A Clinic to Teach Good Programming Practices

Matt Bishop
B. J. Orvis

UC Davis
Computer Security Laboratory

Contact Us

Matt Bishop
mabishop@ucdavis.edu

B. J. Orvis
wsorvis@ucdavis.edu

Department of Computer Science
University of California at Davis
One Shields Ave.
Davis, CA 95616-8562

Problem Statement

- Few students write well-written programs
  - Curriculum already crowded
  - Emphasis in *most* courses on getting programs working right
- Key question: ***how can we improve quality of programs that students write throughout undergraduate, graduate work?***

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

Writing Clinics

- 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 writing
- 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 in English (and, really, all majors …)
  - Clinicians assume students know some elements of style
  - Level of students affect what clinic teaches

How 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
- Readers can focus on program correctness

Analysis

- Assist students
  - Strictly adjunct to existing classes
  - Instructor has to incorporate use of clinic into deadlines, assignments, grading
- Assist instructors
  - Students ignore feedback, get lower grade
  - Clinicians must take different instructor grading styles into account
- ***Interaction with students critical***

Experience

- Tested in computer security class at UC Davis
  - 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

The Program

- 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

Initial Problems

programming problem	***before***	***after***
TOCTTOU race condition	100%	12%
Unsafe calls (strcpy, strcat, etc.)	53%	12%
Format string vulnerability	18%	0%
Unnecessary code	59%	53%
Failure to zero out password	70%	0%
No sanity checking on mod time	82%	35%
Poor style	41%	N/A

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

Conclusion

- Need to do this for more classes
  - Spring: doing it for ECS 40, second course in programming
- Use educational metrics to evaluate success
  - And to figure out how to make clinic more effective
- If successful, can help improve state of programming *without* impacting material taught in computer science classes