

CIS/TCOM 551

Computer and Network Security

Spring 2005
Lecture 1

Course Staff

- Steve Zdancewic (Instructor)
 - e-mail: stevez@cis.upenn.edu
 - web: www.cis.upenn.edu/~stevez
- Matt Blaze (Instructor)
 - e-mail: blaze@cis.upenn.edu
 - web: www.crypto.com
- Eric Cronin (Teaching assistant)
 - e-mail: ecronin@cis.upenn.edu
 - web: www.cis.upenn.edu/~ecronin
- Gaurav Shah (Teaching assistant)
 - e-mail: gauravsh@seas.upenn.edu
 - web: www.cis.upenn.edu/~gauravsh

Course Information

- Course Web Page:
 - www.cis.upenn.edu/~cis551
- News group:
 - upenn.cis.cis551
- Textbook: none
 - Assigned reading: articles and web pages
 - Lecture slides will be available on the course web pages
- Office hours will be announced as soon as we coordinate them.

1/14/05

CIS/TCOM 551

3

Prerequisites

- Would like to learn about computer and network security.
- Some programming experience
 - C or C++ helpful (but not necessary - you can pick up what you need to know)
 - Java
- Some computer networks experience
 - Do you know what a protocol stack is?
 - Do you generally understand TCP/IP?
 - TCOM 500
 - CSE 331

1/14/05

CIS/TCOM 551

4

Course Topics

- System Security
 - Hacker behavior, intrusion & anomaly detection, hacker and admin tools
- Networks & Infrastructure
 - TCP/IP, Denial of Service, IPSEC, TLS/SSL
- Basic Cryptography
 - Shared Key Crypto (AES/DES), Public Key Crypto (RSA)
- Crypto Software & Applications
 - Open SSL library, authentication, digital signatures
- Trust and Configuration Management
- Malicious Code
 - Buffer overflows, viruses, worms, protection mechanisms
- Covert Channels

1/14/05

CIS/TCOM 551

5

Grading Criteria

- 15% Midterm 1 - tentative date: Feb. 10th
- 15% Midterm 2
- 25% Final exam
- 20% Two individual projects
- 20% Group project
- 05% Course participation

- Policies:
 - No collaboration on individual projects
 - No individual work on group projects
 - Only “reasonable” regrade requests permitted

1/14/05

CIS/TCOM 551

6

Student Background...

1. How many of you have programmed in C or C++?
2. How many of you have programmed in Java?
3. How many of you have written shell scripts?
4. How many of you have never done any programming?
5. How many of you can explain how a buffer overflow exploit works?
6. Have any of you written a buffer overflow exploit?
7. How many of you can explain how TCP/IP works?
8. How many of you have set up a wireless network?
9. How many of you have had experienced a virus or worm attack on some computer you care about?
10. Have any of you written a virus or worm?

1/14/05

CIS/TCOM 551

7

Student Background...

11. How many of you regularly use SSH or SFTP?
12. How many of you can explain how they work?
13. How many of you have run a packet sniffer or port scanner?
14. How many of you can define the term "Trusted Computing Base"?
15. How many of you have used a debugger?
16. How many of you are masters students?
17. How many of you are PhD students?
18. How many of you are undergraduates?

1/14/05

CIS/TCOM 551

8

Zdancewic's Background

- Undergraduate degree at CMU
- Ph.D. from Cornell University in 2002
- This is my 3rd year at Penn
- Research area is programming languages and security:
 - Most security problems are caused by bad software
 - Programming languages are the most important tool in software development
 - How can we improve programming languages so that it is easier to build secure and reliable software?
 - Technical details center around: static program analysis (e.g. type systems), compiler technology, language theory (CIS 500)
 - Recent work: *information-flow properties* (protecting confidential data in software)

1/14/05

CIS/TCOM 551

9

Outline

- Try to answer the question:
 - What is security?
 - What do we mean by a secure program?
- Historical context
 - Basic definitions & background
 - Examples of security
- Focus on one widespread example:
 - Buffer overflows

1/14/05

CIS/TCOM 551

10

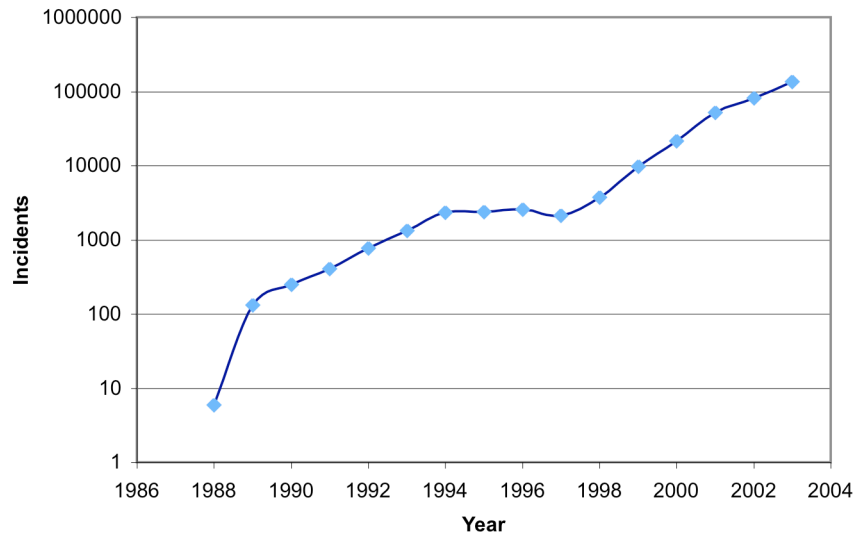
Software Vulnerabilities

- Every day you read about new software vulnerabilities in the news
 - Buffer overflows
 - Cross-site scripting
 - Format-string vulnerabilities
- Check out www.cert.org for plenty of examples

