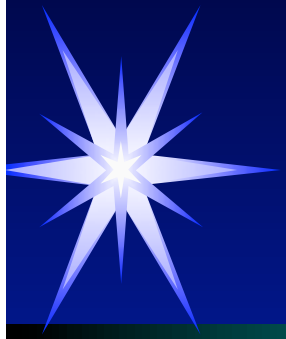
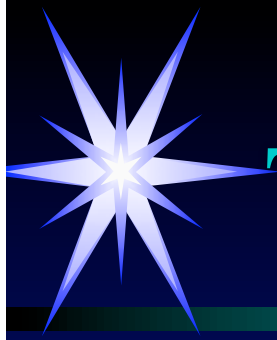


Learning and Experience in Computer Security Education

Matt Bishop
Department of Computer Science
University of California at Davis
Davis, CA 95616-8562 USA

email: mabishop@ucdavis.edu
phone: +1.530.752.8060



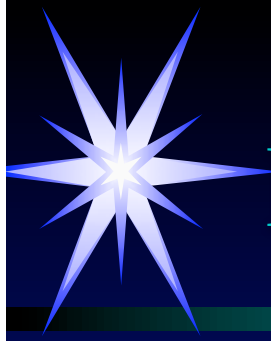
Theme of This Talk

Del dicho al hecho, hay mucho trecho

— Spanish proverb

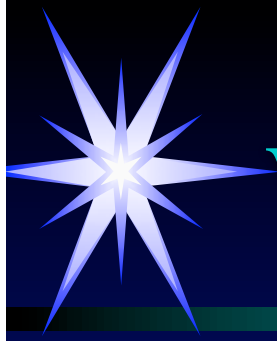
Types of Education

- Academic
  - Emphasizes how things work, and why
- Vocational
  - Emphasizes how to do things with current technology



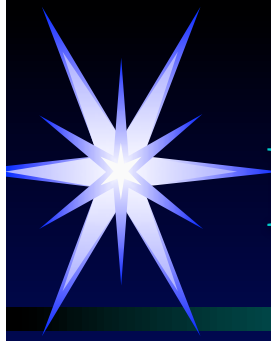
Importance

- Applied computer security in great demand
  - Compared to other computer science jobs
  - Taking into account the worldwide economic crisis
- Computer security research draws on real problems
  - "Devil is in the details" applies here
  - Environment constrains the definition of the problem, as well as its solution



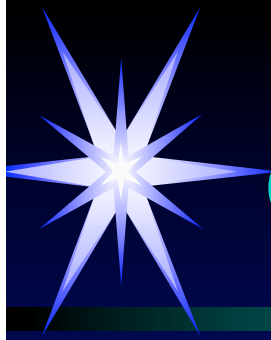
Why?

- Applying what you learn to real problems, situations makes abstract knowledge, theories real
- Also gives practice in interpreting and modifying models in light of real-world constraints



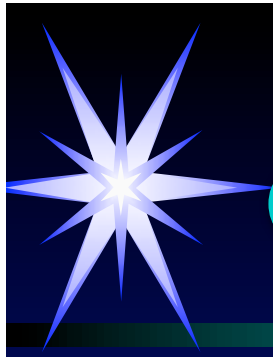
How?

- Work with non-academic organizations
  - Deal with current security problems
  - Will deal with future security problems
- Must recognize their interests
  - What can courses, projects do to advance *their* programs?
  - How can courses, projects advance students' understanding of, skills in, the subject?



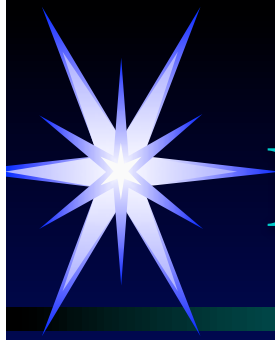
Goals of Interaction

- Depends on the nature of the organization
- Two general forms
  - Consumer organizations (users of security)
  - Producer organizations (produce security)



