General Information

Instructor

Matt Bishop
Office hours: M 3:00PM–4:00PM and Th 4:00PM–5:00PM Pacific Coast time, or by appointment
Office: 3059 Engineering Unit II *Email*: bishop@cs.ucdavis.edu
Phone: (530) 752-8060 *WWW*: http://seclab.cs.ucdavis.edu/~bishop
Note: Please put *ECS 253* in the subject of all email to help me see it quickly!

Lectures

TuTh 12:10PM–1:30PM in Room 1070, Banier Hall

Course Outline

Elements of cryptography and data security; system security, and network security. Both theory and applications will be covered, but theory will be emphasized.

Course Goals

Some goals we hope you achieve:
1. learn the importance of computer security;
2. understand how to use cryptography in support of security services
3. learn the basic theory and practise of secure systems;
4. understand the types of security services needed for network security; and
5. analyze or survey some aspect of computer security and cryptography in depth.

Text

We will be using draft chapters of a book in preparation (*Computer Security: Art and Science*). These will be available at the bookstore.

Computer Programs

The homework assignments, and your project, may require computer programs. Any computer programs written for this class must be well documented, cleanly written, and have a manual page or write-up describing how to use it, its input, and its output. Include sample runs. If you have C or C++ available, I would prefer you use one of those; if not, please check with me.

Course Web Page, Handouts, and Newsgroup

The web page *http://nob.cs.ucdavis.edu/~ecs253* contains links to all course handouts (except for the published/copyrighted papers). If that is not available, go to my web page and follow the link in the "Quick Index" section.
Because we have some students without access to the UC Davis campus newsgroups, information about this class, homework assignments, office hours, and so forth, will be posted to the web page as well as to the *ucd.class.ecs253* newsgroup. Read this newsgroup (or web page) daily, especially near the time assignments are due. You are responsible for everything posted. This newsgroup is not for discussion about the class, for but information from the instructor to you.
If you want to post things about the class, please use the discussion newsgroup *ucd.class.ecs253.d.*, or send the instructor a mail message asking that something be posted. Discussing something in this group is perfectly fair! Postings from both newsgroups will be copied to the web page regularly.

Homework

There will be 5 homework assignments. The due date will be on each assignment. I will try to have your homework

graded as quickly as possible, usually within three class periods after I receive it.
Homeworks are due on the given date. If you have a reason for turning it in late, please discuss it with me. I'm quite liberal about due dates in a graduate class, but I do *not* want to have to grade lots of papers at the end of the term, and I cannot post answers until all the homework is in. If you turn homework in late without making arrangements first, I will take 10% off your score for every day it is late (including weekends, holidays, and any other classification you care to use!)
Please think your answers through before writing them down in final form; a request for a proof requires a proof, not a statement that "it's probably right, and here are 15,000 examples to show it;" a request for a discussion should be treated as an essay question, with a main theme and arguments for and against the answer. It is fair to present the factors that affect your answer; it is not acceptable to begin by giving one answer in the introduction and a different answer in the conclusion! (Yes, you'll lose points.) And, always show your work; if you simply write down a correct answer and do not show how you got that answer, you will not get any credit.
All homework must be submitted electronically, in text, Postscript, or PDF format. Please use the *handin* program described in *All About Homework*.

Project

This class requires a term project requiring you to do outside reading, or apply what we've learned in class to a realistic situation, or extend your knowledge beyond what is done in class. The project is an integral part of the course, because it demonstrates you've learned enough to go beyond what we talked about in class. The section **Projects** describes the requirements in some detail and suggests possible projects, as well as the required intermediate reports.

Grading

50% Homework
50% Project
Note that there are no exams.

Academic Integrity

Please see the Spring 2001 *Class Schedule and Room Directory* for a general discussion of this. In particular, for this course:

- All work submitted for credit must be your own. You may discuss your assignments with classmates, with instructors, or with readers in the course to get ideas or a critique of your ideas, but the ideas and words you submit must be your own. Unless explicitly stated otherwise in the assignment, collaboration is considered cheating and will be dealt with accordingly.
- For written homework, you must write up your own solutions and may neither read nor copy another student's solutions. You are at liberty to do research on a problem, but if you use outside sources, you must cite them. Please do *not* copy an answer from somewhere els, because that defeats the purpose of the homework (which is to give you practice working with concepts and methods).
- For programs, you must create and type in your own code and document it yourself. Note that you are free to seek help while debugging a program once it is written.

A good analogy between appropriate discussion and inappropriate collaboration is the following: you and a fellow student work for competing software companies developing different products to meet a given specification. You and your competitor might choose to discuss product specifications and general techniques employed in your products, but you certainly would not discuss or exchange proprietary information revealing details of your products. Ask the instructor for clarification **beforehand** if the above rules are not clear.