Recent Security Vulnerabilities

- Microsoft Internet Explorer 6 / Windows XP SP2.
- Announced 1/7/2005 as *Extremely Critical*
- “Some vulnerabilities have been discovered in Internet Explorer, which can be exploited by malicious people to compromise a user's system, conduct cross-site/zone scripting and bypass a security feature in Microsoft Windows XP SP2.”
- Example of a Cross-Site Scripting vulnerability
 - Embed bogus URLs that contain attack scripts embedded in them.
 - IE does not perform appropriate validation of URL inputs.

CERT Incidents



1/14/05

CIS/TCOM 551

13

What do we mean by security?

- What does it mean for a program to be secure?
- Comments generated from class discussion:
 - Someone may hack in to the system and retrieve or modify data they shouldn't.
 - An attacker may gain inappropriate access to a system.
 - A program should behave as expected.
 - It should store/transmit sensitive data in a secure manner.

 - A user should be authenticated ... so you know who's doing what.

 - A program should only be run by those who are allowed to run it.

 - A secure program should allow some kind of backtracking of recently used actions.

1/14/05

CIS/TCOM 551

14

When is a program secure?

- When it does exactly what it should?
 - Not more.
 - Not less.
- But how do we know what a program is supposed to do?
 - Somebody tells us? (But do we trust them?)
 - We write the code ourselves? (But what fraction of the software you use have you written?)

When is a program secure?

- 2nd try: A program is secure when it doesn't do something it shouldn't.
- Easier to specify a list of "bad" things:
 - Delete or corrupt important files
 - Crash my system
 - Send my password over the Internet
 - Send threatening e-mail to the present posing as me
- But... what if most of the time the program doesn't do bad things, but occasionally it does? Is it secure?

When is a program secure?

- Claim: Perfect security does not exist.
 - Security vulnerabilities are the result of violating an assumption about the software (or, more generally the entire system).
 - Corollary: As long as you make assumptions, you're vulnerable.
 - And: You *always* need to make assumptions!

- Example: Buffer overflows
 - Assumption (by programmer) is that the data will fit in the buffer.
 - This leads to a vulnerability: Supply data that is too big for the buffer (thereby violating the assumptions)
 - Vulnerabilities can be *exploited* by an *attack*.

1/14/05

CIS/TCOM 551

17

When is a program secure enough?

- Security is all about tradeoffs
 - Performance
 - Cost
 - Usability
 - Functionality

- The right question is: how do you know when something is secure enough?
 - Still a hard question
 - Requires understanding of the tradeoffs involved

- Is Internet Explorer 6 secure enough?
 - Depends on context

1/14/05

CIS/TCOM 551

18

How to think about tradeoffs?

- What is it that you are trying to protect?
 - Music collection vs. nuclear missile design data
- How valuable is it?
- In what way is it valuable?
 - Information may be important only to one person (e.g. private e-mail or passwords)
 - Information may be important because it is accurate and reliable (e.g. bank's accounting information)
 - A computer system may be important because of a service it provides (e.g. Google's web servers)

1/14/05

CIS/TCOM 551

19

Historical Context

- Assigned Reading:
Saltzer & Schroeder 1975
The Protection of Information in Computer Systems
 - available from course web pages
- Unauthorized information release
 - *Confidentiality*
- Unauthorized information modification
 - *Integrity*
- Unauthorized denial of use
 - *Availability*
- What does “unauthorized” mean?

1/14/05

CIS/TCOM 551

20

Example Security Techniques

- Labeling files with a list of authorized users
 - Access control (must check that the user is permitted on access)
- Verifying the identity of a prospective user by demanding a password
 - Authentication
- Shielding the computer to prevent interception and subsequent interpretation of electromagnetic radiation
 - Covert channels
- Enciphering information sent over telephone lines
 - Cryptography
- Locking the room containing the computer
 - Physical aspects of security
- Controlling who is allowed to make changes to a computer system (both its hardware and software)
 - Social aspects of security

Case Study: Buffer Overflows

- First project: Due: 25 Jan. 2005
- <http://www.cis.upenn.edu/~cis551/project1.html>
- Assigned Reading:
Aleph One (1996)
Smashing the Stack for Fun and Profit

Buffer Overflows

- The #1 source of vulnerabilities in software
- Caused because C and C++ are not safe languages
 - They use a “null” terminated string representation:

```
“HELLO!\n”
```

- Standard library routines *assume* that strings will have the null character at the end.
 - Bad defaults: the library routines don’t check inputs
- Easy to accidentally get wrong
- ...even easier to maliciously attack

1/14/05

CIS/TCOM 551

23

Buffer overflows (cont’d)

- Basic problem is that the library routines look like this:

```
void strcpy(char *src, char *dst) {  
    int i = 0;  
    while (src[i] != “\0”) {  
        dst[i] = src[i];  
        i = i + 1;  
    }  
}
```

- If the memory allocated to `dst` is smaller than the memory needed to store the contents of `src`, a buffer overflow occurs.

1/14/05

CIS/TCOM 551

24