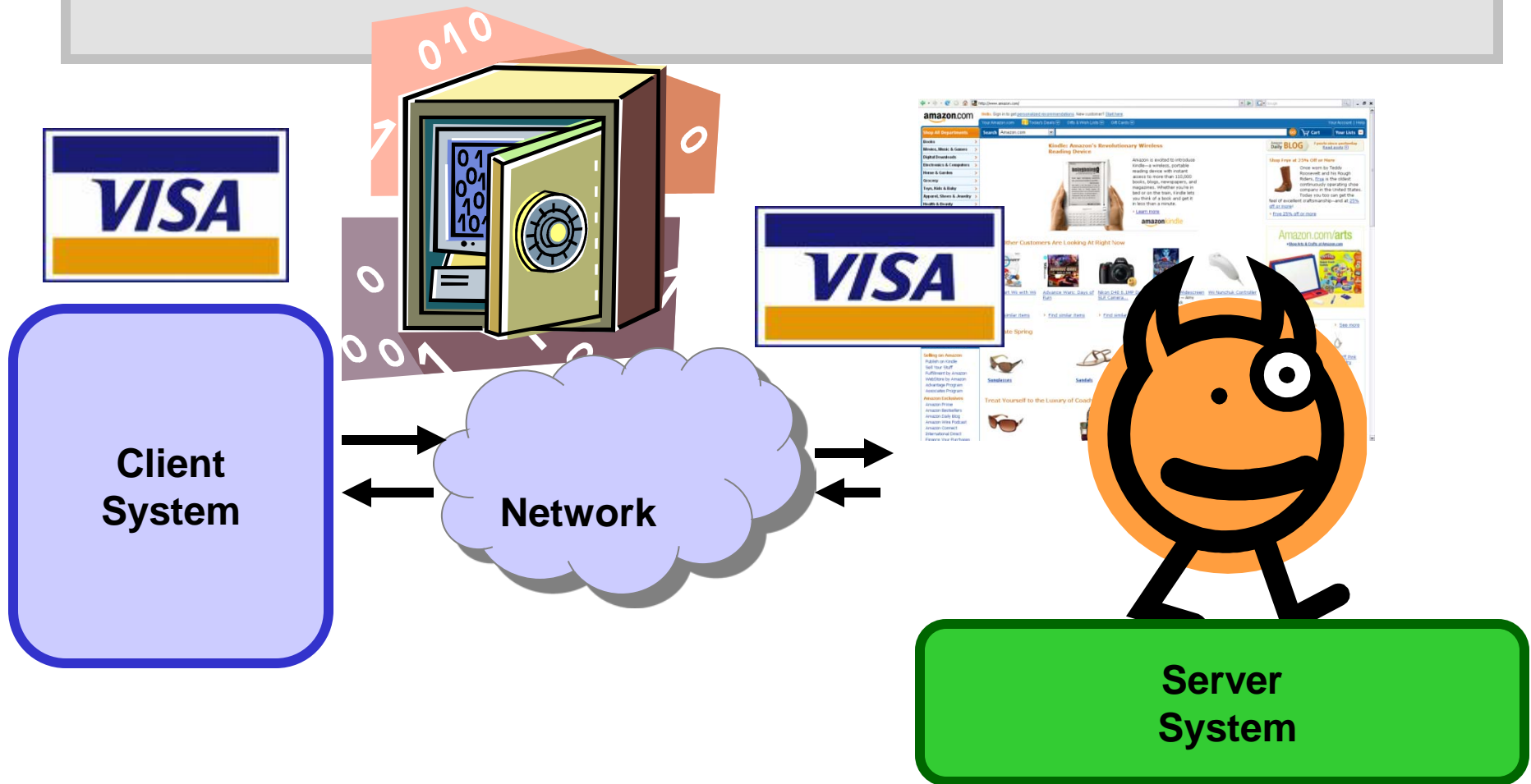
PROJECT 1

- *Note:* These two classes of attacks currently top the charts of common security vulnerabilities databases