Consumer Organization Goals

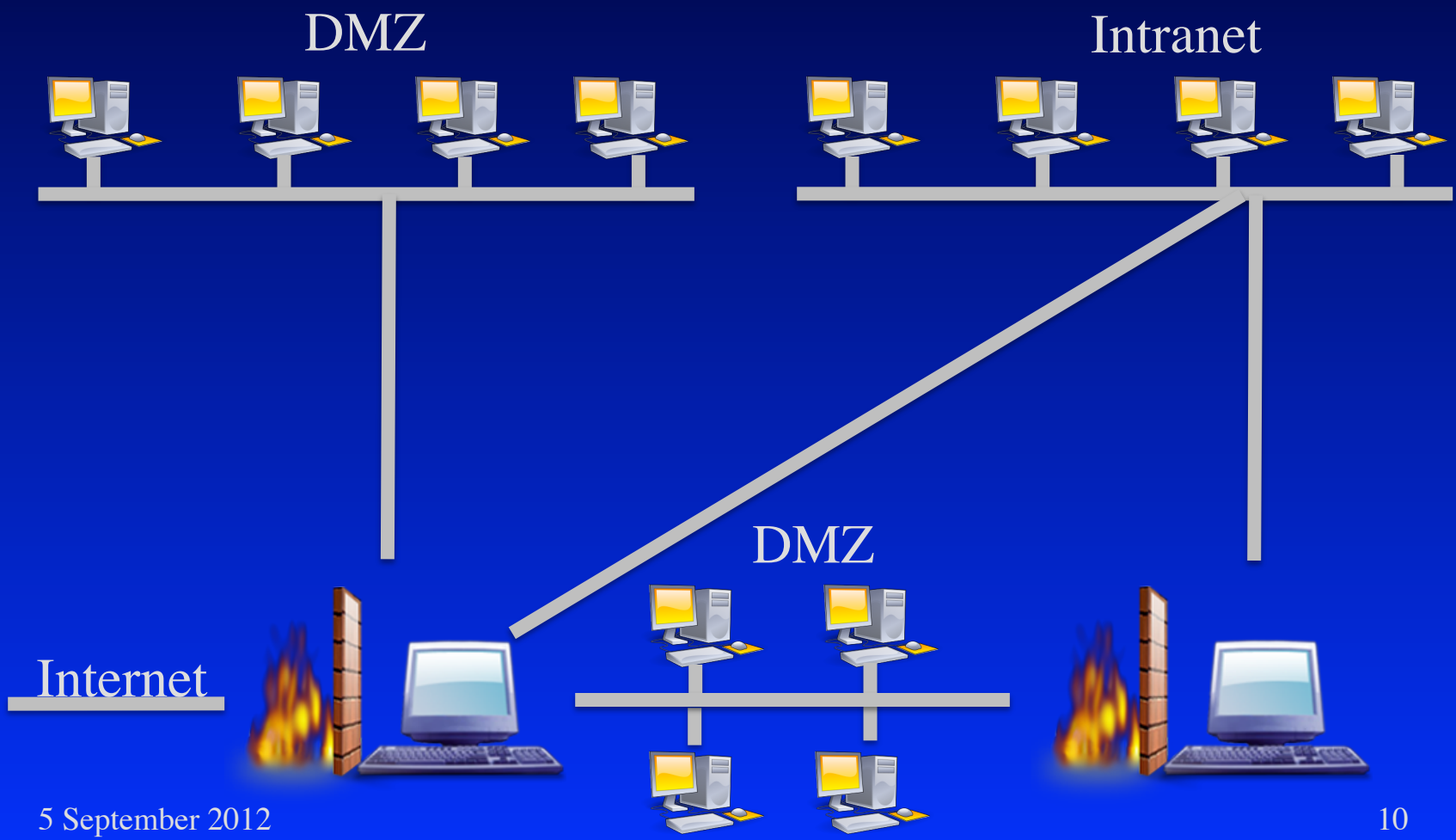
- Learn about the environment in which the organization functions
- Learn about how the organization's mission and environment affect security needs
- Example: organization has public web site
  - What are its security needs?
  - What are the effects of not understanding this?
  - Amazon, Privalia, redcoon, eBay: make information available; lack of availability hurts business
  - Military: make presence known; lack of availability does not interfere with main mission

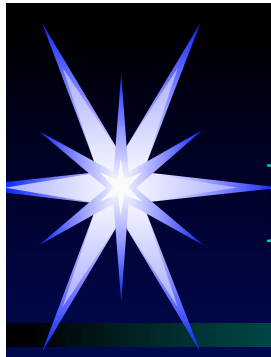


Key Point: What Is Security?

