Software Engineering Lifecycle

Authors: Jan G. Hogle, Susan Gerhart

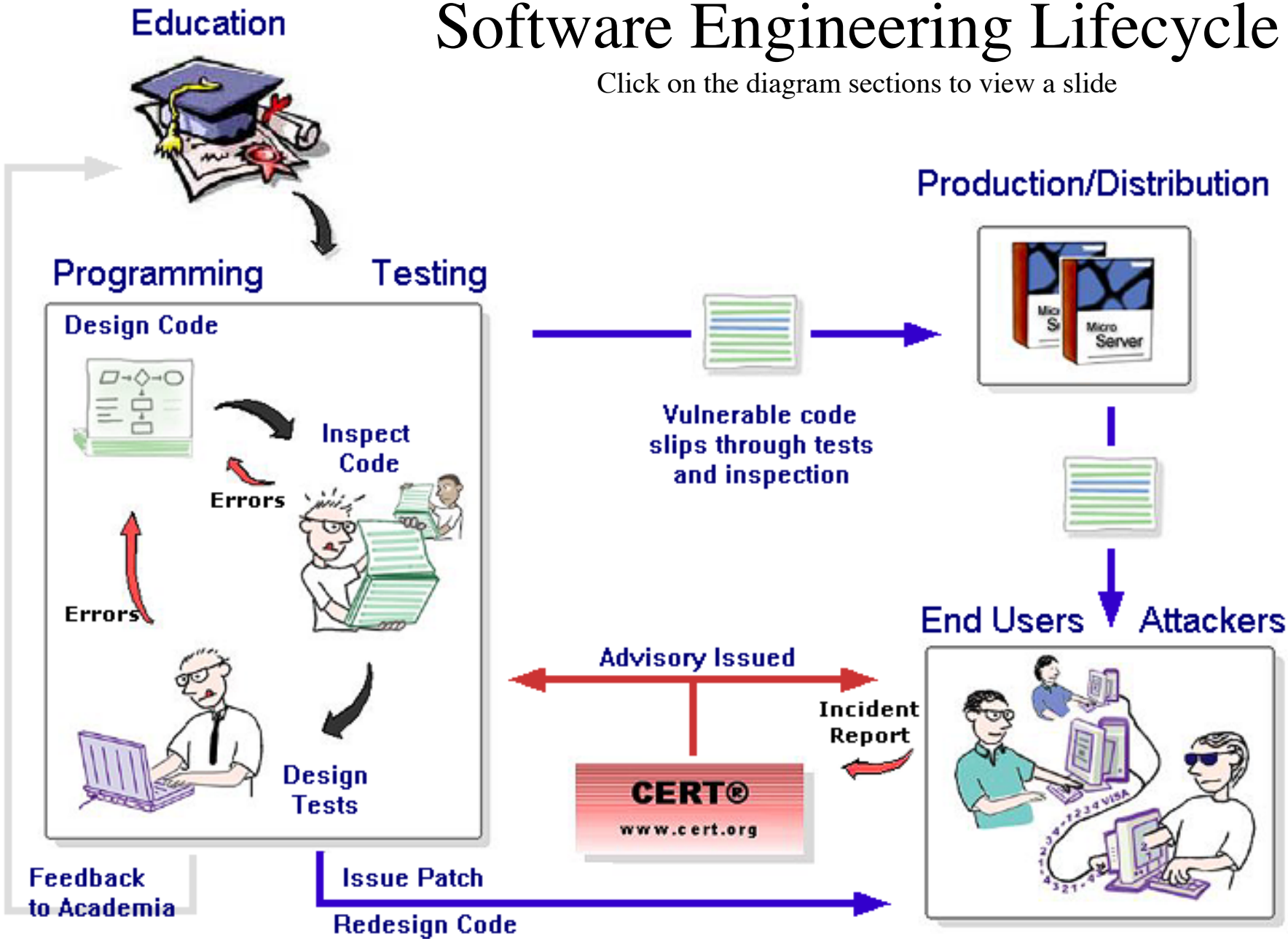
This Document was Funded by the National Science Foundation
Federal Cyber Service Scholarship For Service Program:
Grant No. 0113627