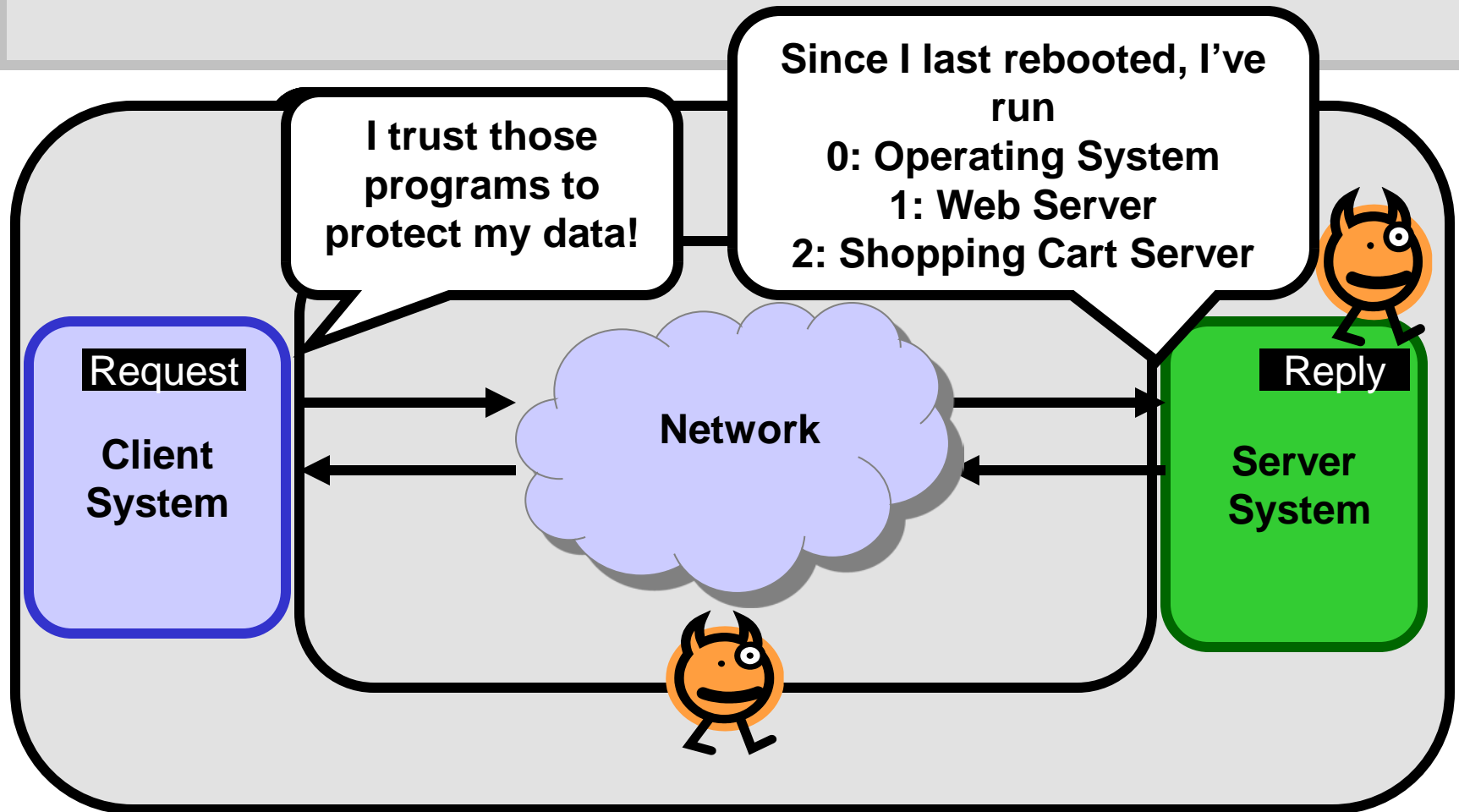
Topics

0. Attacks
1. **Software Security Architectures**
2. Security Analysis of Software
3. Language-based Security
4. Run-time Security Enforcement

Trusted Computing



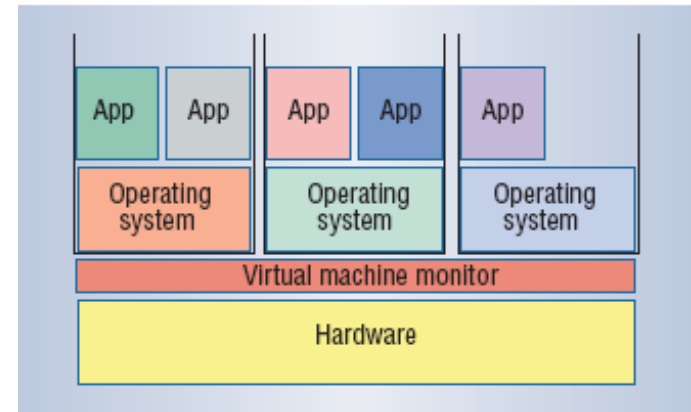
Remote Attestation



Architecture designed to guarantee integrity property of server code measurements in the presence of adversaries

Isolation (is good for security)

- Virtual machine-style architecture aims to provide *isolation* between multiple VM's running on the same physical machine



- Will cover a number of related approaches and their implications for security
 - Java VM and web security, software-based fault isolation,...

