University of Arizona

CSc 620

Security Through Obscurity

Christian Collberg

January 23, 2002

Administrivia

Slide 0–1

Class	:	620 – Security Through Obscurity
Lecturer	:	Christian Collberg
Email	:	collberg@cs.arizona.edu
WWW	:	http://www.cs.arizona.edu/∼collberg
Office	:	734
Office Hours	:	Open door policy.
Phone	:	621-6612
Lectures	:	Tue & Thu 3:30-4:45, GLD-S 701

Slide 0–2

Scope

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. In this class, we're interested in one particular kind of security, namely 'security through obscurity'. This means that we will be studying

- code obfuscation,
- software watermarking, and
- software tamper-proofing.

Code obfuscation is a technique for protecting a program from being reverse engineered by turning it into a much less readable program.

Slide 0–3

Scope...

Software watermarking is used to protect a program from software piracy. The idea is to embed the customer's name/credit-card number/etc. into the program in order to be able to trace it back to the customer should they sell it on to someone else.

Software tamper-proofing are techniques for making sure that a user cannot change a program, for example to remove a watermark.

We will also look at how to write a computer virus since viruses use code obfuscation and tamper-proofing to protect themselves against virus scanners.

Topics

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.

Slide 0–4

Topics

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.

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.

Computer Viruses How do you write a computer virus? How does the virus avoid detection?

Slide 0–5

Assessment

1. There will be 2–3 small initial assignments that will allow you to familiarize yourselves with Java bytecode and the tools we will be using. These will be worth a total of 10% of your grade.
2. You will work in a team of 2 on a project within the SandMark framework. Typically, this will involve implementing a Java obfuscation or watermarking algorithm, but other ideas of your own are welcome. This is worth 60% of your grade.
3. You will present 1–2 papers or patents. This is worth 20% of your grade.
4. There will be a number of quizzes during the class to make sure you are reading the papers and participating in class. These will be worth 10%.

Slide 0–6

More Rules

1. You will make a presentation of the finished project to the class.
2. 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.
3. 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 0–7

More Rules...

4. 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.

The Presentations

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

1. Select one of the papers I have chosen for us to read.
2. Coordinate with me to make sure no one else has chosen the same paper.
3. Read (and understand...) the papers.
4. Write a ($\approx 3 - 4$) page summary. Use the LaTeX typsetting system.
5. Come talk to me about the paper.
6. Prepare the presentation. Make copies of the summary for the audience.
7. Give the presentation (25 minutes + 5 minutes discussion).

Reading a Research Paper

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...

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

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 $\lambda \sum_1^n(\Gamma(n))$!
   - Are they awake? If not, make sure everything's on the handout!
3. Avoid too much technical detail!!!
4. Avoid too much technical detail!!!

Slide 0–12

Preparing a Presentation...

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?

Slide 0–13

Giving the Presentation

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...

Slide 0–14