Secure Programming: A Way of Life (or Death)

Matt Bishop
Computer Security Laboratory
Dept. of Computer Science
University of California at Davis
Disclaimer

- All opinions expressed here are not necessarily those of the:
  - Computer Security Laboratory
  - Department of Computer Science
  - University of California
  - Any other organization mentioned in here
  - U.S. government and any of its agencies
  - Anyone else you can think of
How important is secure coding?

If it is so important, why aren’t we “priming the pump” seriously?
Weinberg’s Second Law

If builders built buildings the way programmers wrote programs . . .

then the first woodpecker to come along would destroy civilization

The Psychology of Computer Programming (1971)
What *Exactly* Are We Talking About?

- 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
Secure programming requires robust programming

You can’t have realistic security without robustness
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
Security is Cumulative

- Composing non-secure modules produces non-secure software
- Can ameliorate this with shims to handle non-secure results
  - Shims provide the security
  - What if they themselves are written, installed, etc. non-securely?
  - What if they can be bypassed?
More Problems

- Refactor code, use external libraries, modules, services
  - You inherit their bugs and assumptions!
- Example: RSAREF2 library buffer overflow (1999)
  - Affected ssh, anything using that library
- Move code into different environment, assumptions may not hold
Basic Principles of Robustness

- Paranoia
- Assume maximum stupidity
- Don’t hand out dangerous implements
- “Can’t happen” means it can
Helping Solve the Technical Problem

- Isolate assumptions, make them explicit
  - Not just your code, but also about what your code calls
  - Allows you to build (informal) preconditions, postconditions for others to use
- Make them part of the documentation (or comments)
- Identify those relating to security explicitly
  - Good judgment needed here
Test to These!

- Make sure your assumptions hold in the environment(s) you intend the software to be used
  - If you state them explicitly, users can check them
  - Think *autoconf* to check these on installation
  - Warn if not met!
What Will Drive Improvement?

- Commercial incentives
  - Financial savings from not cleaning up so many messes
  - Maintenance simpler
  - Better reputation (of products, company)
  - Easier to bring new programmers, software engineers up to speed on products
  - Issues of software liability
What Will Drive Improvement?

- Government incentives
  - Financial savings from not cleaning up so many messes
  - Maintenance simple
- Better defenses for national security
You can’t say “I’m not responsible for anything”

Chain of distribution (i.e., supply chain) liability exists now

You can limit liability somewhat by defining use, environment

Then you’re liable in that context but (probably) not in others

It is coming . . .

EULAs may not be enforceable (“contracts of adhesion”)
So What’s Holding Us Back?

- **Commercial**
  - No legal liability for bad software
  - More expensive to make; longer time to market
  - Lack of people who write robust code

- **Government**
  - Need to spend more money to get it
  - Need to pay more attention to installation, maintenance, use
  - Need to recruit people who can use it well
So What’s Holding Us Back?

- **Academic**
  - Robust coding not seen as integral to programming
    - Textbooks *loaded* with examples of non-robust programming
  - Lack of support for *enforcing* and *grading* for it in non-introductory classes
    - Ties into lack of graders who really know about this
  - Lack of faculty who understand robust programming
    - And intimidation factor for those who *know* they don’t understand it
Lack of Resources

- Assurance costs!
  - Industry expected to deliver secure, robust products without offset for the extra effort in delivering them
  - Academia expected to teach *and reinforce* robust programming without offset for the extra effort in supporting this
Lack of People

- Need to teach people how to write robust programming
  - Need to emphasize this in the practice, supporting it both in education and industry
- Continuous practice is central to reinforcing, maintaining, extending skills
Focus for Rest of Talk

- How can industry and government work with academic institutions to do this?
  - Carrots, not sticks
  - Security tuned to environment, use
  - What is “secure” varies among companies and government organizations

- Everyone lacks resources!!!
How Do We Teach Secure Programming?

- SESS report has suggestions

- Key conclusion: *no one sector can improve the state of the practice on its own*
  - “We must all hang together, or we shall all hang separately” (B. Franklin)
Questionable Idea #1: Testing Students’ Knowledge

- Who creates the tests?
- Who is being tested?
- How do you know that you are testing what is important (that is, the “right thing”)?
- Who determines what is an acceptable result?
- Teaching for the test, not the material
Questionable Idea #2: Unsupported Mandates

- The support has to come from somewhere
  - It’s like a zero-sum game
- What do you want to weaken?
  - If you only have so many resources, something will have to give
  - You don’t want to weaken the core foundation of understanding *why* certain programming paradigms are critical
What Can Academia Do?

