Software Security Engineering
Lecture 1

Nancy R. Mead, SEI
nrm@sei.cmu.edu

© 2012 Carnegie Mellon University

Outline

About this course

Software assurance challenges

Foundations for software assurance

Software assurance guiding principles

© 2012 Carnegie Mellon University

Course topics

- Security models and methods in the areas of:
  - lifecycle process models
  - risk management
  - requirements engineering
  - architecture and design
  - coding and testing
  - governance and management
- If time permits, acquisition of newly developed and COTS software will also be discussed.

© 2012 Carnegie Mellon University

Prerequisites

- Undergraduate software engineering course
- Undergraduate information security course
- Equivalent background

- Note: The course will tend to assume that students have software engineering background, such as knowledge of common lifecycle models

© 2012 Carnegie Mellon University

Educational Activities

- Class will be lecture and discussion, with guest lectures on some topics
- Readings from textbook, papers, reports
- Homework assignments
- Project including selected software development activities:
  - Lifecycle security management plan
  - Selection of process model (Agile, Spiral, etc.) and rationale
  - Security risk analysis
  - Development of misuse cases/attack trees
  - Security requirements elicitation
  - Architectural Trade-off analysis/QAW
  - Design of security features (e.g. Access control mechanisms)
  - Inspection

© 2012 Carnegie Mellon University

Text and other sources

- Allen, Julia H., Barnum, Sean, Ellison, Robert J., McGraw, Gary, & Mead, Nancy R. *Software Security Engineering: A Guide for Project Managers*. Addison Wesley Professional, 2008. (Available from Addison-Wesley and Amazon.com)

- U.S. Department of Homeland Security. *Build Security In Website*

- Additional papers, SEI reports, CERT podcasts, webinars, etc. as needed

© 2012 Carnegie Mellon University

Grading Criteria

- 50% individual assignments
- 50% team project

- Grading will take into consideration completeness, creativity, deep insights, thinking outside the box. Sources must be cited. Material lifted from another source must be in quotes.

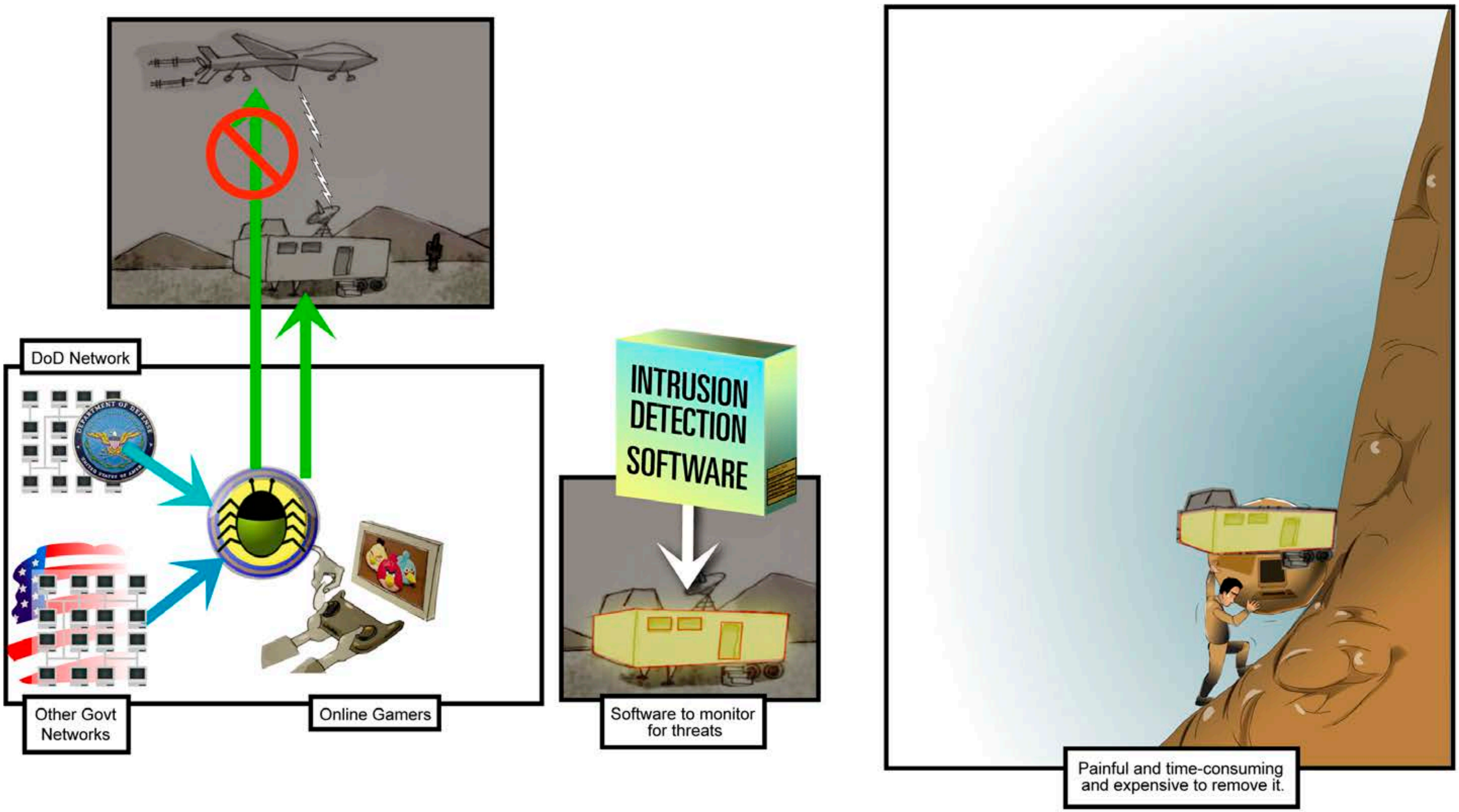
- Assignments are to be turned in or posted to Blackboard BEFORE class on the day they are due. Assignments not turned in on time will lose 10% for each day late.

