General Information

Instructor

Matt Bishop, 3059 Engineering Unit II; phone: 752-8060;
email: bishop@cs.ucdavis.edu; web page: http://seclab.cs.ucdavis.edu/~bishop
Office hours: MWF 11:00-11:50AM, by appointment or by chance

Teaching Assistant

John Hughes, 053 Engineering Unit II
email: hughesj@cs.ucdavis.edu
Office hours: to be determined

Lecture

MWF 10:00AM – 10:50AM in 1204 Haring Hall

Discussion Section

W5:10 – 6:00PM in 1150 Hart Hall
We will use these to make up some classes. Material presented here ***will be*** on exams.

Course Outline

Introduce principles, mechanisms, and implementations of computer security; learn how attacks work, how to defend against them, and how to design systems to withstand them

Course Goals

Some goals we hope you achieve:

1. learn about security in the UNIX system and programming environments;
2. learn how to attack a system, and to defend it by analyzing the system for vulnerabilities and ameliorating those problems;
3. understand the strengths, and weaknesses of cryptography as a tool of security'
4. learn how access to systems, resources, and data can be controlled;
5. learn the basics of writing security-related programs;
6. learn about security in networks;

Text

There are four required texts for this course.

- Sun Tzu, *The Art of War*, edited by James Clavell, Dell Publishing Co., New York, NY ©1996 (ISBN 0-385-29985-0)
- Niccolò Machiavelli, *The Prince*, Penguin Books, New York, NY ©1995 (ISBN 0-14-044107-7).
- Saul D. Alinsky, *Rules for Radicals*, Vintage Books, New York, NY ©1971 (ISBN 0-394-71736-8).
- John Brunner, *The Shockwave Rider*, Ballantine Books, New York, NY ©1975 (ISBN 0-345-32431-5).

A recommended text is:

- S. Garfinkel and E. Spafford, *Practical UNIX & Internet Security*, Second Edition, O'Reilly and Associates, Inc., Sebastopol, CA. ©1996.

In addition, we shall use parts of the text *Computer Security: Art and Science*. Readings from this text will be distributed in class.

Computers

All registered students have been given an account on the computer science instructional machines in the basement. See the section **CSIF Accounts** for more information. ***Change your password as soon as you can; if it is not changed within a week, your account will be disabled and you will have to see a system programmer to have it reset.***

Course Handouts

Most course handouts, programs, and samples will be available in the directory *~cs153* on any of the CSIF DECStations, and on the WWW. The URL for the class is

http://wwwcsif.cs.ucdavis.edu/~cs153/

It can also be accessed through my web page (look at the *classes* section).
For copies of handouts not on the web site, please see me or the TA.

Class Newsgroup

Information about this class, homework assignments, and so forth, will be posted to the newsgroup *ucd.class.ecs153*. Read this newsgroup daily! **You are responsible for everything posted to this newsgroup.** Please do not post to this newsgroup; we'll use it to put out important information. If you want to post things about the class, please use the discussion newsgroup *ucd.class.ecs153.d*. Discussing something in this group is perfectly fair!

Homework

Homework is due at the beginning of class on the date stated on the homework. See the section **All About Homework** for more information.

Extra Credit

Extra credit in this course will be tallied separately from regular scores. If you end up on a borderline between two grades at the end of the course, extra credit will count in your favor. However, failure to do extra credit will never be counted against you, because grades are assigned on the basis of regular scores. You should do extra credit if you find it interesting and think that it might teach you something. However, it is not wise to skimp on the regular assignment in order to do extra credit.

Grading

30% Homework	20% Midterm exam
30% Term Project	20% Final exam

Exams

Midterm — Friday, November 5, 1998, in class
Final examination — Friday, December 18, 10:30AM–12:30PM
These are open book/open notes exams. No early or late exam will be given; if you miss an exam for medical reasons (you ***must*** document this; no other excuses are acceptable), you may be allowed or required to take a make-up exam, or the other parts of the course will be counted proportionally more (the choice is the instructor's). In particular, forgetting the time or place of an exam is ***not*** an excuse for missing it!

Academic Integrity

Please see the *Fall 1998 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 teaching assistants or 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.
- 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 or a teaching assistant for clarification ***beforehand*** if the above rules are not clear.

