**Scope I**

Due to the advent of the Web, computer security is a hot topic right now. Many new start-ups are investing heavily in E-commerce, where security is a major concern.

Security can mean different things to different people, and what you'll learn in this class is certainly very different from what you'd learn in other security courses.

We'll look at *language-based* approaches to security, i.e. any security technique that uses ideas from compiler and programming language design.

**Scope II**

We're interested in three types of security issues:

- **Protection against violation of the integrity of computer systems.** How do we protect a computer system from adverse effects of downloading (possibly hostile) code (eg. applets)?
- **Protection of intellectual property rights.** How do we protect a program from being illegally re-distributed?
- **Protection against violation of privacy rights.** Are there language-based alternatives to encryption?
Steganographic software In some countries (France, PRC?) it’s illegal for private citizens to send encrypted messages. Steganography is a way of hiding an plain-text (unencrypted) message within an object (often a digital image, video, or audio clip) so that none (except the recipient) can detect it. We’re interested in adding steganographic messages to software objects.

Tamper-Resistant Software In some cases it’s important to make sure that no-one has altered a program before we run it, or to detect that the code has been tampered with.

Software watermarking A watermark is a “copyright notice” added to an object (often a digital image, video, or audio clip) by the owner of that object. If the object is illegally copied, the owner can prove that he’s the original owner. We’re interested in applying watermarks to software objects.

Software fingerprinting A fingerprint is “customer identification number” added to an object (often a digital image, video, or audio) by the owner of that object. If the object is illegally copied, the owner can prove who made the illegal copies. We’re interested in applying fingerprints to software objects.

Software obfuscation A software developer can buy a competitor’s product, decompile it, and reuse the code in his own product. This can be a very cost-effective – albeit illegal – software development strategy. To protect themselves against code theft, a developer can obfuscate his code prior to selling it, i.e. making it so ugly and unreadable that it’s of no use to anyone.

Executing untrusted code There are many cases when we want to execute “untrusted code”, code that we did not write ourselves, and which may contain bugs, viruses, etc. Verification, Software Fault Isolation, Proof-Carrying Code, are recent methods for protecting against such malicious code.
To minimize the amount of work we have to do for our projects, we should try to use existing software as much as possible. Here are some tools which may prove to be useful:

**Software I**

- **EEL** A binary rewriting tool for Sparc/Solaris.
  [http://www.cs.rice.edu/~jan,w repell.html](http://www.cs.rice.edu/~jan,w repell.html)

- **Soot** A Java rewriting tool:
  [http://www.cs.purdue.edu/homes/soot/javaClass](http://www.cs.purdue.edu/homes/soot/javaClass)

- **BooT** A Java rewriting tool:

- **SourceAgain** A Java decompiler:

---

**Papers I**

- Luis Sarmenta. Protecting Programs from Hostile Environments: Encrypted Compilation, Obfuscation, and Other Techniques.
  [http://www.cs.arizona.edu/~collberg/Publications/](http://www.cs.arizona.edu/~collberg/Publications/)

- Collberg, Thompso:n. Low Manufacturing Cost, Resilient, and Stealthy Opaque Constructs (POP/L98).
  [http://www.cs.arizona.edu/~collberg/Publications/](http://www.cs.arizona.edu/~collberg/Publications/)

  [http://www.cs.arizona.edu/~collberg/Publications/](http://www.cs.arizona.edu/~collberg/Publications/)


### 620 - The Project

You will work in a team of 2 (or 3) on a project that you choose yourself. A list of projects will be provided, but I'd much rather you came up with your own project ideas. Rules:

1. Cool, wacky, creative, novel projects are good. Boring projects are bad.
2. A really cool, weird project that never quite works will get a better grade than a perfectly implemented boring idea.
3. I decide what's boring and cool.
4. The project should be (at least marginally related to programming languages, compilers, and security. Check with me before you start.
5. You must provide documentation in the form of a written report.
More Project Rules I

6. You will make a presentation of the finished project to the class.

7. This is a security class. We may occasionally discuss ways that criminals attack systems and software. You are not allowed to use this information to do anything illegal. If you do, you’re on your own. You could fail the class, be kicked out from the university, face criminal prosecution, or worse.

8. If you decide to implement a system that makes use of algorithms or techniques that are protected by patents, you must not sell your system or release it in the public domain, unless you have the proper permissions from the patent holder.

Slide 1–16

More Project Rules II

9. In contrast to other classes you may have taken in the past, you may freely exchange ideas and code with other students in the class. In fact, this is encouraged. If you write a particularly useful set of library functions that you think may be useful to others, then please share them with the class. If you use code provided by others (including code culled from the net), you must acknowledge this in the documentation.

Slide 1–17

620 – The Presentations

Each student will make 1–2 presentations of research papers and/or software patents to the class:

1. Select a topic.

2. Coordinate with me to make sure no one else has chosen the same topic.

3. Choose one or more papers within the topic. A set of papers will be provided for you to choose from, but you are encouraged to search out your own papers.

4. Read (and understand...) the papers.

5. Write a (∼3 – 4) page summary. Use the \LaTeX{} typsetting system.

6. Coordinate with me to select a suitable time for the presentation.

Slide 1–18

The Presentations II

7. Prepare the presentation. Make copies of the summary for the audience.

8. Give the presentation (25 minutes + 5 minutes discussion).

Slide 1–19
Reading a Research Paper I

1. Read the abstract, the introduction, and the conclusion.

2. Do you know what’s going on? If not, you need to do some background reading:
   (a) Check the paper’s reference (bibliography) section.
   (b) Look for tech-reports by the same author; they often contain details left out in a published article.
   (c) Journal articles are more detailed than conference articles.
   (d) Textbooks are sometimes helpful, but not for “new” topics.

Reading a Research Paper II

3. Read the rest of the article, but skip proofs and other technical detail.

4. Read it again, but this time make up and work some examples to make you understand the details better.

5. Read some other articles in the same area but by different authors to give you a different perspective.

Preparing a Presentation I

1. Think of good lectures you’ve been to: how did the lecturer organize the presentation?

2. Decide what audience you are talking to:
   • Are they familiar with the topic? If not, spend a lot of time on background information!
   • Are they mathematically minded? If not, avoid $\lambda \sum_{i=1}^{n} (\Gamma(n))$!
   • Are they awake? If not, make sure everything’s on the handout!

3. Avoid too much technical detail!!!

4. Avoid too much technical detail!!!

Preparing a Presentation II

4. Decide what the audience should remember after the talk. E.g.:
   • What problem is the authors trying to solve?
   • What previous attempts have there been at solving this problem?
   • What’s the authors’ principal idea?
   • Is this a solved problem now, or does the paper leave some open problems?
   • Is it, in your opinion, a good or a bad paper? Why?
1. Make sure you have adequate notes:
   - If you’re likely to be very nervous, write a script.
   - Otherwise, topic headings will suffice.
2. Talk slowly and clearly.
3. Rehearse the presentation:
   - Give the presentation to yourself.
     Read it out loud! It takes time to get used to hearing your own voice.
   - Time yourself. Do you have enough material; too much material?
   - Give the presentation to a friend.
     Not your boy/girlfriend; you’ll want someone who dares to criticize you...