- Varies among different organizations
  - Defined by “security policy”
- Example: “securing cyberspace”

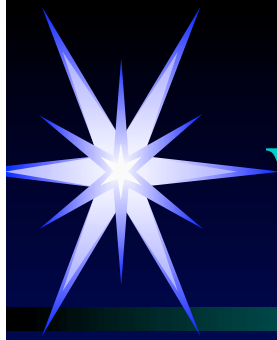
Example: Firewalls and DMZs





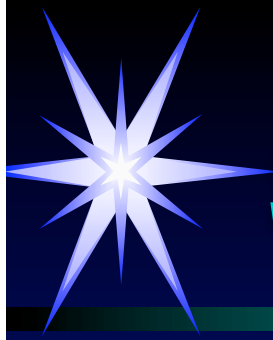
External Factors

- What factors affect security of an organization?
  - Personnel
  - Organizational structure
  - Political
  - Legal
    - US: DMCA, export restriction laws on cryptographic technology
    - Spain: Ley Sinde, Ley 32/2003



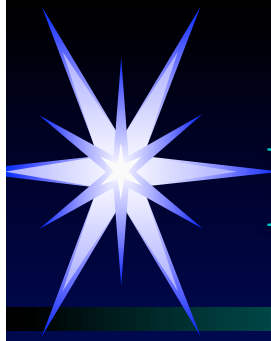
Why Not Textbook Examples?

- Tend to focus on implementation of systems more than policies, procedures, processes
- Implementations, especially configurations, may change rapidly
- Need to be able to adapt to these
  - Or deal with changes equally rapidly!



Summary: Consumer Goals

1. To experience how the technical and non-technical considerations affect the security requirements
2. To learn how those same considerations affect the implementation of those requirements
3. To put into practice the theory studied in school, whether the specifics of the systems being used were discussed in class

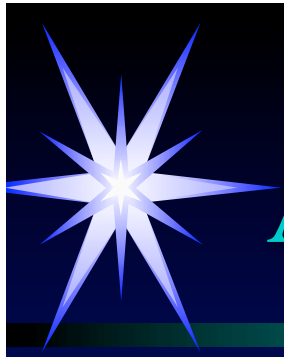


Producer Organization Goals

- Those of consumer organizations
- Develop requirements for marketable products
- Analyze new attacks
- Implement, deploy, manage security products

Developing Requirements

- Market concerns and forces
- What are the real threats?
- What do (potential) customers *perceive to be* the real threat?

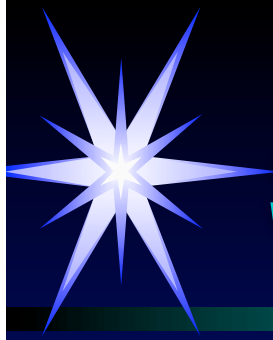


Analyze New Attacks

- It's forensics!
  - Both technical and non-technical factors
- Doing this in class gives experience in a controlled environment
  - In real life, more complexity, time pressure, less information
- Can also practice explaining to non-technical people

Quality of Product

- Learn to design (plan) well
- Learn to implement, test well
- Human factors
  - Audience
    - UNIX (by programmers for programmers) vs. Windows (for everyone)
  - Use
    - Mental models: go from Microsoft Word to LaTeX (or *vice versa*)



Summary: Producer Goals

- Goals of consumer organizations
- To develop requirements to meet specific market needs and pressures
- To identify and analyze new threats in an environment where time is critical
- To design, implement, and deploy robust software

Methods of Interaction

- Internships
- Organization members as teachers, mentors
- Joint research

Internships: For the Student

- Work as an employee or trainee
- As part of their job, learn how to apply what they studied in class
- Learn what they do not know (and need to know)

Internships: For the Academy

- Learn what organizations deem important for students to know
- Get ideas on what to add, delete from the curriculum or from exercises
- Create contacts for joint research, funding
  - Via EU, State, Comunidad Autónoma, Province
  - But government funding in Spain is being cut
    - As it is everywhere else . . .