Syllabus

	date	topic	reading
1.	Fri, Oct 2	Introduction: what is computer security	COMP; TZU
2.	Mon, Oct 5	Design principles, trust, and risk analysis	COMP
	Wed, Oct 7	***no class*** (National Information Systems Security Conf.)	
	Discussion	Social engineering	SEA
3.	Fri, Oct 9	Robust programming rules and examples	RP
4.	Mon, Oct 12	Robust programming rules and examples (*con't*)	RP
5.	Wed, Oct 14	Vulnerability models, attack models, relationship	VULN
		last day to drop courses	
6.	*Discussion*	Penetration analyses, Flaw Hypothesis Methodology	VULN
7.	Fri, Oct 16	Intrusion detection	INTD
		homework #1 due	
8.	Mon, Oct 19	Classical cryptography; ROT-13, Enigma, DES	DES
9	Wed. Oct 21	Authentication; *crypt*(3); Diffie-Hellman	AUTH
		last day to add courses	
		project selection due	
10.	*Discussion*	Passwords; good vs. bad, aging, one-time, challenge-response	SKEY, RSA
11.	Fri, Oct 23	Attacks: password guessing, scavenging, spoofing, others	
	Mon, Oct 26	***no class*** (Network Security Conf.)	
	Wed, Oct 28	***no class*** (Network Security Conf.)	
	Discussion	The DOVES Project	
12.	Fri, Oct 30	User IDs, privilege, and access control bits	IDEN
		homework #2 due	
13.	Mon, Nov 2	Access control lists, access rings	CA, MACH
14.	Wed, Nov 4	Capabilities, file descriptors; TOCTTOU flaws	RACE
		last day to change to P/NP or S/U grading	
	Discussion	Review for midterm	
15.	Fri, Nov 6	***midterm***	
16.	Mon, Nov 9	Mandatory access controls, compartments, Bell-LaPadula	MSEC
17.	Wed, Nov 11	Lattice models and Biba, Lipner's Commercial Model	MINT
		project design/outline due	
18.	*Discussion*	Clark-Wilson	
19.	Fri, Nov. 13	Chinese Wall, procedural controls	CWM
		homework #3 due	
20.	Mon, Nov. 16	Malicious logic: viruses, logic bombs, malicious worms	CA, ALIN
21.	Wed, Nov. 18	Other types of access controls: role-based, originator control	OAC
22.	*Discussion*	Types as protection; limiting writing, isolation, quarantine	
23.	Fri, Nov. 20	*to be determined*	

	date	topic	reading
24.	Mon, Nov 23	Checking: cryptographic checksums, file system analysis	
25.	Wed, Nov 25	Trust in environment: shell variables, process environment **homework #4 due**	BRUN
	Discussion	*tripwire*and other UNIX-oriented integrity tools	TRIP
	Fri, Nov 27	***no class*** (Thanksgiving)	
26.	Mon, Nov 30	Network threat models; Internet security architecture	NB
27.	Wed, Dec 2	Network-based attacks and defenses	CSAS, §21
	Discussion	*tcp_wrappers* and other UNIX network security tools	VENE
28.	Fri, Dec 4	Denial of service (network and otherwise))	
	Mon, Dec 7	***no class*** (Annual Computer Security Applications Conf.)	
	Wed, Dec 9	***no class*** (Annual Computer Security Applications Conf.)	
	Discussion	Review for final	
29.	Fri, Dec 11	Limits of security; the HRU result **homework #5 due**	HRU

Key: all except the four class texts will be handed out in class
ALIN: Saul D. Alinsky, *Rules for Radicals*
AUTH: M. Bishop, "Authentication and Identification" (handout)
CA: M. Bishop, "Controlling Access" (handout)
COMP: M. Bishop, "Computer Security" (handout)
CWM: M. Bishop, "Chinese Wall Model" (handout)
DES: National Bureau of Standards, "The Data Encryption Standard".
HRU: M. Bishop, "Basic Fundamental Results" (handout)
IDEN: M. Bishop, "Identity and Authentication" (handout)
INTD: M. Bishop, "Intrusion Detection" (handout)
MACH: Niccolò Machiavelli, *The Prince*
MINT: M. Bishop, "Commercial Policies" (handout)
MSEC: M. Bishop, "Mandatory Security Policies" (handout)
NB: M. Bishop, "Network Security Basics" (handout)
NP: M. Bishop, "Network Protocols" (handout)
OAC: M. Bishop, "Originator-Controlled Access Control and Role-Based Access Control" (handout)
RACE: M. Bishop, "Race Conditions" (handout)
RP: M. Bishop, "Robust Programming" (handout)
RSA: R. Rivest, A. Shamir, L. Adleman, "Digital Signature Implementation"
SEA: I. Winkler, "Social Engineering"
SKEY: N. Gorsuch, "S/Key"
TRIP: E. Spafford, "Experiences with *tripwire*"
TZU Sun Tzu, *The Art of War*
VENE: W. Venema, "*README*, tcp_wrappers distribution
VULN: M. Bishop, "Vulnerability Analysis" (handout)