© 2012 Carnegie Mellon University

Software Assurance Challenges

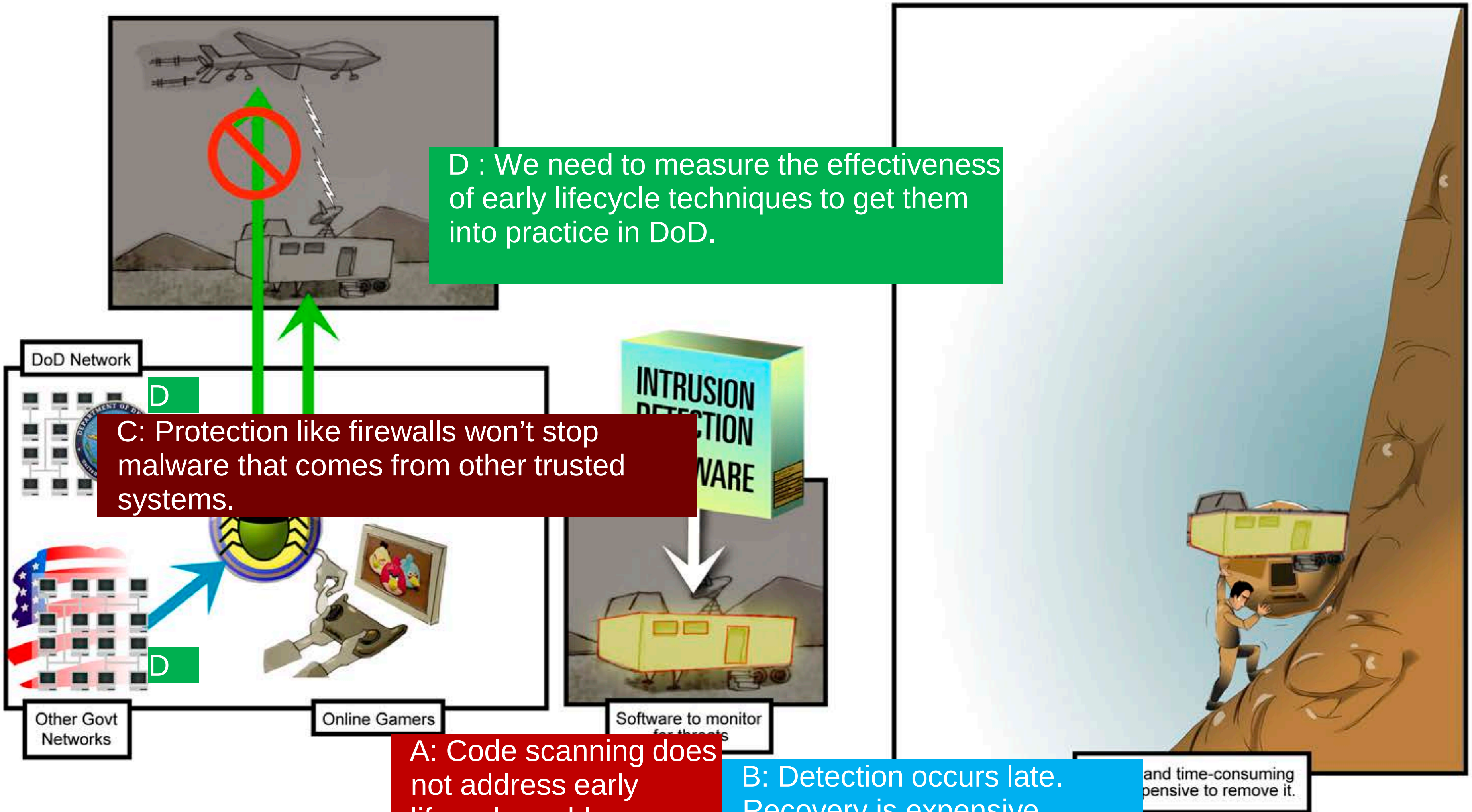
© 2012 Carnegie Mellon University

Scenario – Drone Virus Attack



© 2012 Carnegie Mellon University

Drone Scenario – Key Challenges