Learning Outcome

- Systematic thinking about system security
 - What is the security mechanism?
 - What is the adversary model?
 - What is the desired security property?
 - Does system guarantee security property in the presence of adversaries?
 - What is the trusted computing base (TCB)? What are our assumptions about the TCB?
- Understanding of specific security architectures for trusted computing and separation

Topics

0. Attacks
1. Software Security Architectures
2. Security Analysis of Software
3. Language-based Security
4. Run-time Security Enforcement

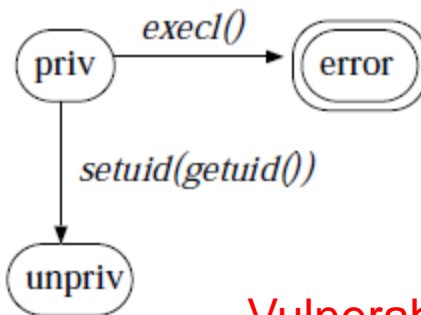
Security Analysis of Software

```
#include <stdio.h>
main(argc, argv)
int argc;          /* the number of arguments in argv */
char * argv[];    /* of null-terminated strings also known */
                  /* as an array of arrays of char. It */
                  /* holds the contents of the command line */
{
    int loopcounter;

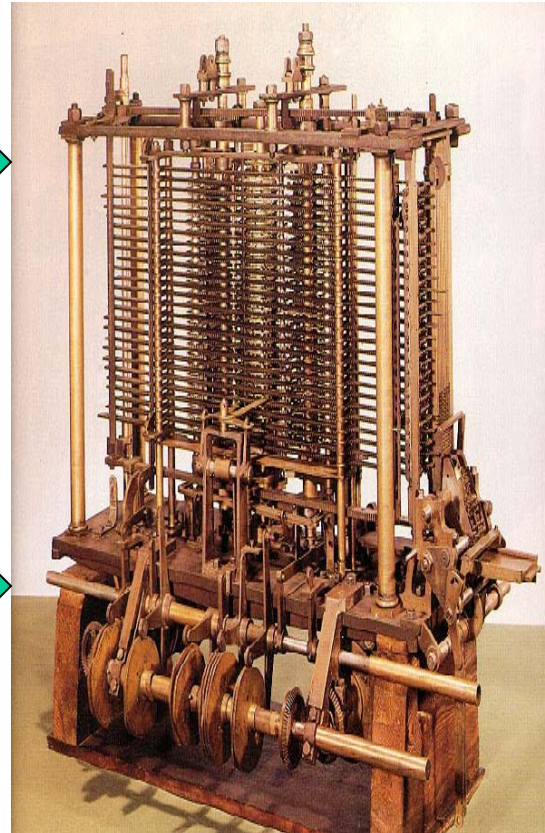
    /* NOTE: The name with which the program */
    /* was invoked is held in argv[0]. */

    for (loopcounter=1; loopcounter < argc; loopcounter++) {
        switch (argv[loopcounter][0]) {
            case '-': {
                printf("OPTION %s\n",
                    &argv[loopcounter][1]);
                break;
            }
            default : {
                printf ("FILENAME %s\n",
                    argv[loopcounter]);
                break;
            }
        }
    }
}
```

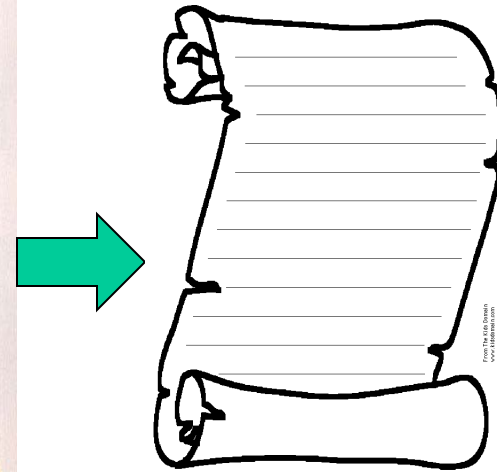
Program



Vulnerability
description



Analytical
Engine



List of potential
vulnerabilities
in program

Learning Outcome

- Understanding of *general methods* of software analysis (model checking, static analysis, dynamic analysis) and their application to identifying *security vulnerabilities* in legacy code written, for example, in C (buffer overflows, race conditions, setuid, chroot, malware...)

Note: Methods could be applied to tomorrow's attacks

- Understanding of strengths and limitations of various methods (false positives, false negatives, ...)
- Hands-on experience with a state-of-the-art industrial software analysis tool – *Coverity Prevent*

PROJECT 2

Topics

0. Attacks
1. Software Security Architectures
2. Security Analysis of Software
3. Language-based Security
4. Run-time Security Enforcement

Security Typed Languages

- Type safe programming languages
 - Think Java (as opposed to C)
 - Better protection for language abstractions such as arrays
 - No buffer overflow attacks on Java programs
- Security typed programming languages
 - Language designed so that a program that compiles correctly does not incur security violation at run-time

What Is a Type System?