CSIF Accounts

The staff has created instructional accounts for all students who are registered for Computer Science courses in Winter Quarter 1998 but who did not already have an account on the Computer Science instructional machines. Find the list of new accounts downstairs. The list is alphabetized by last name. Look down the list until you find your name. In the "Account Name" column is the name of the computer account that has been assigned to you. It's also called a "login name". (It does not apply to IT [Information Technology, essentially the campus computer center] accounts, to ACS [Academic Computing Services, a unit of the College of Engineering] accounts, or to ECE [Electrical and Computer Engineering] accounts. It applies only to your Computer Science account.) The Computer Science instructional machines are in the basement of the east wing of building Engineering II.

If your name does not appear on this list, either you already have an account on the Computer Science instructional machines, or your name was not included on the class lists that we got from the registrar. The latter may occur if you have added a Computer Science class, as opposed to pre-registering for one. In this case, your account will be added the day after the department receives notification from the registrar that you have added the class. Your account name and login name will then be posted as described above.

An initial password has been assigned to your account: the last 8 numeric digits of your student id number. (The student id number has 9 numeric digits in it, so omit the first one, and omit the hyphens to form your initial password.) Because your student id number is something that other people can probably find out , a password based on your student id number is not very secure. That is, someone other than you could login to your account, including very nasty people. Since you are the only one who is authorized to use your account, it is imperative that you change your password as soon as possible. Use the *yppasswd*(1) command when logged in to the Computer Science instructional machines to do so. If you have not changed your password by 5pm on the Monday of the second week of classes, logins on your account will be disabled. To get logins on your account re-enabled, stop by room 047, where the student assistants are located. You will need to bring your student id card.

Disk quotas have been installed. You may store no more than 10 MB of data in your home directory.

If you have any questions, send mail to "support".

Computers

All programs and commands written for homework must work on any of the DEC, HP, and SGI systems unless specifically stated otherwise in the assignment, since security problems often arise when porting software. The workstations are named as follows:

dec1, ..., dec62	Digital Equipment Corp. workstations running Ultrix
sgi1, ..., sgi23	Silicon Graphics, Inc. workstations running IRIX
hp2, ..., hp25	Hewlett-Packard workstations running HP/UX

Please be aware that some of the workstations may be down at any given time, so if the one you try doesn't seem to work, try another one.

You can log on to the instructional systems in two ways: directly into a workstation in the basement of Engineering Unit II, or through a network login program (*rlogin*(1) or *telnet*(1)) on the UC Davis network.

All About Homework

The homework will consist of both programming exercises and written questions. This handout describes some general thoughts and techniques for doing homework, as well as what is required, how to submit it, how late homeworks are handled, and other administrative matters.

Turning In Homework

All homework is due at the **beginning** of the class 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; we'll try for three class periods, but can't guarantee it .

For written homework, please turn in either an ASCII or a PostScript 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 they display it properly, it should be okay). If your file is a postscript file, choose a name that ends in ".ps"; if it is an ASCII file, please choose a name that ends in ".txt". For programs, turn in the source code and any related information (such as man pages and README files).

For programs, you are free to use any programming language that is available on the CSIF and that the ECS 153 graders can get to; so 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!

Turn in both your written exercises and programs electronically. Suppose your program for homework 3 is in the files *answers.ps* and *prog.c* Then, to turn it in, say

```
handin cs153r hw3 answers.ps prog.c
```

This program will submit your files to the ECS 153 grader. (A manual page for the *handin* program is attached.)

Doing Written Exercises

When you are asked to analyze something, or explain something, please be complete, and ***show your work*** (including any commands you give, and their output, to show how you did the problem); otherwise, even if you get the right answer, you will get *ZERO* points. Think your answer through and do a rough draft. Write clearly and cogently. If the question asks for an opinion, state your opinion clearly, justify it, and don't ramble. Answers which start, "My opinion is yes …" and conclude with " … on the other hand it could equally well be no" won't get much credit.

