Security Engineering fiche

Pierre Colson

December 2022

Contents

Markdown version on *github*
Compiled using *pandoc* and *`gpdf` script*

Introduction

- Security is usually added on, not engineered in
  - Standard security properties (CIA) concern *absence of abuse*
    - **Confidentiality**: No proper disclosure of information
    - **Integrity** No proper modification of information
    - **Availability** No proper impairment of functionality/service
- Sofware is not *continuous*
- Hackers are not typical users
  - A system is **safe** (or **Secure**) if the environment cannot cause ot to enter an unsafe (insecure state)
    - So, abstractly, security is a *reachability* problem
- The adversary can exploit not only the system but also the world
- Security Engineering = Software Engineering + Information Security
- **Software Engineering** is the application of systematic, quantifiable approaches to the development, operation, and maintenance of software; i.e applying engineering to software
- **Information Security** focuses on methods and technologies to reduce risks to information assets
- **Waterfall model**
  - *Requirement engineering* : What the system do ?
  - *Design* : How to do it (abstract) ?
  - *Implementation* : How to do it (concrete) ?
  - *Validation and verification* : Did we get it right ?
  - *Operation and maintenance*
  - Problems
    - The assumption are too strong
    - Proof of concept only at the end
    - Too much documentation
    - Testing comes in too late in the process
    - Unidirectional
- **Summary**
  - Methods and tools are needed to master the complexity of software production

  - Security needs particular attention
    * Security aspects are typically poorly engineered
    * Systems usually operate in highly malicious environment
  - One needs a structured development process with specific support for security

Requirements Engineering

- **Requirements engineering** is about elicting, understanding, and specifying what the system should do and which properties it should satisfy
- **Requirements** specify how the *system should and should not behave* in its intended environment
  - **Functional requirements** describe *what system should do*
  - **Non-functional requirements** describe *constraints*
- Security almost always conflicts with *usability* and *cost*
- Analysis → Specification → Validation → Elicitation → Analysis ...
  - **Elicitation** : Determine requirements with stakeholders
  - **Analysis** : Are requirements clear, consistent, complete
  - **Specification** : Document desired system behavior
    * *Functionality* : what the softwate should do
    * *External interfaces* : how it interacts with people, the system's hardware, other software and hardware
    * *Performance* : its speed, availability, response time, recovery tim eof various software functions, etc
    * *Attributes* : probability, correctness, maintaintability, security, etc.
    * *Design constraints imposed on the implementation* : implementation language, resource limit, operating system environment, any required standard in effect, etc.
  - **Validation** : Are we building the right system?
- Standards and guidlines provide good strating points, but they must be refined and augmented to cover concrete systems and the infornations they process
- **Authorization policy** : knowing which data is critical is not wnough
  - Information access policy (Confidential, Integrity)
  - Good default is base on *least-priviledge*
- **Summary**
  - Security requirements are both functinal and non-functional
  - Standards and guidlines help with the high level formalization
  - Models help to concretize the details
    * However full details usually only present later after design
  - Models also useful for risk analysis

Summary

- **Introduction**
  - Methods and tools are needed to master the complexity of software production
  - Security needs particular attention
    * Security aspects are typically poorly engineered
    * Systems usually operate in highly malicious environment
  - One needs a structured development process with specific support for security
- **Requirements Engineering**
  - Security requirements are both functinal and non-functional
  - Standards and guidlines help with the high level formalization
  - Models help to concretize the details
    * However full details usually only present later after design
  - Models also useful for risk analysis