D : We need to measure the effectiveness of early lifecycle techniques to get them into practice in DoD.

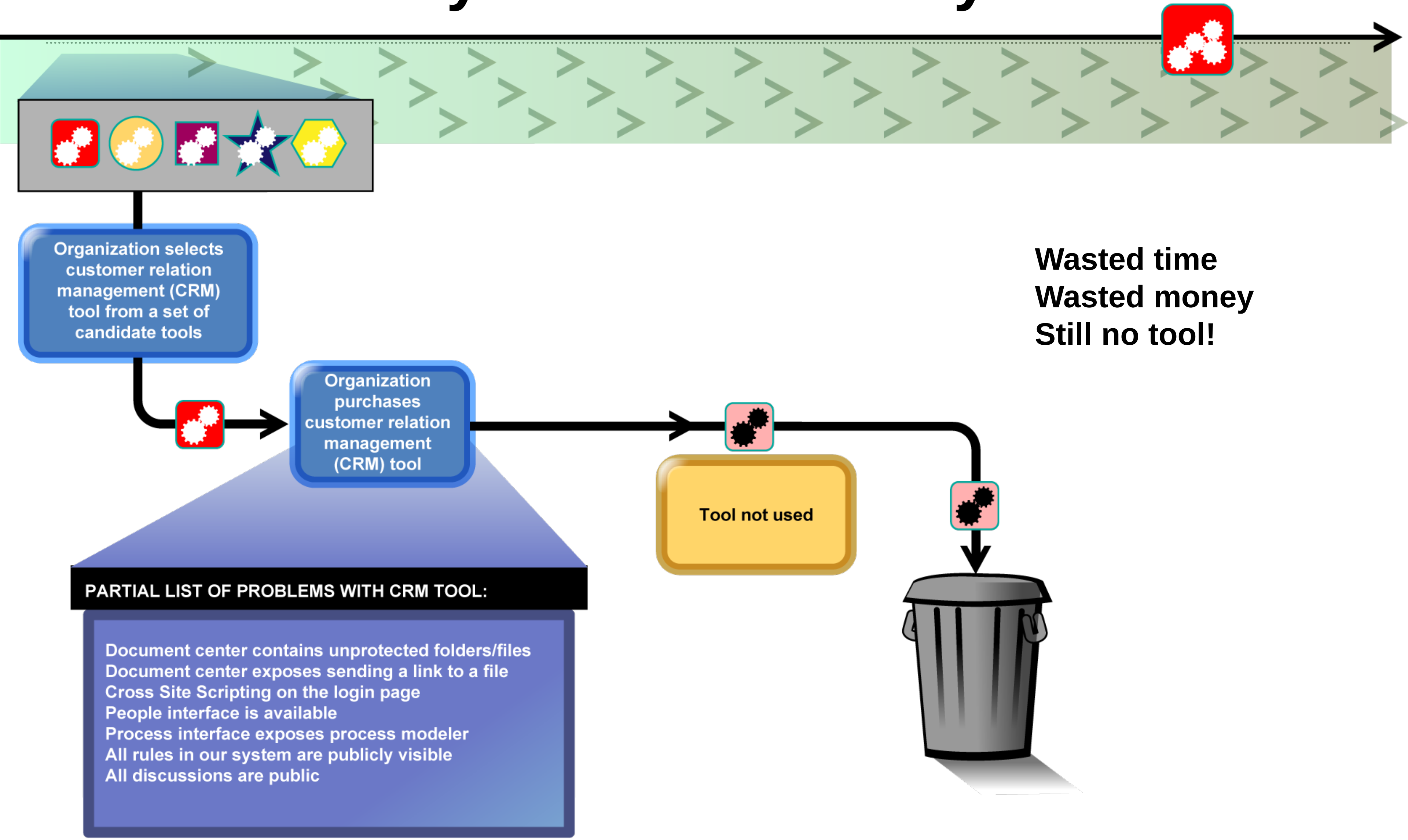
C: Protection like firewalls won’t stop malware that comes from other trusted systems.

