Panel 2: Software Assurance Education

Samuel T. Redwine, Jr., Chair  
James Madison University  
redwinst@jmu.edu

Joseph Saur  
Georgia Tech Research Institute  
joseph.saur@jsic.jfcom.mil

Hun Kim  
Department of Homeland Security  
hun.kim@dhs.gov

Shawn A. Butler  
Carnegie Mellon University  
shawnb@cs.cmu.edu

Abstract

Software engineering processes for building safe and secure software have existed for a long time. However, these processes—particularly for secure software engineering—have not been widely taught within colleges and universities leading to a shortage of graduates skilled in these areas. This panel will discuss the increasing need for colleges and universities to produce graduates that are skilled in building safe and secure software. Panelists will share their experiences teaching courses in these areas and future directions for curricula.

Panel outline

In each of the last two years, the CERT Coordination Center has reported roughly 4000 vulnerabilities, demonstrating that software security is a significant problem. Exploitation of these vulnerabilities continues to grow exponentially, with well over a hundred thousand incidents reported to the CERT CC in 2003. The problems with software are not just in operating systems and web servers; a large portion of the new vulnerabilities now being reported are within applications.

For more than twenty years, much has been known about safe and secure software development that builds on the necessary foundation of well organized, managed and disciplined software engineering processes. Distributed systems and the Internet have complicated systems design, but methods for producing dependable software remain largely unchanged. Understanding how to develop secure software is a necessity since software produced by developers often interacts with users via the Internet.

Among the questions the panel will consider are:

• Is there a difference in the methods used to produce safe or secure systems? How are the methods the same or different? Can they be effectively taught in the same course?
• What types of class projects that will help students understand how to build safe or secure software?
• What are the roles of formal methods in producing secure and safe software?
• Should changes be made in beginning courses to introduce the proper mindset and techniques as well as avoiding students developing “bad” habits?
• What lessons can be learned from the Safety Argument and documentation processes that can be used in developing secure software?
• Are design vulnerabilities or code vulnerabilities more fundamental?
• What are the roles of threat and hazard analyses in system requirements determination?
• Why are security and safety emergent system properties not simply specialized security functionality?

Panel participants have experience with both undergraduate and graduate software engineering education. The panel has been organized to include perspectives from both the academic community and those who depend on its graduates.