Doing Programming Exercises

We cannot overemphasize the importance of taking time to design your program, or outline and draft your answer, thoroughly. More programming problems arise from improper design than anything else, and the few hours you spend on design will be amply repaid by shorter coding and debugging phases. So do think the design and interfaces through, and — as always — try to find the simplest way to do the assignment (within the limits given in the assignment, of course)!

Do not leave assignments for the last minute. The assignments are non-trivial and will require significant design time before you start programming and debugging. When we decide on the due dates, we assume you will spend significant amounts of time on design as well as coding and debugging. If you choose not to do this, you will have difficulty finishing the assignments on time.

We do not mind being asked for help; indeed, we welcome it because it helps us know what the students are finding difficult or confusing, and sometimes a few words about the problem in class will clarify the assignment immensely. We **do** mind being asked for help before you have tried to think the problem through; the classic objectionable question (this really happened) occurred on a homework assignment in which the class was given a buggy program. The assignment said the program did not work, and the homework was to debug it and make it work. Within 10 minutes of the end of the class during which the assignment was given out, the instructor got this request for help: "The program doesn't run. What do I do now?"

So, before asking for help, please be sure that you have:

- spent a significant amount of time on the design of your solution;
- used a debugger (or the debugging commands of the simulator) if the problem is a programming bug;
- read all relevant handouts, sections of the textbook, and news articles (because your question may be answered there); and
- tried everything you could think of to solve the problem.

When you come to us, or send us a note, asking for help, please bring whatever you have done to solve the problem, because the first question we will ask you is "What have you tried to solve the problem?" This isn't because we think you're wasting our time; it's because understanding how you have tried to solve the problem will help us figure out exactly what your difficulty is and what we can do to help you. Remember, we will do everything we can to avoid solving the problem for you; when we give you help, our goal is to help *you* solve the problem yourself.

What We Look For In Programming Exercises

When we grade your homework, we look for simplicity, clarity, elegance, and documentation. Here's a rough weighting of the various factors that go into the grade of each program:

Correctness	60%
Commenting, ease of reading	20%
Clean, readable output	10%
Documentation (README, man page, *etc*.)	10%

If a program does not compile (or assemble), the maximum you can get is 30% of the value of the program. So check your programs before you submit them!

Late Homework

You can turn in your homework up to one class period late (unless the assignment says otherwise). If you turn it in late, we will grade it normally, and then deduct 20% as a late penalty. Requests for exceptions will be handled on a case-by-case basis.

Grade Appeals

If you feel that there is an error in grading, please come see me or the TA and we'll look over it (and possibly talk with you about it). However, don't dally; any such request must be made within one week of when the grades were made available. After that, we won't change your grade.

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

Term Project

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 is to give 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 Ground Rules

You may select a project from the list below (in most cases, you will need to refine or limit the suggestions). You may also think of a project on your own. The project can be a detailed research report or survey, or a programming project. In any case, ***check with me*** before beginning to be sure it is a reasonable project and no-one else has chosen it. Please select something that interests you!

Some Suggestions for Project and Report Topics

- Malicious logic and biology: how computer worms, viruses, *etc.* compare to their biological counterparts
- Security requirements in an academic environment (or another environment; medical environments are a hot topic right now)
- Automating policy checking (to ensure your computer/site meets a given policy) and/or definition
- Authenticating users and systems (especially over untrusted networks)
- Factoring a number
- Electronic voting machines and computer security
- Modifying access control mechanisms to the UNIX system (for example, adding rings or capabilities)
- Rights and amplification of rights in a capability-based system
- Secure electronic mail: proposed standards
- Design a program (or set of programs) to break a cipher; for example, a cryptographers' toolkit (you will have to narrow this down a *great deal*)
- Analyzing and/or testing programs for vulnerabilities (pick a couple as examples)
- Intrusion detection and incident response (incident response is a new, and very hot, area right now)
- Write a large (useful) program using the techniques we discussed in class, and argue convincingly why it is "secure" (mail server, WWW server, *etc*.; these may have limited functionality)
- Analyzing a system's or site's security
- Security features of IP version 6 (or ATM, or SSL, or another protocol): how good are they?
- Comparing Windows NT security tools and UNIX security tools (with respect to functionality, trustworthiness, ease of use, *etc*.)
- Developing a security tool (you can pick what you want to write, but please check with me first!)
- Attacking systems; how, who, why, and so forth

Time Line

You must turn in the following. Use the *handin* program to submit electronic copies, as described in **All About Homework**.

date	weight	what
October 21, 1998	10%	Project selected
November 11, 1998	30%	Design or outline completed.
December 18, 1998	60%	Project completed.