A: Code scanning does not address early lifecycle problems

B: Detection occurs late. Recovery is expensive.

© 2012 Carnegie Mellon University

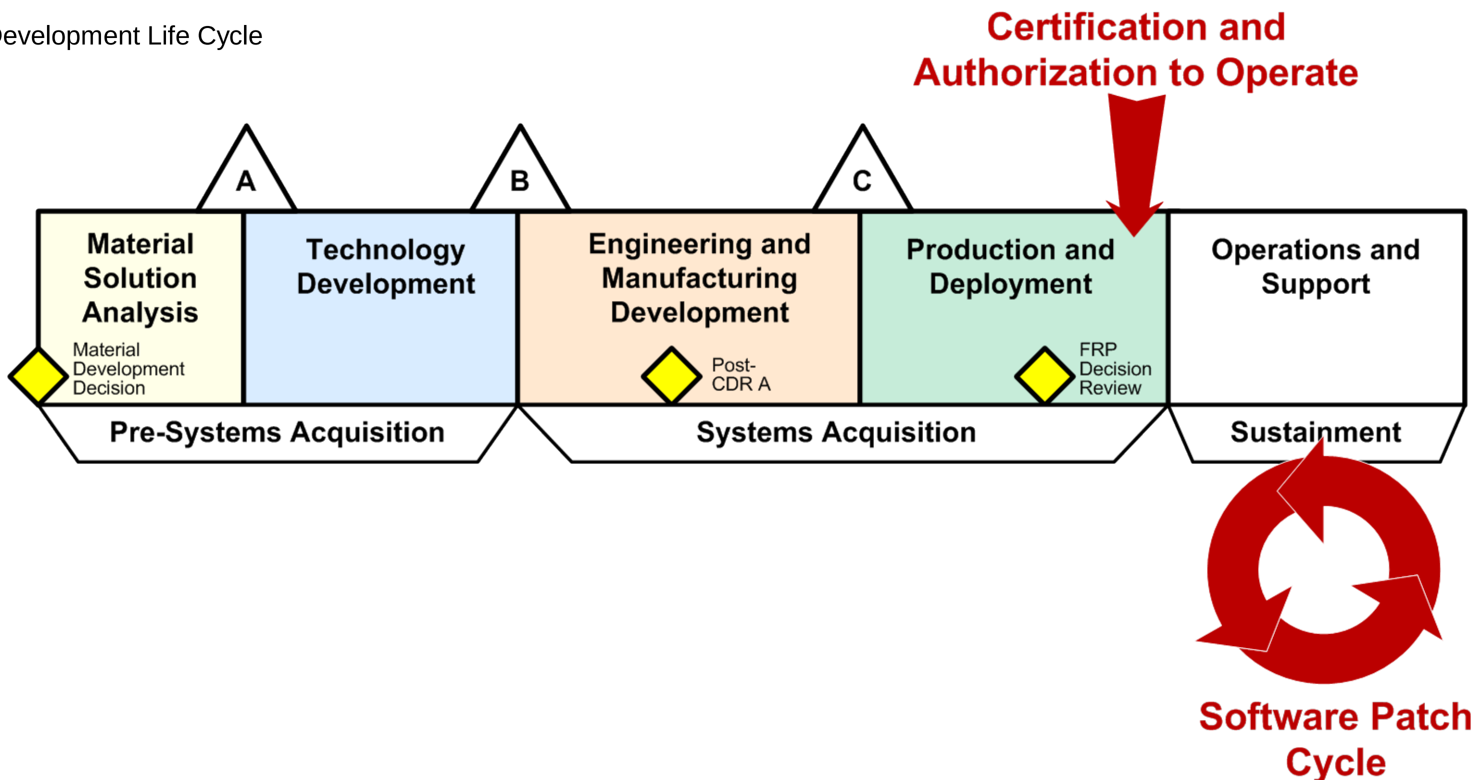
Is There Really a COTS Security Problem?



