What is Knowledge?

- It is more than simply a list of things we know or a collection of facts
- Knowledge vs Information:
  - Knowledge is information in context—information put to work using processes and procedures
  - Examples:
    - Information: A checklist of potential security bugs in C and C++
    - Knowledge: The same information built into a static analysis tool is knowledge.
- Relying on information alone (e.g. a checklist of common programming errors) is usually inadequate for an organization’s security.

Common Misconceptions

1. Software security is not a coding issue
    a. A checklist of common coding issues to avoid, intended for developers to read will never be complete, and hard for developers to remember.
    b. Using those errors to build static analyzer is much better.
2. Software security is not simply adopting various security features and/or conventions
    a. Adopting generic classes for filtering input: An application dependent white-list of properties is better than generic black-lists.
3. Overuse of the checklist.
    a. Checklists are by their very definition incomplete.
    b. Attackers are also aware of common checklists
        i. "No virus writer worth his salt will release a new virus without first running every available commercial antivirus checker against it as an acceptance test"

Necessity of Knowledge Management

- Security knowledge is transferred to younger generations through apprentice approach.
- However number of experienced security practitioners is low.
- So only few people can directly access knowledge earned through his/her mentor’s experience.
- Even in these cases, practitioners can access the knowledge of one or two masters.

Software Security Knowledge Catalogs

- **Principles:** statement of general security wisdom derived from experience.
  - Useful for diagnosing architectural flaws in software and practicing good security engineering.
  - Relevant SDLC artifacts: Security requirements, Software architecture and design
- **Guidelines:** Recommendation for things to do or to avoid during software development.
  - Guidelines exist for a specific technical context (.NET, Linux kernel etc)
  - Guidelines can help uncover both architectural flaws and implementation bugs.
  - Relevant SDLC artifacts: Security requirements, Software design and code.

