Lecture 1

Introduction to Software Security

References
1. Noopur Davis, "Developing Secure Software," SEI, DHS
2. Noopur Davis, "Secure Software Development Life Cycle Process," Build Security In (DHS)

Introduction - Terms

- Defect
  - Fault and error in software
  - Failure in software
- Vulnerability
  - A security weakness in design and implementation
  - That might be exploited to cause loss or harm
    - E.g., Vulnerable to unauthorized data manipulation

Introduction- Terms

- Threat
  - A set of circumstances that has the potential to cause loss or harm
- Attack
  - A human (another system) who exploits a vulnerability perpetrates the system
- Security countermeasure (security measure; security service)
  - An action, device, procedure, or technique that removes or reduces a vulnerability

3

3

Introduction – Security with Development

- Most security vulnerabilities comes from defects
  - Defects introduced unintentionally into software during development
  - To reduce software vulnerabilities significantly
    - Reduce the overall defects
  - To achieve reduction in vulnerabilities
    - Focus on the specific types of defects
- Security integrated with software development lifecycle
  - Security must be “built-in” while the software is being developed
  - Not just “bolted-on” after development

4

4

Defective Software is Seldom Secure

- Software benchmark studies
  - Average defects of released software
    - About 1 to 7 defects per thousand lines of code
- Analysis by the SEI's CERT
  - Over 90% of software security vulnerabilities
    - Caused by known software defect types
  - Most software vulnerabilities arisen from common causes
    - Top 10 causes account for about 75% of all vulnerabilities, E.g.,
      - Improper input validation, Integer overflow

5

5

Defective Software is Seldom Secure

- Another analysis of forty-five e-business applications
  - 70% of the security defects were software design defects
  - Caused by sophisticated architectural and design
    - Inadequate authentication
    - Invalid authorization
    - Incorrect use of cryptography
    - Failure to protect data
    - Failure to carefully partition applications

6

6

Defective Software is Seldom Secure

- But most design defects caused by simple oversight
  - declaration errors
  - logic errors
  - loop control errors
  - conditional expression errors
  - failure to validate input
  - interface specification errors
  - configuration errors
  - failure to understanding basic security issues
- Also, vulnerabilities are a result of
  - Poor coding, testing, and sloppy software engineering

7

7

Defective Software is Not Inevitable

- Common response to defects in software
  - Inherently prone to defects and
  - Defective software somehow inevitable
- However, a recent study
  - When software engineers follow defined, measured, and quality-controlled practices,
  - Defects of products can be reduced to
    - An average of 0.6 defects per thousand lines of code
    - 10 to 100 times fewer defects

8

8

Cost of Reducing Defects

- Does it cost too much to reduce defects?
  - Defect-free software consistently
    - Meet their schedules and spend less time on software repair
  - The fewer the defects in the software,
    - The lesser the schedule error
  - Software with vulnerabilities
    - Needs to pay tangible costs of
      - fixing and releasing patches for vulnerabilities

9

9

Secure Software Development

- To reduce vulnerabilities in software
  - Defects management
    - through out the software development life cycle
  - Security concerns
    - through out the software development life cycle

10

10

Defect Management through SDLC

- Defect management includes
  - Defect removal
  - Defect measurement
- Defect removal
  - Each time defects are removed, they should be measured
    - To decide whether to move the next step, or to stop and take corrective actions
  - Multiple defect removal points in the SDLC

11

11

Defect Management through SDLC

- Defect removal and measurement points in SDLC
  - Threat modeling
  - Architectural analysis
  - Design verification
  - Design review
  - Code review
  - Static and dynamic code analysis
  - Unit test
  - Penetrate test
  - System test

12

12

Security through SDLC

- Apply security measures (practices) to SDLC
  - Understand common causes of security vulnerabilities
    - Common Weakness Evaluation (CWE) Top 25 Most Dangerous Software Errors
  - Define best practices against common causes
    - Threat modeling
    - Security risk analysis
    - Secure design principles, e.g.,
      - Defense in depth, least privilege
    - Static code analysis
    - Checklist-based inspections and reviews
    - Testing methods