Internships: For the Organization

- Short term: get a well-educated worker
- Long term: potential hire?
- Very long term: influence how their area of computer science is taught
  - Curricula set in a variety of ways
  - Usually requires review by an accrediting agency such as ANECA National Agency
  - Accreditors may or may not use standards (in Spain, standards are from the European Space for Higher Education)

Guest Lecturers

- Explain (show) the practice of security
- Focus on *why* it is done the way it is done
- Use dialogue to involve students in discussion
- Discuss alternatives, advantages and disadvantages

More Involvement

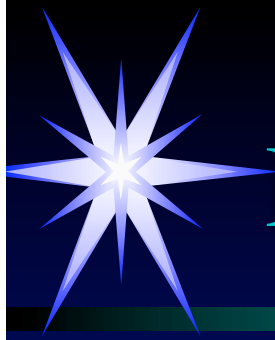
- Create, evaluate laboratory exercises
- Have students study systems, etc. under supervision of employees
- Show students how to apply work in realistic exercises

More Involvement

- Joint research, company- or other-funded
- Students, faculty work together with organization to define, study problems
- Organizations get benefit of academic research while supporting student, faculty work

Benefits

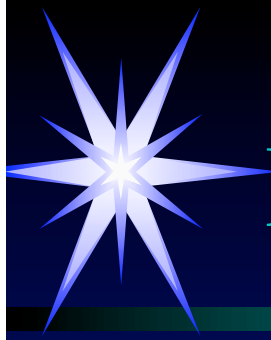
- Students gain varying degrees of practical experience
- School gets support for students, stays current in examples and importance of theory
- Organization gets benefits of access to intelligent, educated people



For What Follows

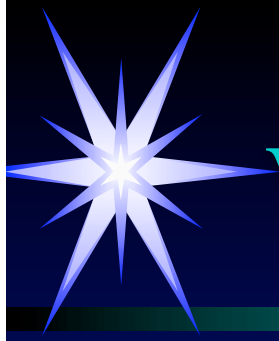
En todas partes cuecen habas y en mi casa, a calderadas

— Spanish proverb



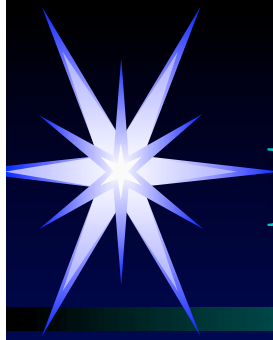
Example: Robust Programming

- Software quality widely regarded as poor
- Affects security, because a "secure" design doesn't necessarily mean a "secure" implementation
- One proposed solution is to require schools to teach "secure coding"



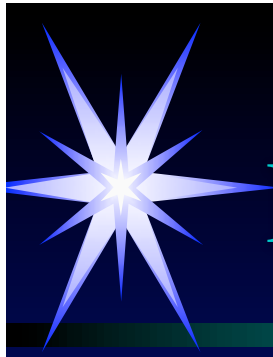
What Is It?

- *Robust programming* prevents abnormal termination, unexpected actions
- *Secure programming* satisfies (stated or implicit) security properties
- Example: buffer overflows
  - *Always* non-robust programming
  - May *or may not* be non-secure programming



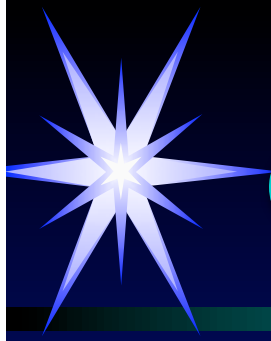
Basic Principles of Robustness

- Paranoia
- Assume maximum stupidity
- Don’t hand out dangerous implements
- “Can’t happen” means it can happen



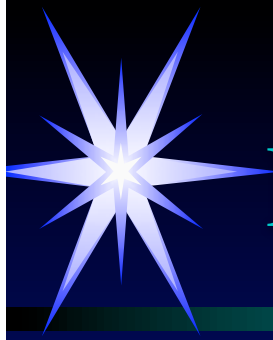
Problem

- We don't build systems that meet security requirements
- We don't write software that is robust
  - Some exceptions in special cases
- Many different models for developing software
  - Agile, waterfall, rapid prototyping, . . .