Software Security Knowledge Catalogs

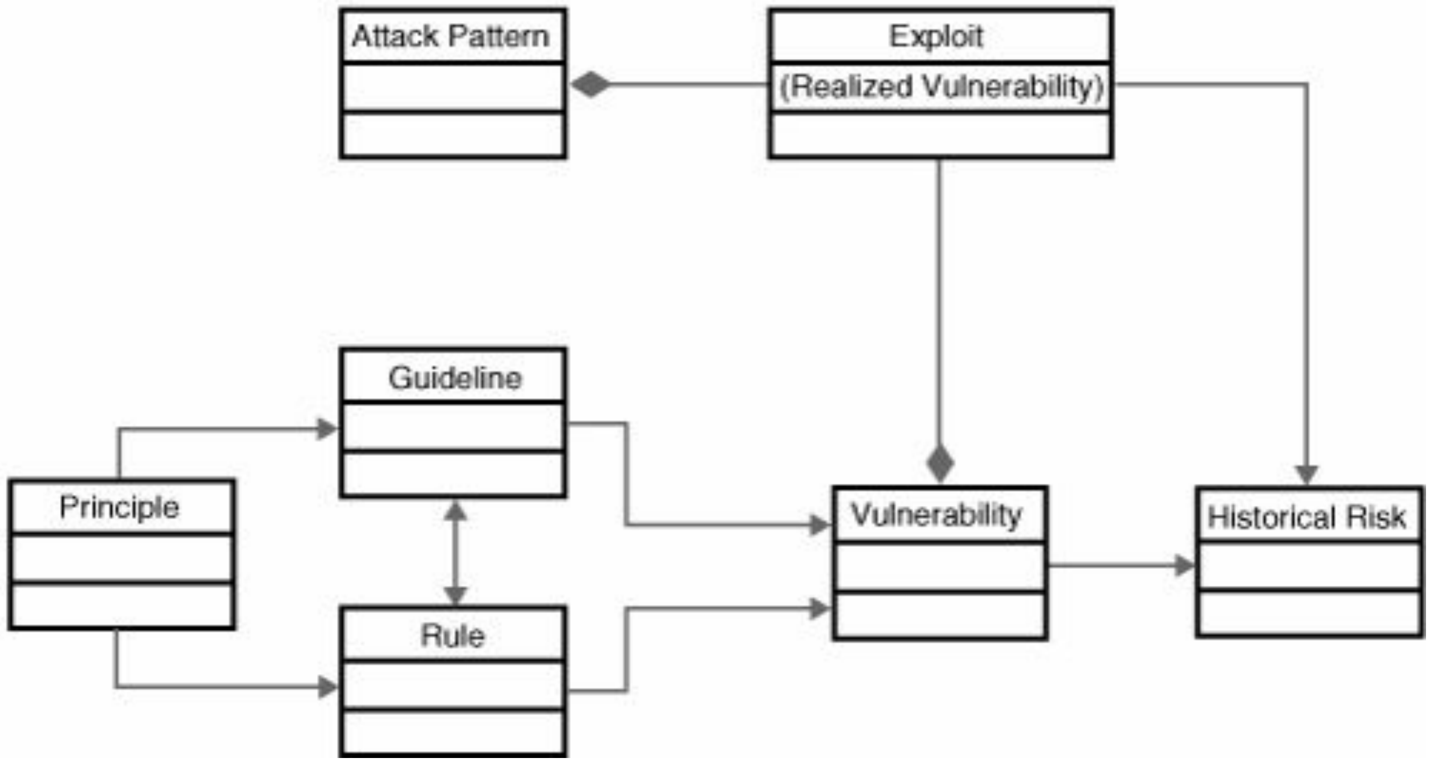
- **Rules:** Recommendation for things to do or to avoid during software development, described at the level of syntax.
  - A rule can be verified through lexical scanning, exists for specific programming languages
  - Rules can help uncover implementation bugs.
  - Relevant SDLC artifacts: Code.
- **Attack Patterns:** generalized pattern developed by reasoning over large sets of software exploits.
  - useful for identifying and qualifying the risk that a given exploit will occur in a software system.
  - designing misuse and abuse cases and specific security tests.
  - Relevant SDLC artifacts: Software design, Abuse cases, test plan, penetration tests

Software Security Knowledge Catalogs

- **Historical Risks:** Risks identified in the course of an actual software development effort.
  - Useful for early identification of potential issues, and potential clues to effective mitigations
  - Relevant SDLC artifacts: Software architecture, design, test plans, deployed software.
- **Vulnerabilities:** Result of defect that can be exploited by an attacker.
  - Relevant SDLC artifacts: Software architecture, design, code, penetration tests, deployed software.
- **Exploits:** An exploit is a particular instance of an attack on a computer system that leverages one or more vulnerabilities.
  - Relevant SDLC artifacts: Penetration tests, deployed software.

Note: These catalogs are grouped into 3 categories: prescriptive, diagnostic and historical knowledge.