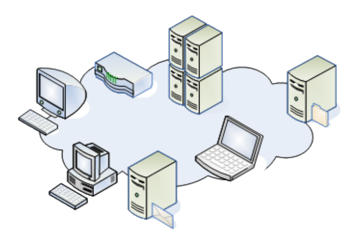
© 2012 Carnegie Mellon University

Current Challenge for Software Assurance

Development Life Cycle



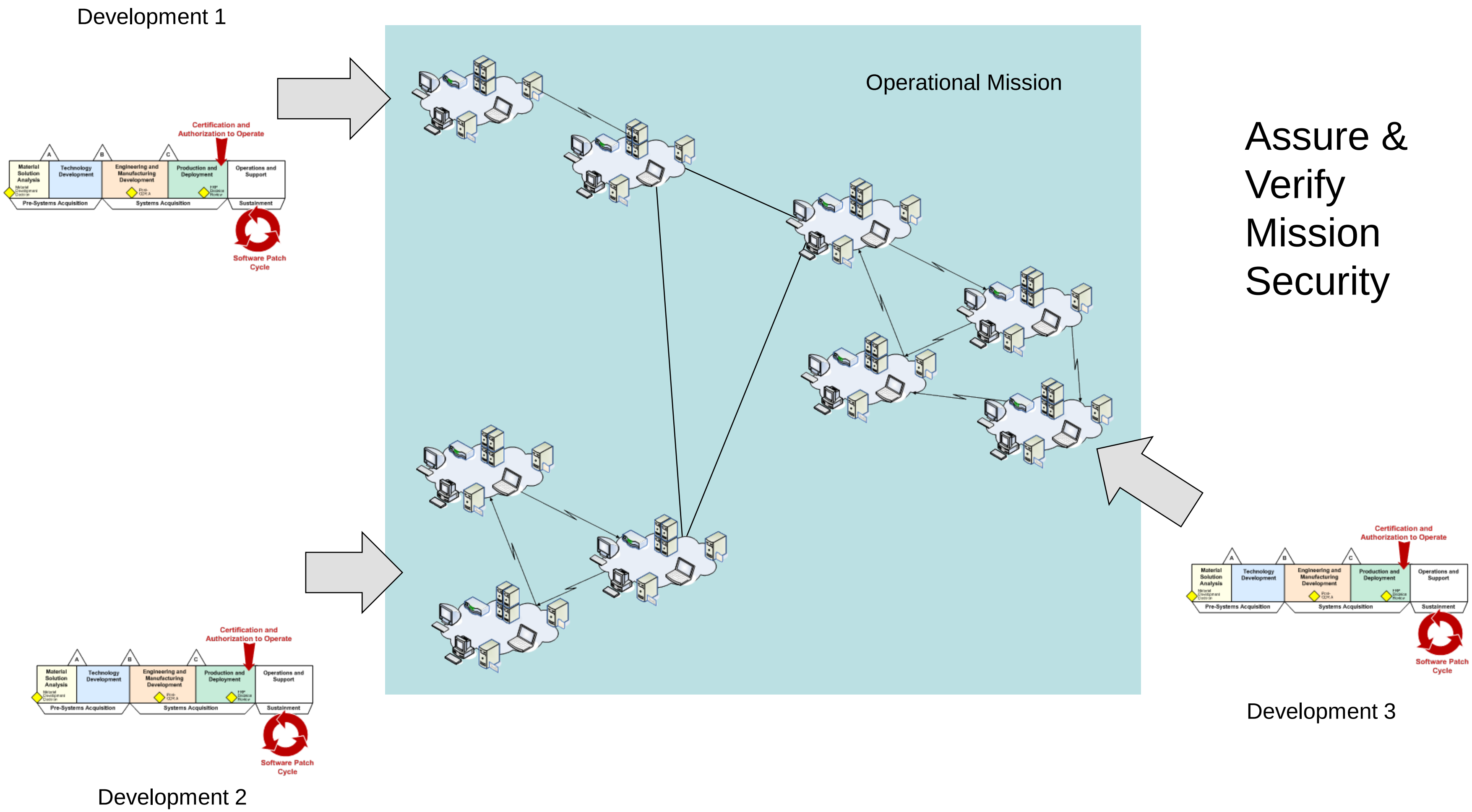
Patch & Pray



47,202 known vulnerabilities as of 9/17/11

© 2012 Carnegie Mellon University

Operational Mission Reality – Systems of Systems



Assure & Verify Mission Security

Foundations for Software Assurance

© 2012 Carnegie Mellon University

Information/IT Security Point of View

- Typically dealing with an organization’s infrastructure provider, their management chain, & the CIO
- End objective is to provide a functional, available, secure operational infrastructure & applications for all users
- Information protection & privacy are demanding increasing attention (regulatory, marketplace pressure)
- Software/application security may or may not be on the radar screen

© 2012 Carnegie Mellon University

Software Security Point of View

- Dealing primarily with software/application developers & their management chain
    - in-house, service provider, purchased software
- End objective is to produce working systems & applications, on schedule, on budget
- Security typically addressed (if at all):
    - During coding and testing
    - During operations/production as an “after the fact” add-on; reactive
    - For COTS, open source, or third party software, as a provider/vendor responsibility

COTS: Commercial Off The Shelf

© 2012 Carnegie Mellon University

Why Software Security? - 1

- Developed nations' economies and defense depend, in large part, on the reliable execution of software
- Software is ubiquitous, affecting all aspects of our personal and professional lives.
- Software vulnerabilities are equally ubiquitous, jeopardizing:
  - Personal identities
  - Intellectual property
  - Consumer trust
  - Business services, operations, & continuity
  - Critical infrastructures & government

© 2012 Carnegie Mellon University

Why Software Security? - 2

- Most successful attacks result from:
  - Targeting and exploiting known, non-patched software vulnerabilities
  - Insecure software configurations
- Many of these are introduced during software design & development
- Increasing trend of assembling systems from purchased parts means getting software acquisition* right with respect to security

- Refer to Polydys & Wisseman. "Software Assurance in Acquisition: Mitigating Risks to the Enterprise." 2007. https://buildsecurityin.us-cert.gov/daisy/bsi/resources/dhs/908.html?branch=1&language=1

© 2012 Carnegie Mellon University

So What Is Software Security?

- Not the same as security software
  - Firewalls, intrusion detection, encryption
  - Protecting the environment within which the software operates
- Engineering software so that it continues to function under attack
- The ability of software to recognize, resist, tolerate, and recover from events that threaten it

- The goal: Better, defect-free software that can function more robustly in its operational production environment

© 2012 Carnegie Mellon University

Security Perspectives

https://www.securecoding.cert.org/confluence/display/seccode/Top+10+Secure+Coding+Practices

© 2012 Carnegie Mellon University

Software Needs to be Trusted

- Exploitation of software defects is estimated to cost the U.S. economy $60 Billion annually
- Software development and sustainment activities must follow proper practices, but there is no authoritative point of reference
- In 2005, U.S. Dept of Homeland Security (DHS) created a group to define a common body of knowledge (CBK) for secure software assurance

© 2012 Carnegie Mellon University

Definition: Software Assurance

- Software assurance (Software Assurance Curriculum Project)

    Application of technologies and processes to achieve a required level of **confidence** that software systems and services **function in the intended manner**, are free from accidental or intentional vulnerabilities, provide security capabilities appropriate to the threat environment, and recover from intrusions and failures.