- Include robustness in evaluation of programs, programming projects
- Create a “secure programming clinic”
  - Like an English clinic, or a writing clinic for law schools
- Provide supplementary material for textbooks, classes
  - These should emphasize robust programming
What Can Industry Do?

- Key is to do more than *say* it is important
- Make clear that the skills are important for hiring
  - Mention their need in job openings
  - Preference to those with skill in this also helps
Work With Students and Faculty

- Internships
  - Students *love* these; good recruiting tool
  - Tasks requiring robust programming emphasize its importance to students

- Help teach students
  - Review students’ code
  - Team with colleges in senior/capstone projects
What Will This Do?

- Increase student demand
  - If students see it as important, they will ask about it in class, evaluate programs, faculty in part on it

- Increase your visibility
  - Good recruiting tools
  - A corporate “good citizen”
What Has SANS Done?

• SANS has a great track record
  • Faculty workshop on exercises for secure programming (Apr. 2008)
  • Participation in summits, other forums to promote teaching secure programming, contributing ideas and enthusiasm
  • Teaching secure programming in its courses
What Can SANS Do?

- Convene collaborative strategic planning effort
  - Among industry, government, academic leaders
  - Identify specific goals; build on SESS results, ideas
  - Figure out how to move forward to implement teaching robust programming on a large scale
  - Need to set priorities, actions to realize them

- Must be done collaboratively
  - That way, the goals and actions are not tied to self-interest; they have more credibility and community support
What Can SANS Do?

- Agitate for targeted, sufficient resources to support the plans
  - Evaluation mechanism to gauge effectiveness
  - Example: if such a plan endorses robust and secure programming for both current students and practitioners, how will we best achieve these goals?
- More immediately: help link practitioners to programs, clinics that need them
  - Connecting people is the hardest part of getting support to academic programs
Government Support

- Act like an industry (see above)
- Government can also fund programs
- Imperative: target funding towards this specific purpose
  - That will require funding to be used for supporting robust programming
  - If done as adjunct, it is likely to disappear in the main purpose of the funding
NSF/DHS CAEIAE Program

- Begun in 1997 with 7 universities
- Office of the Assistant Secretary of Defense for Networks and Information Integration does oversight
- Academia funded via IASP (scholarships) and Capacity Building grants
- Program has very limited funding
Scholarships for Service supports graduate students

Capacity building funds to improve state of education in computer security

- Secure programming projects funded here are prototypes
- Focus on tool development, building awareness
- Funding not large
NSF Funding Sources

- Program funded in the Commerce/Justice/Science appropriations bill
- Could add extra funding targeted to secure programming projects specifically
- Focus for NSF alone should be on experimental projects to learn what works, what doesn’t, and under what conditions
Augment NSF’s budget to cover this

Otherwise, the state of computer security (and other research) will degrade, impairing our nation’s strengths in science and technology as well as other areas.
National Initiative for Cybersecurity Education

- Run by Dept. of Education and NSF
- Being transferred to NIST
- Mission:

  “bolster formal cybersecurity education programs encompassing kindergarten through 12th grade, higher education and vocational programs”
Component 2 includes developing a “cybersecurity capable workforce”

- Analogy between teaching cybersecurity and subjects like reading, writing, science, mathematics

- Analogy with writing good English seems appropriate

- Seems to be a natural program to drive this
Key Point

- Build on existing programs
- Understand that academia is a different environment—completely
  - Business models don’t work well because the “end product” is intangible
A Really, Really Important Point!

- Academic institutions are rarely governed hierarchically
  - Example: UC has an administration and an Academic Senate
  - The Senate, not the administration, has responsibility for the curriculum

Effect: no-one can order faculty to teach something in a particular way
Conclusion

- The state of practice can, and must, change
- Teaching robust programming, *and nothing more*, will not help
- The marketplace must also change, as must current practice
- The public will be the main driver
  - Unfortunately, probably through lawsuits . . .