Syllabus

#	Date	Topic, Readings, and Other Information
1.	Tuesday, April 3	Introduction to Computer Security *Reading*: §1
2.	Thursday, April 5	Foundations Part 1: Access Control Matrix, HRU *Reading*: §2, §3.1–3.2
3.	Tuesday, April 10	Foundations Part II: Take-Grant, SPM *Reading*: §3.3, 3.4
4.	Thursday, April 12	Confidentiality Policies *Reading*: §4.1–4.4, §5.1, §5.2.1–5.2.2, §5.3 (not 5.3.1), §5.4
5.	Tuesday, April 17	Integrity Models *Reading*: §6
6.	Thursday, April 19	Other Models *Reading*: §7
7.	Tuesday, April 24	Basic Cryptography *Reading*: §9
8.	Thursday, April 26	Cryptographic Protocols *Reading*: §10.1–10.4, §10.6, §11.1–11.3
9.	Tuesday, May 1	Authentication, Identity *Reading*: §12, §14
10.	Thursday, May 3	Design Principles, Access Control Mechanisms *Reading*: §13, §15
11.	Tuesday, May 8	Information Flow and the Confinement Problem *Reading*: §16, §17
12.	Thursday, May 10	Malicious Logic *Reading*: §18
–.	Tuesday, May 15	**no class** (SANS)
13.	Thursday, May 17	Network Security
–.	Tuesday, May 22	**no class** (National Colloquium on Information Systems Security Education)
14.	Thursday, May 24	Formal Methods: Specification, Design

15.	Tuesday, May 29	Vulnerability Analysis and the Flaw Hypothesis Methodology *Reading*: §19
16.	Thursday, May 31	Auditing and Intrusion Detection
17.	Tuesday, June 5	Secure Programming and Administration
18.	Thursday, June 7	*to be arranged*

These topics are tentative and subject to change. If you want to hear about something and don't see it, please let me know—I'm usually quite happy to talk about it!

Projects

Why a Project?

This course covers a very large discipline, and – perhaps more so than many other areas of computer science – the discipline of computer security runs through many other areas. Because the class has a very limited amount of time, we will only touch the surface of many topics. The project gives you an opportunity to explore one of these topics, or some other area or application of computer security that interests you, in some depth.

The specific goal of the project is to produce a paper. The paper may document software (or hardware) work, so you may choose that kind of project. The paper must either be of publishable quality, or be publishable should some (small amount) of additional work be done.

You are free to work singly or in groups. Groups should have between 2 and 4 people; if you want to have more than 4, please check with me *first*.

Suggestions for How to Proceed

First, choose a topic. Good ways to find a topic are to think about an area of computer science you enjoy, and try to relate it to computer security (or vice versa); talk to some other graduate students and see if what they are doing suggests any ideas; think of ways security of the system you're working on could be made better; go to the library and browse for an interesting-looking paper; and so forth. The major computer security journals are *Computers & Security*, *Journal of Computer Security* and the *ACM Transactions on Information and System Security,* but articles appear in almost all journals; the major conferences are *Crypto* and *Eurocrypt* (for cryptography), *Symposium on Research in Security and Privacy, National Computer Security Conference*, and the *Annual Computer Security Applications Conference.* If you need more help or have questions, feel free to talk to me.

Some Suggestions for Project and Report Topics

The following are just to get you thinking. You will need to do much refinement for each!

- Analyze your favorite Internet or network protocol with respect to specific security requirements. Is it adequate, or should changes be made to enhance its ability to meet stated goals?
- Do a historical survey of computer viruses or worms. You will need to examine the differences of types of viruses (or worms) as well as giving a chronology.
- UC Davis has an electronic mail security policy. Is it reasonable or realistic? What are the legal implications? Could you improve it from the point of view of system administration?
- Look at attack signatures and derive a little language to capture some class of them. Can you generalize your language to include as many attacks as possible? Focus on the temporal aspects.
- Add temporal logic to the Take-Grant Protection Model.
- The non-interference and non-deducibility results are related to multi-level security used to protect confidentiality. Can you either extend those results to the Biba integrity model, or set up a similar notion for integrity-based or availability-based models?
- How would you look for non-secure settings of environment variables in an executing program? Can you develop a wrapper that will check those values whenever a subprocess is spawned? (The motive here is that we may not have access to the source code, but can wrap the program so when it executes, the wrapper controls execution and can stop the wrapped program to check state.) You may need to hack a kernel to do this.
- Design and implement Karger's Trojan Horse checking scheme. Be sure you check *login*, *mail*, *etc.* because those are the programs attackers will instrument.
- Pick a class of vulnerabilities, analyze it, and design tools to check for those problems in program. Substantiate any claims of success by implementing a prototype and using it.
- Take a popular security tool and improve it by adding to it, simplifying the user interface, or in some other fashion. Support your claim of having improved it with some tests to demonstrate the new tool does work better.
- *Or whatever you think you will find interesting …*

What Is Due When

All submissions are to be made through the *handin* program.

Tuesday, April 16	By this time you should have chosen your project. Turn in a 2–3 paragraph write-up of what you want to do, and why; list several sources (at least 3), and describe how you plan to go about completing the project. Your submission is to be in HTML format; a template is on the web page. I will post this to the web page.
Tuesday, May 15	By this time your project should be well underway. Turn in a *detailed* outline or design document (the latter if your project is primarily implementation). Be specific about what you are doing, how, and what you expect (hope!) will be the result. Motivation is important; why should anyone other than you care about your result? Again, your submission is to be in HTML format; a template is on the web page. I will post this to the web page.
Thursday, June 7	Your completed project is due. I will not post these on the web page unless you want me to (please indicate this in your submission). If you do, please supply HTML!