13

13

Secure Software Development Life Cycle Process

- Four SDLC Activities
  - Security engineering, assurance, project management, risk activities
- Security Engineering Activities
  - Process that develops secure software
    - E.g., Security requirement specifications, secure design, use of static analysis tools
- Security Assurance Activities
  - Process that establishes confidence that a software meets its security needs
    - E.g., verification, validation, expert review, product review

14

14

Secure Software Development Life Cycle Process

- Project Management Activities
  - Project planning and tracking resource allocation and use for security activities
- Risk activities
  - Security risk identification and management activities

15

15

Microsoft's Trustworthy Computing Security Development Lifecycle

- MS adopted security activities and deliverables in each phase of MS software development process
  - Requirement's phase
    - Definition of security feature requirements and assurance activities
  - Design phase
    - Threat modeling
  - Implementation
    - Use of static analysis tool and code reviews
  - Testing
    - Security testing
  - Verification
    - Final code review of new and legacy code
  - Release phase
    - Final security review by the Central MS Security team

16

16

Agile Methods

- Emphasize the importance of writing well-structured code and investing effort in code improvement
  - Short development iterations, minimal design upfront, for anyone to change any part of code, minimal or no documentation
  - E.g., Extreme Programming, Scrum, Lean Software Development, Crystal Methodologies, Feature Driven Development, and Dynamic Systems Development Methodology
  - Incremental development methods
  - Small and medium-sized business systems

17

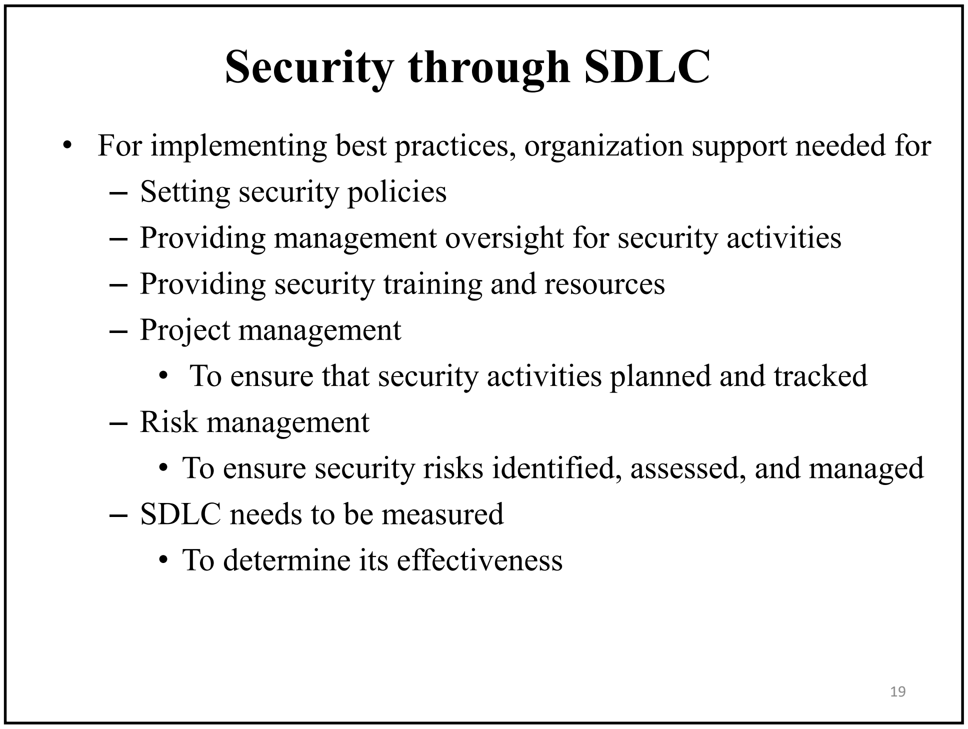
17

Agile Methods

- Suited for applications where the requirements usually change rapidly during the development process
- Likely to make system maintenance more difficult and expensive
- Conflict with secure SDLC
  - Threat modeling vs short requirement and design
- Security activities
  - Focused on code

18

18



Security through SDLC

- For implementing best practices, organization support needed for
  - Setting security policies
  - Providing management oversight for security activities
  - Providing security training and resources
  - Project management
    - To ensure that security activities planned and tracked
  - Risk management
    - To ensure security risks identified, assessed, and managed
  - SDLC needs to be measured
    - To determine its effectiveness

19

19