Secure Programming Education
Matt Bishop

Contact Information
Matt Bishop
Department of Computer Science
University of California at Davis
1 Shields Ave.
Davis, CA 95616-8562

phone: (530) 752-8060
e-mail: bishop@cs.ucdavis.edu
www: http://seclab.cs.ucdavis.edu/~bishop
Problem Statement and Goals

- Few students write robust programs
  - Curriculum already crowded
  - Emphasis in most courses on getting programs working right
- How can we improve quality of programs that students write throughout undergraduate, graduate work?
  - In particular, how can we get students to think about security considerations?

“Secure” Programming

- Meaningless without definition of “security”
  - Some requirements implicit
- Notions usually implicit here
  - Robustness: paranoia, stupidity, dangerous implements, can’t happen here
  - Security: program does not add or delete privileges, information unless specifically required to do so
- Really, just aspects of software assurance
How to Do It, Approach 1

- Add security to exercises for general classes
  - Intro programming: integer or buffer overflow
  - Database: something on SQL injection
  - Programming languages: type clashes
  - Operating systems: race conditions
- Workshop held in April looked at ways to do this (thanks, SANS!)
  - Web site under development
  - Proposal for future workshop being developed

---

How to Do It, Approach 2

- Students must know how to write
  - Critical in all majors requiring communication, literary analysis skills
- Many don’t
  - Majors provide support for writing in classes (law, English, rhetoric, etc.)
- Does not add material to curriculum
  - Instructors focus on content, not mechanics
  - Provides reinforcement
Secure Programming Clinic

- Genesis: operating system class
  - TA deducted for poor programming style
  - Dramatic improvement in quality of code!
- Programming foundational in CS
  - Just like writing is in English (and, really, all majors …)
  - Clinicians assume students know some elements of style
  - Level of students affect what clinic teaches

How the Clinic Functions

- Assist students
  - Clinicians examine program, meet with student to give feedback
  - Clinic does not grade style
- Assist instructors
  - Clinic grades programs’ styles
  - Meet with students to explain grade, how the program should have been done
  - Class readers can focus on program correctness (as defined by assignment)

Interaction with students is critical to success
Some Experience

- Tested in computer security class
  - Class emphasizes robust, secure programming
- Setup for class
  - Class had to analyze small program for security problems
  - Class applied Fortify code analysis tool to larger program, and traced attack paths
    - Thanks to Fortify for giving us access to the tool!

How It Worked

- Write program to check attributes of file; if correct, change ownership, permissions
  - If done wrong, leads to TOCTTOU flaw
- Students had to get program checked at clinic before submitting it
  - Students sent program to clinician first
  - Clinician reviewed program before meeting with student
  - Student then could modify program
Results

<table>
<thead>
<tr>
<th>Programming Problem</th>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOCTTOU race condition</td>
<td>100%</td>
<td>12%</td>
</tr>
<tr>
<td>Unsafe calls (strcpy, strcat, etc.)</td>
<td>53%</td>
<td>12%</td>
</tr>
<tr>
<td>Format string vulnerability</td>
<td>18%</td>
<td>0%</td>
</tr>
<tr>
<td>Unnecessary code</td>
<td>59%</td>
<td>53%</td>
</tr>
<tr>
<td>Failure to zero out password</td>
<td>70%</td>
<td>0%</td>
</tr>
<tr>
<td>No sanity checking on modification time</td>
<td>82%</td>
<td>35%</td>
</tr>
<tr>
<td>Poor style</td>
<td>41%</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Notes

- Unsafe function calls
  - 4 did not set last byte of target to NUL
- Unnecessary code
  - 2: unnecessary checking; 7: errors or unnecessary system calls
- Zero out password
  - 2 did so at end of program
- Sanity checking (*not* pointed out to all)
  - 4 found it despite no mention
- Style greatly cleaned up
Observations

• Students required to participate upon pain of not having program graded
  ◦ Probably too harsh; 7/24 did not do program
• Clinician not TA
  ◦ Students seemed to prefer this
  ◦ In general, students unfamiliar with robust, secure programming before class
• Clinic uses handouts for other classes

Further Work Needed

• Need to do this for more classes
• Need more helpful material, especially for beginning students
• If successful, can help improve state of programming without impacting material taught in computer science classes
Project Goals

- Extend web pages to provide students help in creating good programs
  - Many out there, but typically at too advanced a level for beginning programming students
- Try clinic in non-security, advanced classes
  - In 2006, also tried for 1 program in second programming course; results good
  - Need more experience to figure out what the best way to run this clinic is

References