“Now! *That* should clear up a few things around here!”

- Jif Example

Secrecy
label

```
int {alice→bob,alice; bob←alice} y;
int {bob→bob} x;
int {alice→bob; bob→alice} z;
if (x == 0)
  z = y;
```

Secrecy violation:
Information about x
is leaked to Alice

- Well-typed program won't have security violations
- Type checking done by compiler

Learning Outcome

- Understanding of basic concepts for type systems – syntax, static and dynamic semantics, type safety
- Understanding the security property of *non-interference* and how type soundness can imply non-interference
- Understanding of type systems to improve security of mobile (PCC) and assembly code (TAL)
- Hands-on experience with building a secure application in Jif – a security-typed programming language that extends Java with security types

PROJECT 3

Topics

0. Attacks
1. Software Security Architectures
2. Security Analysis of Software
3. Language-based Security
4. Run-time Security Enforcement

Run-time Security Enforcement

- Enforce security using mechanisms that monitor systems as they *execute* (as opposed to *static* analysis of code)
 - Captures properties that cannot be precisely enforced statically, e.g., properties that depend on specific inputs
- Some examples
 - *Stackguard*: Dynamic checks to prevent buffer overflows
 - *Taint Analysis*: Information flow control, malware detection
 - Stack inspection, firewalls, applet sandboxing, displaying security warnings, ...

Learning Outcome

- Understanding of fundamentals of run-time enforcement of security properties
 - What class of security policies can be enforced using run-time monitoring?
 - How do we specify policies for run-time monitors to enforce?
- Understanding of a general run-time mechanism for guaranteeing control flow integrity (CFI)
 - CFI guarantees absence of control hijacking attacks

Summary: Course Topics & Goals

Course Topics:

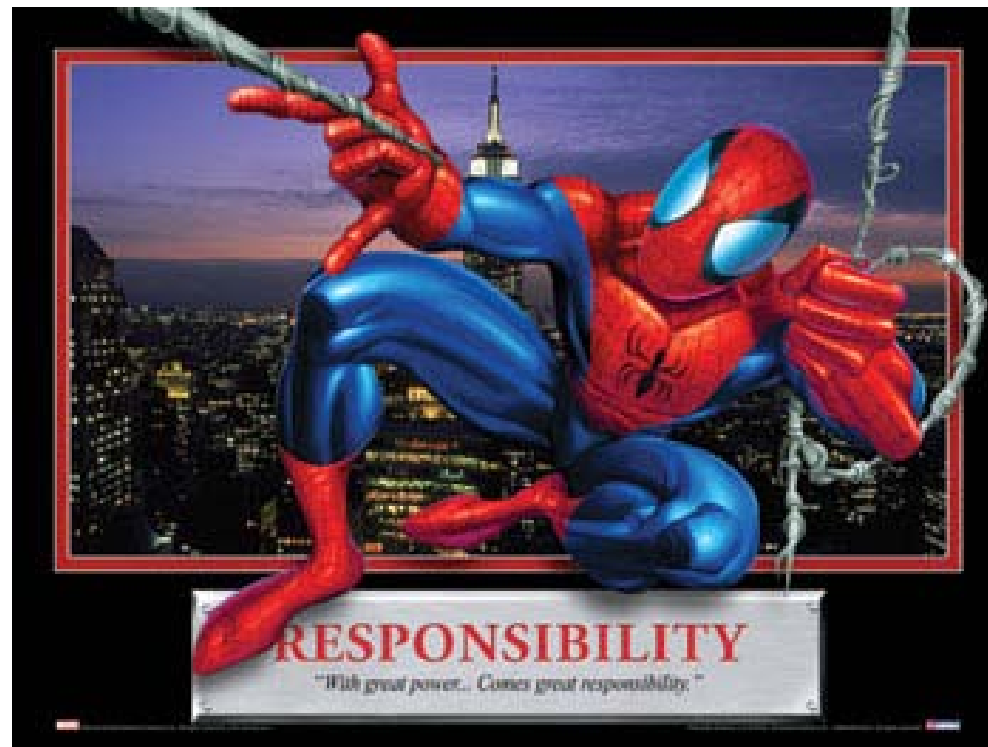
0. Attacks
1. Software Security Architectures
2. Security Analysis of Software
3. Language-based Security
4. Run-time Security Enforcement

Course Goals:

1. Understand current software attacks and defenses
2. Understand general principles of secure software design and analysis

A Final Comment

- Do not try attacks at home! Our goal is to educate so you can *defend*, not *attack*



The End

Let's get started...