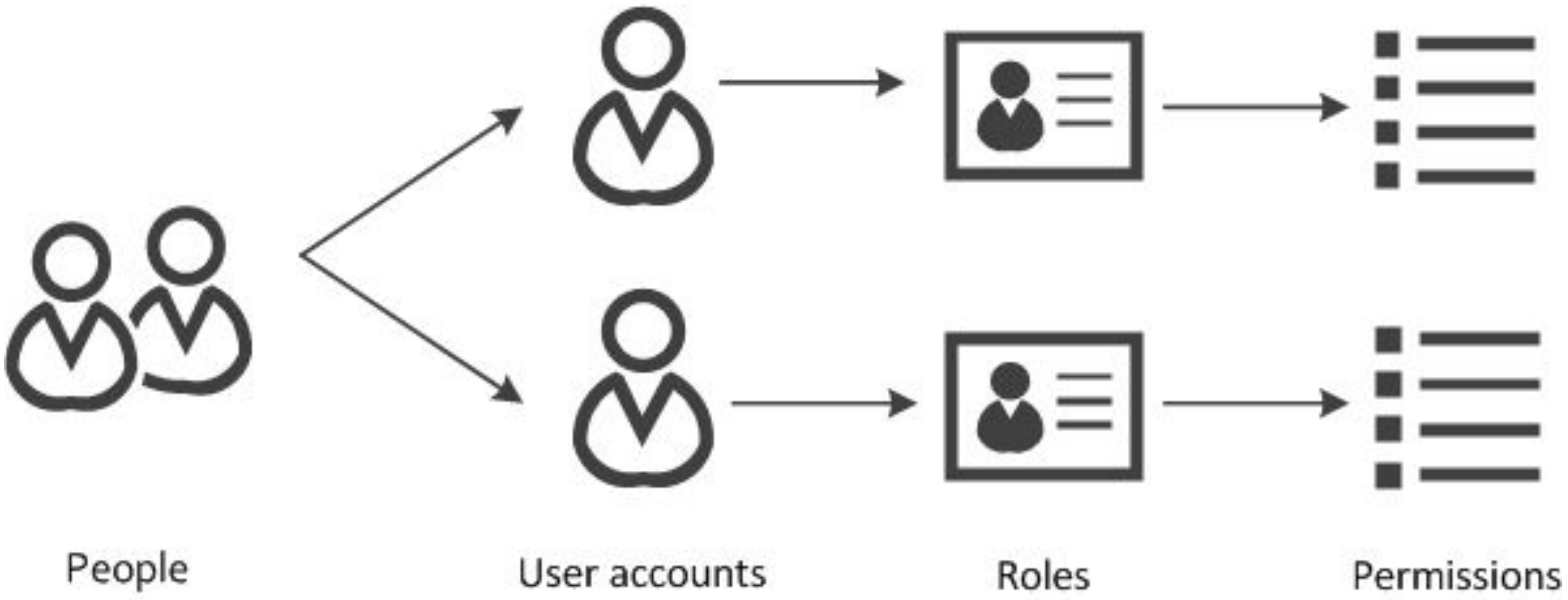
Organization and Interrelation of software security knowledge



Knowledge Catalog Example (Principle Catalog)