© 2012 Carnegie Mellon University

Goal of the CBK

- Serve as a basis for
- "defining workforce needs and competencies, leveraging sound practices, and guiding curriculum development for education and training relevant to software assurance"
- Reference: Redwine, S., Software Assurance: A Guide to the Common Body of Knowledge to Produce, Acquire and Sustain Secure Software V1.1, https://buildsecurityin.us-cert.gov/bsi/dhs/927-BSI.html

© 2012 Carnegie Mellon University

Strengths of the CBK

- Provides help for the U.S. government to ensure that it is getting secure software
- Provides 300 pages of recommendations for *what* practices are needed

© 2012 Carnegie Mellon University

Limitations of the CBK

- Missing:
  - Information about ***why*** the practices are required
  - Guidance as to ***how*** the practices should be applied to a range of situations

© 2012 Carnegie Mellon University

Addressing the Gaps

- DHS enlisted the SEI CERT program to coordinate the development of a curriculum for a Master of Software Assurance (MSwA) degree program (***what*** and ***how***)
  - Built on the CBK and other sources to develop a curriculum body of knowledge and associated outcomes
  - Identified the need for a coherent set of guiding principles for secure software assurance
- SEI CERT and the Software Engineering Program at Oxford University, UK collaborated to build a set of principles (***why***)

© 2012 Carnegie Mellon University

Security Principles

- Saltzer and Schroeder* defined security as “techniques that control who may use or modify the computer or the information contained in it”
- Described the three main categories of concern:
    - Confidentiality
    - Integrity
    - Availability

* Reference: Saltzer and Schroeder, “The Protection of Information in Computer Systems.” *Communications of the ACM*, 1974.

© 2012 Carnegie Mellon University

Technology Environment in 1974

- S360 in use from 1964-1978
- S370 came on the market in 1972
- COBOL & BAL programming languages
- MVS operating system released in March 1974
- Patches were carefully tested to minimize operational disruption

© 2012 Carnegie Mellon University

Changes since 1974

- Internet
- Morris worm – November 2, 1988
- 50,000+ software vulnerabilities and exposures (CVE)
- Java, C++, C#
- Mobile computing
- Cloud
- Etc.

© 2012 Carnegie Mellon University

Software Assurance Guiding Principles

© 2012 Carnegie Mellon University

Principles of Software Assurance

- A set of principles to guide learners in understanding the WHY of software assurance

© 2012 Carnegie Mellon University

Principle 1: Risk

- Perception of risk drives assurance decisions
  - Assurance implementation choices (policies, practices, tools, restrictions) are based on the perception of threat and the impact should that threat be realized
  - Perceptions are built based on successful attacks – the current state of assurance is largely reactive – more successful organizations react and recover faster, learn from the reactive responses or others, and are more vigilant in anticipating and detecting attacks
  - Misperceptions are failure to recognize threats and impacts – "how could it happen to us?"
  - Risk decisions must be shared among all stakeholders and technology participants to ensure a consistent and effective implementation

© 2012 Carnegie Mellon University

Principle 2: Interactions

- Highly connected systems (e.g. Internet) require alignment of risk across all stakeholders otherwise critical threats will be unaddressed (missed, ignored) at different points in the interactions
  - There are costs to addressing assurance which must be balanced against the impact of the risk
  - Risk must also be balanced with other opportunities (performance, reliability, usability, etc.)
  - Interactions occur at many technology levels (network, security appliances, architecture, applications, data storage, etc.) and are supported by a wide range of roles – effective assurance requires consist risk recognition and response at all levels

