Teaching Assurance Using Checklists

Matt Bishop

Dept. of Computer Science

University of California, Davis

Outline

- Security and assurance
- Why teach assurance?
- Types of teaching
- Using checklists
- Thoughts on secure programming
- Conclusion

Opening Thought

Overconfidence breeds error when we take for granted that the game will continue on its normal course; when we fail to provide for an unusually powerful resource—a check, a sacrifice, a stalemate. Afterwards the victim may wail, “But who could have dreamt of such an idiotic-looking move?”

—Fred Reinfeld, *The Complete Chess Course*

Security and Assurance

- Assurance: confidence that an entity meets its requirements
- Security: policy stating what is, and is not, allowed
- Security assurance: confidence that an entity correctly implements the given security policy

Why Teach Assurance

- Easy answer: to improve the state of software, hardware, operations, etc.
- Another answer: to teach how to analyze a problem and test proposed solutions to find the most appropriate one for the particular situation
  - Difference between being able to do arithmetic and higher mathematics

Importance

- Easy answer provides immediate solutions
  - Tailored for a particular problem
  - Using available technologies
- Other answer promotes long-term solutions
  - Tailored for a class of problems
  - Develop technologies to solve problems

Key Question

- Do we want to solve existing problems, or anticipate future problems and devise technologies to handle them *before* the problems become widespread?
  - Not either-or!

Types of Teaching

- Training
  - Focus is on steps needed to secure entities
- Academic education
  - Focus is on understanding principles and how to apply them to situations
  - Focus is on discovering principles, developing methodologies

Difference

Education is what survives when what has been learnt has been forgotten.

—B. F. Skinner

Checklists

- First axis: guidance or specific?
  - Guidance checklists prompt memory, and require user to understand when to follow, and ignore, items in list
  - Specific checklists list items to be performed, and require user to perform items in list

Checklists

- Second axis: use for doing or auditing?
  - Doing checklists list items to be done, and the user must perform them
  - Auditing checklists list items to be checked, but the auditor need not do the items; she must check they are done

Applying This

- Training: specific checklists usually more appropriate than guidance checklists
- List of steps to perform
- Assume a particular environment
- Often assume users have common basis
  - Understand the steps in the checklist
  - Understand any ancillary, but omitted, information needed to apply the steps

Applying This

- Academic education: guidance checklists usually more appropriate than specific checklists
- List of principles, ideas as a memory prompt
  - No substitute for understanding!
- General enough for most environments
  - Students learn how to apply ideas, in effect specializing the generic checklist

Deriving Academic Checklists

- Determine goal
- Apply any particular constraints, requirements
  - Arise from assumptions, intended environment
- Derive set of principles, other that students should know
  - Principles already known; this is a taxonomy or codification of them for pedagogic purposes
- If appropriate, apply these to a particular domain
  - May lead to a specific checklist

Example: “CBK”

- Common Body of Knowledge
  - Information everyone is expected to know
    - Finesse the question of who “everyone” is …
  - Essentially, a checklist of topics
    - Topics must be organized cohesively
      - Must be an easily understood design for presenting them in the manner that the CBK does
    - Each topic’s inclusion must be justified
      - Too much information out there
    - Each topic must be placed in historical context
      - Sometimes yesterday’s key technology is today’s dinosaur
      - Sometimes old ideas have new applications

Importance of History

- Ideas and principles are important
  - Must understand the context in which they arise
  - Present frameworks in which one can test new ideas, solutions
- Technology may be less so
  - Good for understanding how ideas were applied
  - Useful for testing new principles, frameworks

Example: Reference Monitor

- All accesses of resource must go through this
  - Bad: coding making it difficult to verify this
  - Good: clean design, coding
- Need to validate mechanism
  - Bad: more complex than needed
  - Good: simple, modular, clean design and coding
- Need to protect its integrity
  - Bad: monitor or associated data can be accessed, altered
  - Good: considerable error checking to detect this

Example: CIS Checklists

- Detailed instructions for securing systems
  - *Intended for practitioners (CIS says this explicitly)*
- Assumptions about environment
  - CIS doesn't state these overall; some items ask user to determine if the step is appropriate
- Very low level
  - "Do you need to run telnet," not "do you want to allow cleartext passwords over the net" or "do you want to allow net access to specific hosts"