Principle of Least Privilege

Knowledge Catalog: Principle of Least Privilege



Knowledge Catalog: Principle of Least Privilege

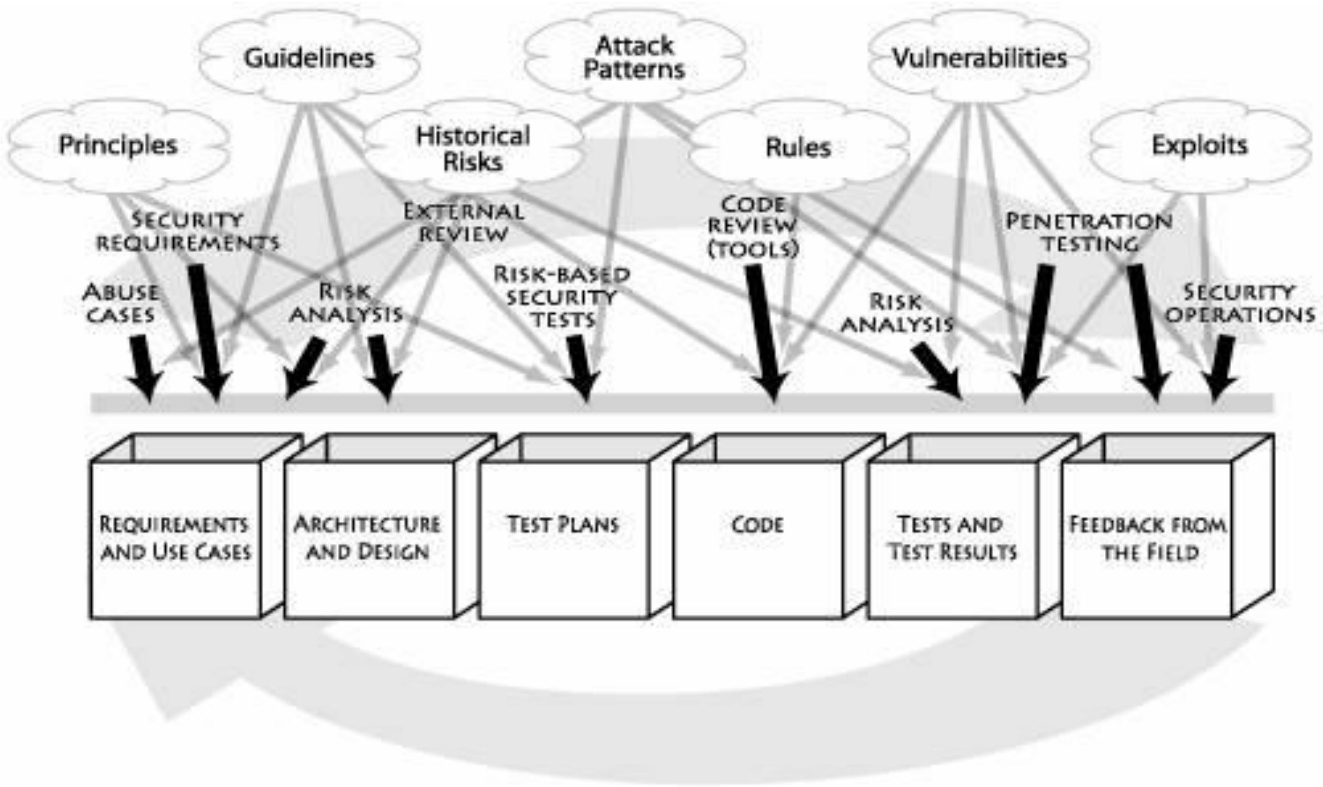
- Every program and every user of the system should operate using the least set of privileges necessary to complete the job.
- Primarily, this principle limits the damage that can result from an accident or error. It also reduces the number of potential interactions among privileged programs to the minimum for correct operation so that unintentional, unwanted, or improper uses of privilege are less likely to occur.
- Thus, if a question arises related to misuse of a privilege, the number of programs that must be audited is minimized.

Security Knowledge and The Touchpoints

- One effective way to apply security knowledge is through the use of software security best practices such as the touchpoints. For example, rules are extremely useful for static analysis and code inspection activities.

- Software security best practices and their associated knowledge catalogs can be applied regardless of the base software process being followed.

