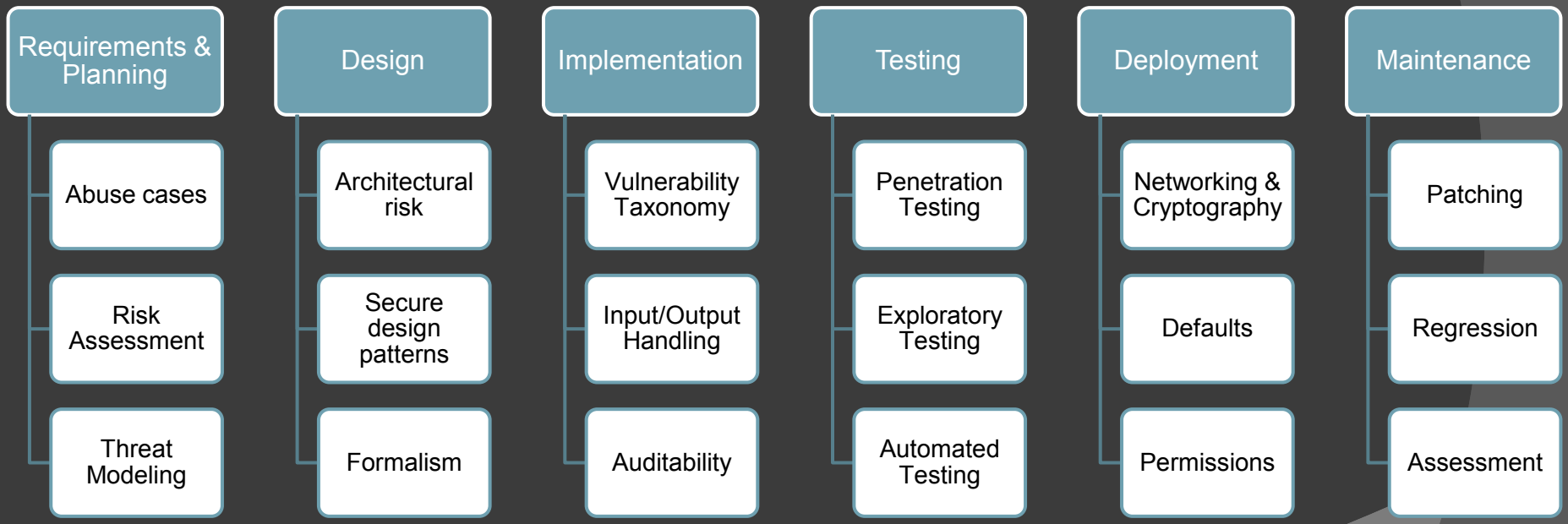
Engineering Secure Software

DEVELOPMENT LIFECYCLE & PRINCIPLES

A Ubiquitous Concern

- You can make a security mistake at every step of the development lifecycle
- ***Requirements*** that allow for privacy violations
  e.g. secretary can view everyone's patient records
- Introducing a ***design*** flaw, e.g. giving plug-ins total access
- Introducing a ***code***-level vulnerability, e.g. buffer overflow
- Missing a vulnerability in code ***inspections*** & ***testing***
- Introducing a vulnerability by regression in ***maintenance***
- Not facilitating a secure **deployment**, e.g. installation defaults

Security at Every Step



Core Security Properties

- Software security breaks into these categories
  - Confidentiality
  - Integrity
  - Availability
- Very broad, multi-dimensional categories
- Some people add in “auditability”, but we consider that part of “integrity”

Confidentiality

- The system must not disclose any information intended to be hidden

  E.g. your credit card information on a website

- Note: open source software can still be confidential

Integrity

- The system must not allow assets to be subverted by unauthorized users

  E.g. changing a prisoner's release date

- We must be able trust what is in the system
  - The data being stored
  - The functionality being executed

Availability

- The system must be able to be available and operational to users

  E.g. bringing down Amazon.com

- These are extremely hard to protect against
  - *Any* system performance degradation that can be triggered by a user can be used for denial of service attacks
  - Concurrency issues, infinite loop, or resource exhaustion

Misc. Philosophies & Proverbs

- Defense in depth
  - If they break into this, they can't get any farther
  - Think Middle-Age castles
  - Original meaning of "firewall", not today's firewall
- Least privilege
  - Every user or module is given the least amount of privilege it needs
  - Evil: `sudo chmod -R a+rw /`

More Misc. Philosophies & Proverbs

- Fail securely
  - Exceptions put the system into weird states
  - Error message information leak
  - Take care of those exceptions properly!
- Security by obscurity
  - You can't rely upon being obscure to be secure
    - Crowds are good at guessing
    - Insiders are corruptible
  - Some notable exceptions: passwords, encryption keys

Even More Misc. Philosophies & Proverbs

- Detect and record
  - Even if you can't always sift through that data ahead of time
  - Post-mortem analysis
- Don't trust [input | environment | dependencies | *]
  - Know what to trust
  - Know how to trust

Even Even More Misc. Philosophies & Proverbs

- Secure by default
  - Don’t rely on your users to use it correctly
  - Convention over configuration
- Keep it simple
  - YAGNI
  - Speculative generality can be risky
  - Minimize the attack surface