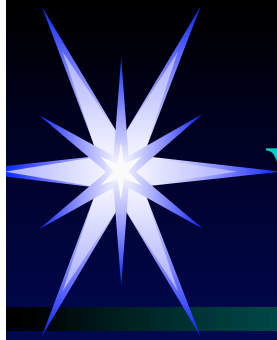
Quality of Code

- Underlying all this is *programming*
  - When coding, you make assumptions about services, systems, input, output
  - Other components you rely on have bugs or may act unexpectedly
- Hard to have robust, secure software when the infrastructure isn't



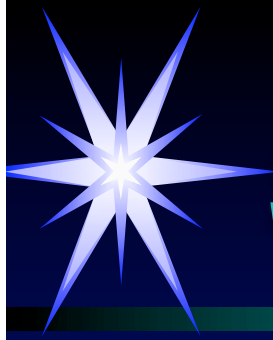
Problems in Teaching This

- Finding real examples
- Finding educational materials
- Adding this to curriculum
  - Reinforcement is a serious problem!
- Finding people who can teach this



Where Students Learn This

- First class in programming
- Examples in other classes; best if drawn from real life
- Not usually reinforced in advanced courses

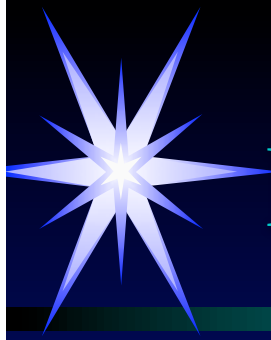


So, How Do We Reinforce This?

- Show practical effects of it to motivate students
- Show actual examples of robust, non-robust programming
- Analyze programs that students write for non-robust code

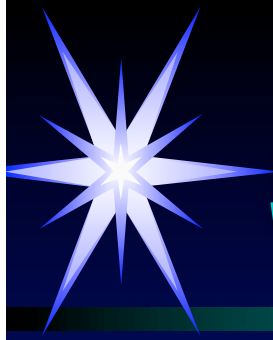
Other Ways To Learn It

- Use special code-checking tools (industry standard)
  - Teaches true, false positives
- Use special languages and development environment
  - No panacea, and rarely used in industry, but every little bit helps
- Use code management tools like CVS, SVN, *etc*.



How To Assess It

- Graders assess robustness as well as grading the assignment
- Clinic to which students can bring programs, get help and advice on checking them
- Suggestion: figure out ways to avoid adding extra classes
- Doing this in multiple classes reinforces material in all programming contexts



Staffing Issues

- Analysts could be grad students with experience in this
- Better: non-academic organizations have folks too — have them start the clinic, train students
  - Organization benefits by enhancing mentor knowledge
  - They can identify prospective future hires
  - Students benefit by working with experienced practitioners
- Same ideas, methods can be applied to non-CS majors

How *Not* To Do This

- Think of academic education as *distinct* from practice
- We teach understanding only; practice is "trivial"
- Students will learn implementations, practices later
  - Problem is that theory without practice is abstract

Example

This truth was first brought home to me more than thirty years ago one December day, as I walked down the road from Argentières to Chamonix after a snowfall, and suddenly from the abyss of unconscious memory a line of Virgil rose into my mind and I found myself repeating

Sed iacet aggeribus niveis informis, et alto

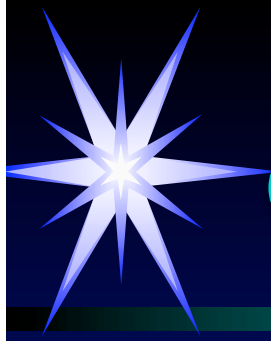
Terra gelu.

I had read the words at school and no doubt translated them glibly "the earth lies formless under snow-drifts and deep frost"; but suddenly, with the snow scene before my eyes, I perceived for the first time what Virgil meant by the epithet *informis*, "without form", and how perfectly it describes the work of snow, which literally does make the world formless, blurring the sharp outlines of roofs and eaves, of pines and rocks and mountain ridges, taking from them their definiteness of shape and form. Yet how many times before that day had I read the words without seeing what they really mean! It is not that the word *informis* meant nothing to me when I was an undergraduate; but it meant much less than its full meaning. Personal experience was necessary to real understanding.

(Sir Richard Livingstone)

How To Look At This

- Students understand best when they see, or do
- Working together, you get practice placed on a firm *theoretical* and *conceptual* foundation
- Practice is an integral part of building understanding



Conclusion

One must learn by doing the thing; for though you think you know it, you have no certainty, until you try

—Sophocles

Thanks to . . .

- My Spanish colleague Urko Zurutuza and Iñaki Arenaza for helpful comments, and especially for inviting me
- You, for listening!

This work was supported by the U.S. National Science Foundation awards CCF-0905503 and CNS-1039564. All opinions expressed are those of the author, and are not necessarily those of any other organization or person.

¿Preguntas?

¡Respuestas! (Espero . . .)