Security Knowledge and The Touchpoints



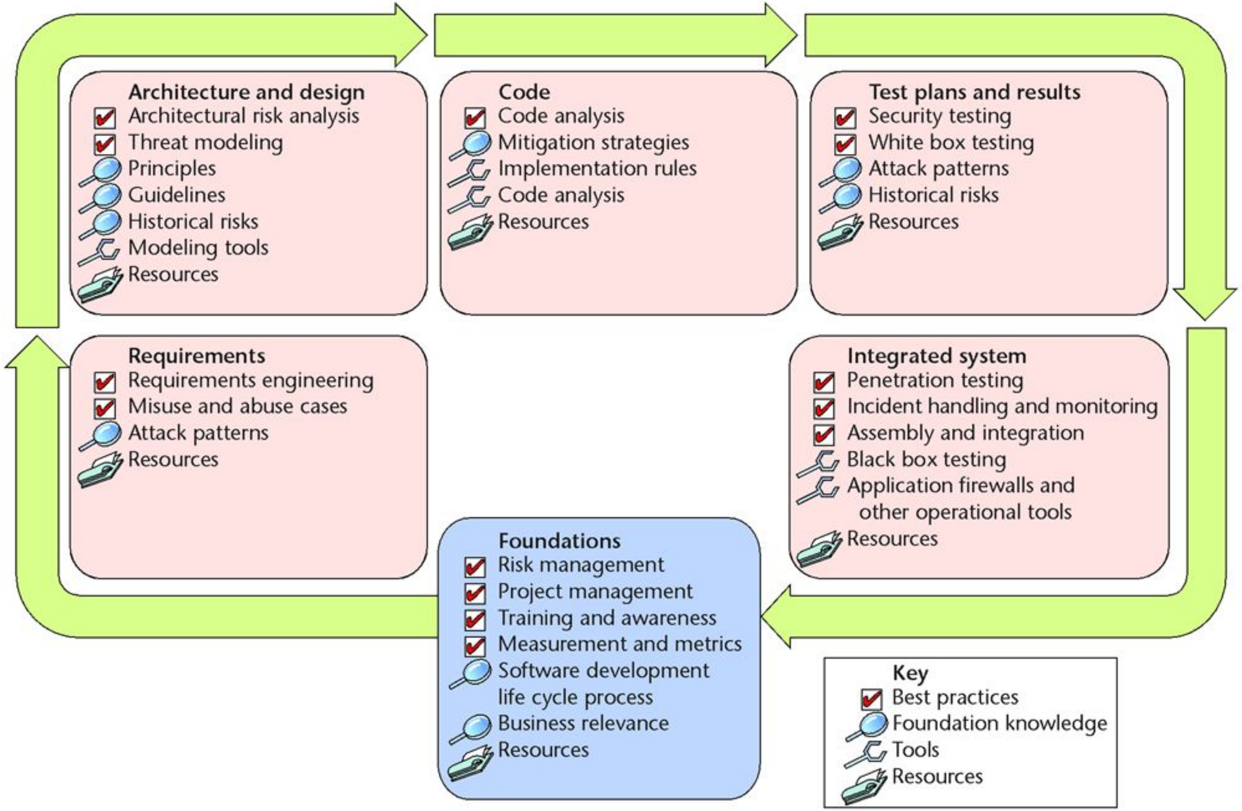
The Department of Homeland Security *Build Security In* Portal (DHS BSI PRTL)

- The U.S. Department of Homeland Security is developing a software security portal known as - DHS BSI PRTL.
- This portal aims to provide a common, accessible, well-organized set of information for practitioners wishing to practice software security.
- The portal effort is expressly aimed at the problem of encapsulating, expanding, and spreading software security knowledge.

Build Security In Portal (DHS BSI PRTL)

- This portal is intended for software developers and software development organizations who want information and practical guidance on how to produce secure and reliable software.
- The catalog is based on the principle that software security is fundamentally a software engineering problem that we must address systematically throughout the SDLC

The Organizing Concept For The BSI portal



Architecture and design
- ✓ Architectural risk analysis
- ✓ Threat modeling
- 🔍 Principles
- 🔍 Guidelines
- 🔍 Historical risks
- 🔧 Modeling tools
- 📇 Resources

Code
- ✓ Code analysis
- 🔍 Mitigation strategies
- 🔧 Implementation rules
- 🔧 Code analysis
- 📇 Resources

Test plans and results
- ✓ Security testing
- ✓ White box testing
- 🔍 Attack patterns
- 🔍 Historical risks
- 📇 Resources

Requirements
- ✓ Requirements engineering
- ✓ Misuse and abuse cases
- 🔍 Attack patterns
- 📇 Resources

Integrated system
- ✓ Penetration testing
- ✓ Incident handling and monitoring
- ✓ Assembly and integration
- 🔧 Black box testing
- 🔧 Application firewalls and other operational tools
- 📇 Resources

Foundations
- ✓ Risk management
- ✓ Project management
- ✓ Training and awareness
- ✓ Measurement and metrics
- 🔍 Software development life cycle process
- 🔍 Business relevance
- 📇 Resources

Key
- ✓ Best practices
- 🔍 Foundation knowledge
- 🔧 Tools
- 📇 Resources

BSI Portal

https://www.us-cert.gov/bsi

Related Paper/Journal

https://www.garymcgraw.com/wp-content/uploads/2015/11/bsi7-knowledge.pdf

Thank You