All About Homework

This handout describes some general thoughts and techniques for doing homework, as well as what is required, how to submit it, and other administrative matters.

Turning In Homework

All homework is due at noon on the due date, unless noted otherwise on the assignment. (This way, you have no incentive to skip the class while finishing your homework at the last minute!) These will be graded and returned to you as quickly as possible; I'll try for three class periods, but can't guarantee it .

For written homework, you must turn in an ASCII, a PostScript, or a PDF version of your answers (you can use any text processor you like to generate these). If you submit PostScript, please be sure the file will print on our department printers (use *ghostscript* or *gs* to check this; if it displays the file properly, the file should print correctly). If your file is a postscript file, please choose a name that ends in ".ps". If it is an ASCII file, please choose a name that ends in ".txt". If your file is a PDF file, please choose a name that ends in ".pdf".

For programs, turn in the source code and any related information (such as man pages and README files). Be sure that we can recompile it *without errors* by typing "make". You are free to use any programming language that is available on the CSIF and that I can get to. C, C++ or assembly is acceptable. Any of the languages in the programming languages class is acceptable (assuming compilers and interpreters are available in the CSIF), and if you can write your programs in such a way that *troff*(1) or *latex*(1) can execute them, that's fine too. (Yes, someone once wrote a BASIC interpreter as a set of *troff* macros. It was very slow, but it worked.) But use lots of comments!

Please turn in your homework electronically. Suppose you want to turn in the files *answers.ps* and *prog.c* for homework 3. To do this, go to the directory containing both and type

```
handin cs253r hw3 answers.ps prog.c
```

This program will submit your files to the ECS 253 grader (namely, me). A manual page for the *handin* program is attached. You have to do this from the CSIF; *handin* does not work from other systems. If you do not have access to the CSIF, you can email it to me; please include the homework as an attachment if you do this.

Late Homework

Homeworks are due on the given date. If you have a reason for turning it in late, please discuss it with me. I'm quite liberal about due dates in a graduate class, but I do *not* want to have to grade lots of papers at the end of the term, and I cannot post answers until all the homework is in. If you turn homework in late without making arrangements first, I will take 10% off your score for every day it is late (including weekends, holidays, and any other classification you care to use!)

NAME

handin – file submission program

SYNOPSIS

/usr/pkg/bin/handin *touser* [*subdirectory* [*files* ...]]

DESCRIPTION

handin provides a secure means of submitting files to another user, recounting what has already been submitted, and listing what subdirectories exist for containing submissions.

USAGE

Submitting files

With *touser*, *subdirectory* and *files* all specified, each file is copied to *~touser*/handin/*subdirectory*/*fromuser*, named with the original file's *basename*(1), and made owned by *touser*. The directory *fromuser* is made if it doesn't already exist and is named after the invoking user. Each file specified should have a *basename*(1) unique among any files already submitted by that user to *subdirectory*, unless overwriting is desired.

Recounting submissions

Without *files* specified, information on previous submissions by the user to the specified *subdirectory* is shown.

Listing existing *subdirectories*

Run with only *touser* specified, **handin** just lists the existing subdirectories (regardless of accessability).

EXAMPLES

The following examples illustrate the use as a homework submission facility to the pseudo-user ``cs101'' created for this purpose:

```
example1% handin cs101
Existing subdirectories (comments in parentheses):
Asn1    (Due Mar 18)
Asn2    (Due Mar 25)
example2% handin cs101 Asn1 part1 part2
Submitting part1... ok
Submitting part2... ok
example3% handin cs101 Asn1
The following input files have been received:
Thu Mar 17 14:50:49 1994        1599 bytes      part1
Thu Mar 17 14:50:49 1994        3412 bytes      part2
```

SEE ALSO

rcvhandin(8)

DIAGNOSTICS

handin itself provides only a little of the diagnostic information that's given and returns the number of errors encountered as its exit status. Any other information comes from *rcvhandin*(8).

Skipping *file*: file non-existant or irregular
The named file didn't exist or was probably a directory. The user should check to make sure that the file they specified was indeed the file they intended to submit.

Skipping *file*: file not readable
The named file was not readable by the user.

Submitting *file*... failed [: *reason*]
The named file was not successfully submitted. If at all possible a reason is provided by *rcvhandin*(8).

Submitting *file*... ok
The named file was successfully submitted.

NOTES

handin is really just a front-end to the *rcvhandin*(8) program. The primary function of **handin** is to open the named *files* with the effective user ID of the invoking user and pass on their contents to the *rcvhandin*(8) program having the effective user ID of *touser*. This design provides a simple and portable means for implementing a file submission facility in even a non-homogeneous, network-file-system environment.

AUTHOR

Lou Langholtz, Department of Computer Science, University of Utah, 1994