Lecture 1

zaterdag 12 november 2022 17:56

software security is about errors

things (i.e. problems) start very early on

software development

memory corruption

static/dynamic code analysis

safe languages

web technology/security

side-channel attacks

bad usage of security/cryptographic features

race conditions

API abuse

Understanding root causes & problems

attacks, countermeasures and their limitations

software is used widespread (high gain provides incentive)

software has to work with inputs from others

remote access (less likely to get caught personally)

software is run on systems which share resources

software development is error-prone (formalising thoughts in a programming language)

software is complex

→ when several people work on code, expectations may deviate, leading to vulnerabilities

bugs ⇒ vulnerabilities

attacks exploit vulnerabilities

updates can fix bugs later → publish at deadline, even if there is a bug

patches may introduce new vulnerabilities

patches are not always applied by customers

there are financial incentives to attack software

halting problem is undecidable

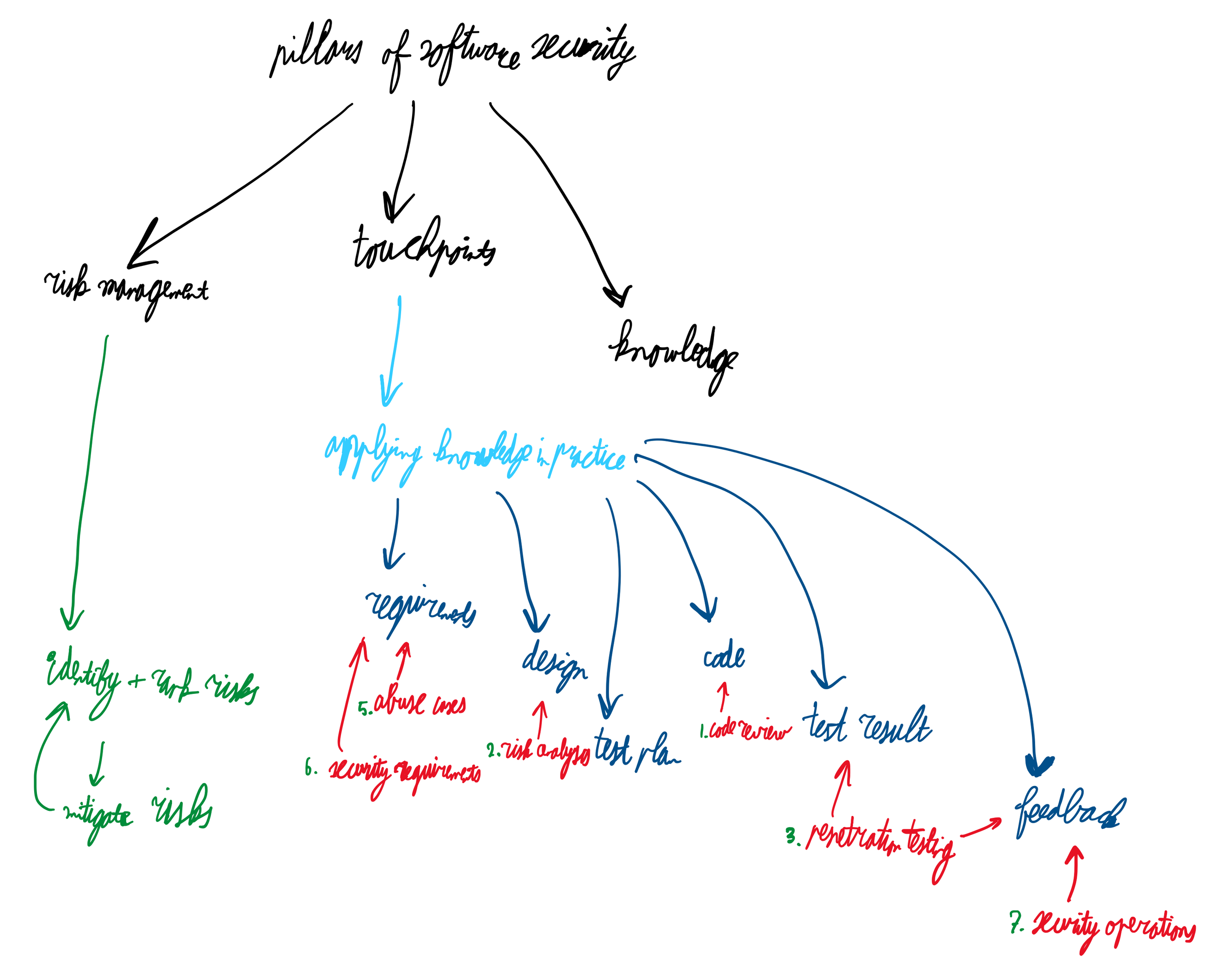
lack of awareness

lack of knowledge

laziness

little incentive to build software securely

put functionality over security



principles

guidelines

historical risks

attack patterns

rules

vulnerabilities

exploits

Errors

↓ has as subset

Weaknesses ← generally used

↓

Vulnerabilities

↓

Exploits

the later in the software development you fix a defect, the more costly it becomes

abuse cases ↪ attacker model

motivation

capabilities

security cast/tail

confidentiality

authentication

integrity

privacy

anonymity

availability

non-repudiation

deniability

7+1 kingdoms

solutions for problems in a kingdom can be re-used in that kingdom

1. input validation/representation
2. API abuse — precondition/assumption/contract violation
3. problems when using security features/tools — incorrect combination of security mechanisms
4. parallelism & consistency
5. error handling/output problems — e.g. leaking information through error messages/improper exception handling/side channels
6. code quality
7. encapsulation/isolation — improper separation of classes
8. environmental assumptions incorrect — DNS, randomness