Using These in a Class

- Academic education focuses on broader concepts
- Security policy exercises:
  - Devise a system that is secure but violates some of the items in the checklist
  - Devise a system that is not secure but does not violate any item in the checklist
- Analysis exercise:
  - Determine the assumptions underlying the recommendations

Example: CBK for Secure Software

- Out "for review and comment only"
  - Disseminate best practices, methods to encourage security in software code development
  - Being developed by DoD and DHS working group, including outsiders
- Audience includes educators, trainers
  - One goal: CBK to help identify material for curriculum, references for that material

Strong Points

- Some discussion of environment
- Some discussion of principles
- Lots of topics in several areas:
  - Secure software requirements, design, construction, validation
  - Tools and methods
  - Processes, management, operations
  - Acquisition

Weak Points

- Little integration of principles into the development of elements of the CBK
  - Section on principles mentions Saltzer's & Schroeder's, discusses some aspects of security assurance and crypto
  - Does not derive steps, information from principles, or show how principles lead to information in CBK
  - Reads like results of a brainstorming session (that's how it was done, to some extent)

Weak Points

- Little history or historical context
  - One paragraph on history
  - Other places eschew original sources for more modern ones
    - Example: reference monitor cites Bishop (2003), not Anderson (1972)
  - Ideas, technologies, methodologies lack much of the context in which they were developed

Thoughts About Use

- Guidance, not specificity
- Training: provides general guidance but not detailed information
  - Does discuss requirements analysis
- Academic education: not structured enough or comprehensive enough to guide curriculum development

Secure Programming

- Style of programming intended to make the program more secure
  - Secure defined in terms of security policy
- Two parts:
  - Security related to the particular problem
  - Security related to generic problems

Particular Problem

- Implement a web server that restricts access to a particular set of people
  - Adequate identification
  - Adequate authentication
  - What access is appropriate for each individual

Generic Problems

- Implement a web server that restricts access to a particular set of people
  - Buffer overflows leading to unauthorized access
  - Hijacking connections leading to unauthorized user taking over a legitimate session after it has been established
  - Race condition allowing a one-time password authentication scheme to accept the same password twice

Focus

- “Secure programming” usually refers to the second
  - Quality of code focuses on buffer overflows, race conditions, type clashes, etc.
- Foreshadowing: checklists typically emphasize this
  - But you need to consider the first, too!
  - * CBK mentioned earlier does so

Conclusion

- Checklists need to be derived in a structured manner, from principles and/or assumptions
- Type of checklist used in education depends upon the goals of education
  - Academic education is *very* different than non-academic education

Final Thought: Clear Overall Goals

Gentlemen,

Whilst marching from Portugal to a position which commands the approach to Madrid and the French forces, my officers have been diligently complying with your requests which have been sent by H.M. ship from London to Lisbon and thence by dispatch to our headquarters.

We have enumerated our saddles, bridles, tents and tent poles, and all manner of sundry items for which His Majesty's Government holds me accountable. I have dispatched reports on the character, wit, and spleen of every officer. Each item and every farthing has been accounted for, with two regrettable exceptions for which I beg your indulgence.

Unfortunately the sum of one shilling and ninepence remains unaccounted for in one infantry battalion's petty cash and there has been a hideous confusion as to the number of jars of raspberry jam issued to one cavalry regiment during a sandstorm in western Spain. This reprehensible carelessness may be related to the pressure of circumstance, since we are war with France, a fact which may come as a bit of a surprise to you gentlemen in Whitehall.

This brings me to my present purpose, which is to request elucidation of my instructions from His Majesty's Government so that I may better understand why I am dragging an army over these barren plains. I construe that perforce it must be one of two alternative duties, as given below. I shall pursue either one with the best of my ability, but I cannot do both:

1. To train an army of uniformed British clerks in Spain for the benefit of the accountants and copy-boys in London or perchance:

2. To see to it that the forces of Napoleon are driven out of Spain.

—Duke of Wellington, to the British Foreign Office, London, 1812