Distributed July 2002

Embry-Riddle Aeronautical University • Prescott, Arizona • USA

Software Engineering Lifecycle

Click on the diagram sections to view a slide



Academia produces students who:

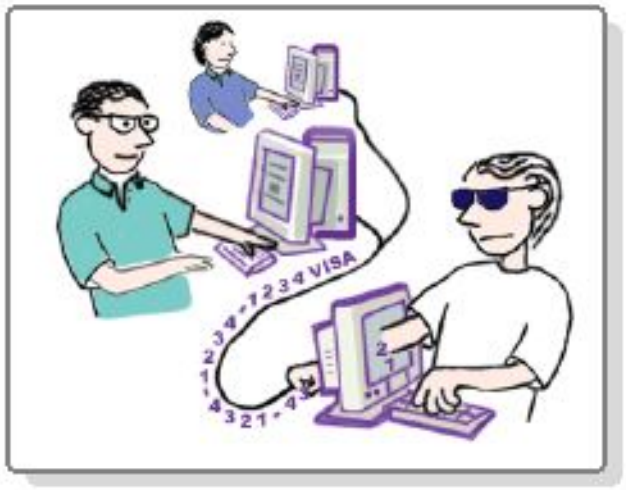
- Aren't tuned into the dangers of buffer overflows
- Can't recognize a buffer overflow vulnerability when they see it, so they make the mistake in coding
- Are careless in their coding as well as inspection and testing tasks
- Are not made aware of buffer overflows by instructors or textbooks

Managers, Developers and QA specialists iterate through cycles of detailed design and coding but...

- Employ poor coding and quality skills learned in school
- Are often forced to use low-level languages like C
- Use established programming techniques that are highly error-prone
- Fail to incorporate inspection and design techniques known to prevent and discover buffer overflow
- Run levels of code they can't control but which are riddled with buffer overflows

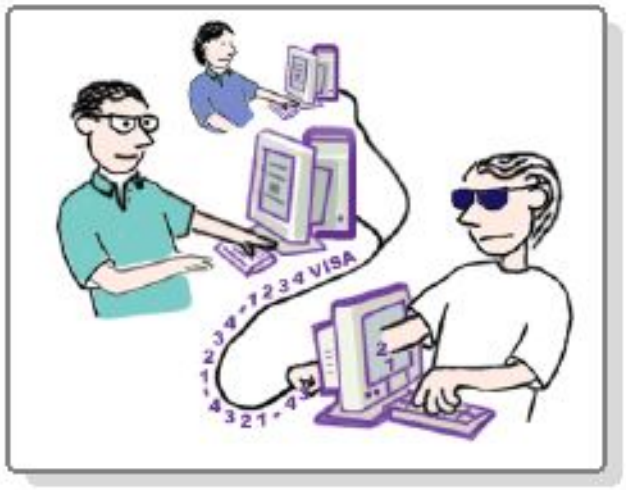
Buffer Overflow Vulnerabilities not detected during development and QA get into products

- Vulnerable code slips through tests and inspection
- New products expose buffer overflows in old code from libraries and other vendors
- Proper use of products to avoid buffer overflows isn't known or documented



An End User may find a Buffer Overflow unintentionally or may search for it

- An ordinary user may observe unusual activity or symptoms of buffer overflow
- Security shops like Eeye and university groups search for vulnerabilities by playing the role of attackers on new and old products