© 2012 Carnegie Mellon University

Principle 3: Trusted Dependencies

- Your assurance depends on other people's assurance decisions and the level of trust you place on these dependencies (system of system problem based on interactions)
  - Each dependency represents a risk
  - Dependency decisions should be based on a realistic assessment of the threats, impacts, and opportunities represented by an interaction
  - Dependencies are not static and trust relationships should be reviewed to identify changes that warrant reconsideration
  - Using many standardized pieces to build technology applications and infrastructure increases the dependency on other's assurance decisions

© 2012 Carnegie Mellon University

Principle 4: Attacker

- There exists a broad community of attackers with growing technology capabilities able to compromise the confidentiality, integrity, and availability of any and all of your technology assets - there are no perfect protections and the attacker profile is constantly changing.
  - The attacker uses technology, processes, standards, and practices to craft a compromise (socio-technical responses).
  - Attacks are crafted to take advantage of the ways we normally use technology or designed to contrive exceptional situations where defenses are circumvented

© 2012 Carnegie Mellon University

Principle 5: Coordination and Education

- Assurance requires effective coordination among all technology participants and their governing bodies
  - Protection must be applied broadly across the people, processes, and technology because the attacker will take advantage of all possible entry points
  - Authority and responsibility must be clearly established at an appropriate level in the organization to ensure effective participation

© 2012 Carnegie Mellon University

Principle 6: Well Planned and Dynamic

- An adaptive response is required for assurance (justified confidence that software functions as intended) because the threat is always changing. Assurance implementation must represent a balance among governance, construction, and operation and is highly sensitive to changes in each of these areas
  - Engineering challenge: Assurance cannot be added later; you must build to the level of acceptable assurance that you need
  - No one has resources to redesign systems every time the threat changes
  - Assurance cannot be readily adjusted upward after the fact

© 2012 Carnegie Mellon University

Principle 7: Measurable

- A means to measure and audit overall assurance must be built in. If you can't measure it you can't manage it
  - All elements of the socio-technical environment must tie together (practices, processes, procedures, etc.)
    - Measuring individual elements may be useful but not sufficient evidence for overall assurance
    - Each participant will address only the assurance for which they are held accountable
  - Effective measurement is well supported by sound engineering and organizational principles - well formed and consistently applied processes are critical to ensure an appropriate measurable response

© 2012 Carnegie Mellon University

Questions?

© 2012 Carnegie Mellon University

Looking Ahead: Lecture #2

I. Software assurance practices

II. Software assurance lifecycle models

III. Software assurance maturity models

© 2012 Carnegie Mellon University

Reading Assignment

- Software Security Engineering book – Chapters 1 & 2: http://www.amazon.com/Software-Security-Engineering-Project-Managers/dp/032150917X

- Saltzer & Schroeder paper: http://web.mit.edu/Saltzer/www/publications/protection/

- HICSS Principles paper: http://csdl.computer.org/dl/proceedings/hicss/2012/4525/00/4525f368.pdf

- Drone attack articles:
  - http://www.informationweek.com/government/security/air-force-says-drone-virus-is-no-threat/231900741?queryText=Air%20Force%20Says%20Drone%20Virus%20Is%20No%20Threat
  - http://csdl.computer.org/dl/mags/co/2011/11/mco2011110015.pdf

© 2012 Carnegie Mellon University

Homework Assignment # 1

1. (80%) Surf the web and find 4 different actual examples of successful intrusion
   - one that resulted from human error, such as giving out a password or downloading a virus
   - one that resulted from a system configuration error
   - one that resulted from software provided an intrusion opportunity because of a flawed development process
   - one that resulted from a vulnerability in a COTS product

Describe how each of these attacks could have been avoided. Consider changes in policy, configuration management, software development practice, and COTS acquisition practices.

2. (20%) Compare and contrast the HICSS Principles paper with the Saltzer and Schroeder Principles paper.

Turn this in BEFORE the next class

© 2012 Carnegie Mellon University

NO WARRANTY

THIS MATERIAL OF CARNEGIE MELLON UNIVERSITY AND ITS SOFTWARE ENGINEERING INSTITUTE IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

Use of any trademarks in this presentation is not intended in any way to infringe on the rights of the trademark holder.

This Presentation may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.