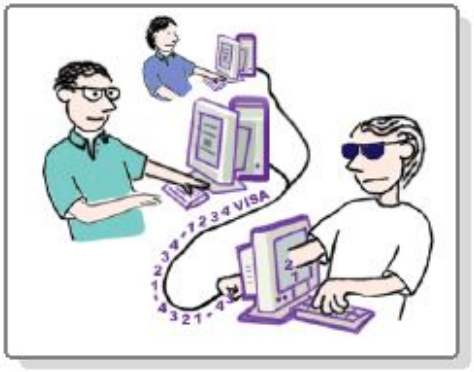


An Attacker finds a way to force a buffer overflow to meet their purposes

- Attackers know common vulnerabilities of vendors and their products
- Attackers learn from the web and from each other how to make buffer overflows occur
- Attackers acquire ways to make buffer overflows lead to hijacking a system or planting seeds for future attacks

When product users find a buffer overflow and alert authorities, a flurry of patching occurs:

- An alert goes to the vendor and official sites like cert.org
- A confirmation, analysis, and explanation goes out to vendors and users as an advisory

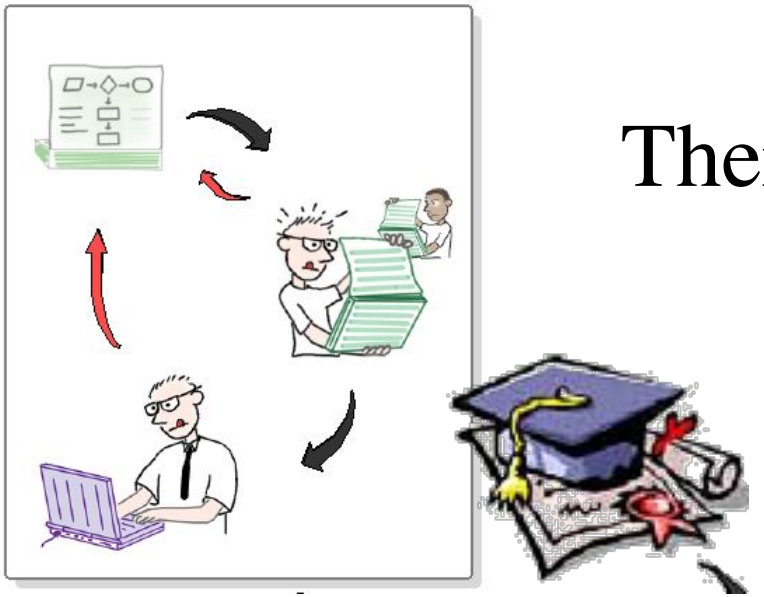


The developer reaction team, security shop, and authorities issue a patch:

- Users of the product must install the patch to protect themselves and others
- Vendors issue multiple patches, often daily

The development organization responds to the buffer overflow vulnerability by:

- ✓ Fixing the underlying code problem in its later versions
- ✓ Replacing patches with corrected code
- ✓ Improving develpment processes and tools to avoid similar buffer overflows



There is no feedback to academia.

If there were, it could:

- ✓ Make the pipeline of students more sensitive to buffer overflows
- ✓ Improve education and training materials - books, exercises, tools
- ✓ Encourage authors and instructors to raise the visibility of the buffer overflow problem
- ✓ Incorporate economic lessons of publicity and cost analyses from journalists and industry analysts

About this Project

This presentation is part of a larger package of materials on buffer overflow vulnerabilities, defenses, and software practices. For more information, go to: http://nsfsecurity.pr.erau.edu

Also available are:

- Demonstrations of how buffer overflows occur (Java applets)
- PowerPoint lecture-style presentations on an introduction to buffer overflows, preventing buffer overflows (for C programmers), and a case study of Code Red
- Checklists and Points to Remember for C Programmers
- An interactive module and quiz set with alternative paths for journalists/analysts and IT managers as well as programmers and testers
- A scavenger hunt on implications of the buffer overflow vulnerability

Please complete a feedback form at http://nsfsecurity.pr.erau.edu/feedback.html to tell us how you used this material and to offer suggestions for improvements.