What is Software Security?

Lecture 1

CS4105 - Software Security

Q2 / 2015 – 2016

Eelco Visser

TU Delft

Team

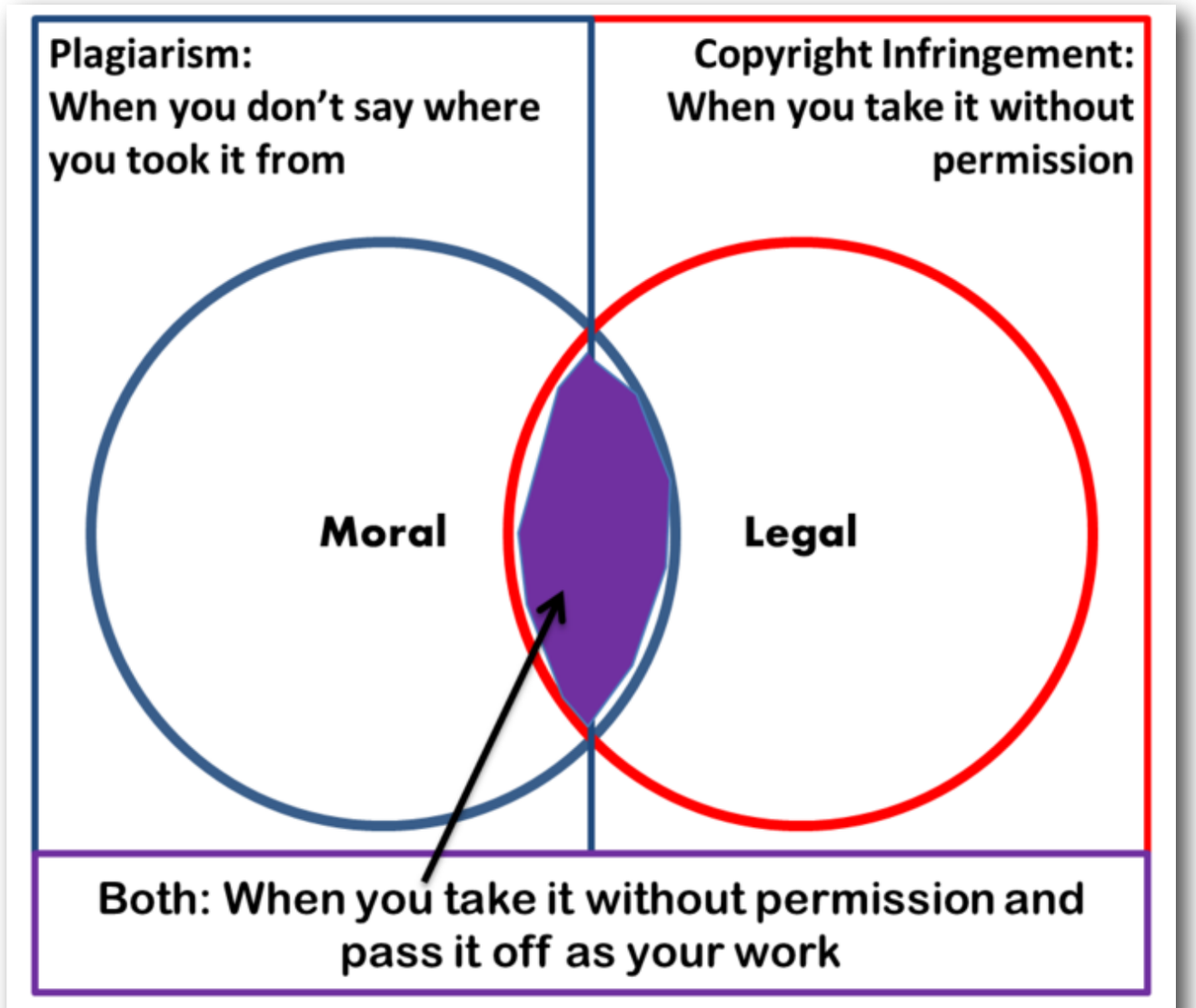
- Prof. Dr. Eelco Visser
  - Software Engineering Research Group
  - Research: Programming Languages
  - TU Delft
  - http://eelcovisser.org
- Prof. Dr. Sandro Etalle
  - Security Group
  - Research: Security
  - TU Eindhoven & U Twente
  - http://www.win.tue.nl/~setalle/
- Danny Groenewegen MSc
  - Software Engineering Group
  - Research: Web Programming Languages (WebDSL)
  - TU Delft
  - http://swerl.tudelft.nl/bin/view/Main/DannyGroenewegen

TU Delft

CODE OF CONDUCT

TU Delft

- [ ] I will not use electronic devices such as laptops, phones, or tablets during lectures.



- My answers to assignments and exams will be my own work (except for assignments that explicitly permit collaboration).

http://en.wikipedia.org/wiki/File:Plagiarism_vs_Copyright_Infringement.png

- ☐ I will not make solutions to assignments and exams available to anyone else. This includes both solutions written by me, as well as any official solutions provided by the course staff.

http://ucblibraries.colorado.edu/about/images/Xerox5135.jpg

- ☐ I will not engage in any other activities that will dishonestly improve my results or dishonestly improve/hurt the results of others.

Motivation

Many security problems in software systems are due to careless use of unsafe programming techniques.

Preventing security problems should be an integral part of the software development process.

The course studies

- the nature of security vulnerabilities in software systems
- techniques to detect and prevent these problems
- the embedding of these techniques in a security-aware software development process.

TU Delft

Objectives

At the end of this course you should understand

- the nature of security vulnerabilities in software systems
- principles for secure software development
- security testing and dynamic analysis techniques
- static analysis techniques and language-based security

… to some extent

TU Delft

Lectures

- Lecture 1: What is Software Security? (Nov 11)
- Lecture 2: Memory-Based Attacks (Nov 18)
- Lecture 3: Language-Based Security (Nov 25)
- Lecture 4: Vulnerabilities in Web Applications (Dec 2)
- Lecture 5: Language-based Security for the Web (Dec 9)
- Lecture 6: Information Flow and Access Control (Dec 16)
- Lecture 7: Security Testing (Jan 6)

TU Delft

Assignments (Groups of 2)

Security Design and Analysis

- D1: Threat model: describe the functional design of a software system and apply architectural risk analysis (threat model)
- D2: Threat model peer review
- D3: security policies: formulate a security design, including authentication, authorization, and audit policies for the system of D1 and argue why your design is safe
- D4 : security policies peer review

Security Bugs and Language-Based Security

- I1: Buffer overflows: construct an attack by exploiting a buffer overflow vulnerability
- I2: safety by construction: implement a translation from a high-level language to a low-level language that ensures safety properties
- I3: web security techniques: examine security vulnerabilities and counter measures in web programming

TU Delft

Exam

- January 27, 2016
- Understanding, concepts
- Multiple choice and open (essay) questions

- Using WebLab

TU Delft

Grading

- Assignment grade = Average(D1:D4,I1:I3)
- Final grade = Average(Exam Grade, Assignment Grade)
- Pass = Assignment Grade >= 5
  & Exam Grade >= 5
  & Final Grade >= 5.75

TU Delft

WebLab

https://weblab.tudelft.nl

Search

WebLab Courses About Admin E. Visser Sign out

WebLab

https://weblab.tudelft.nl

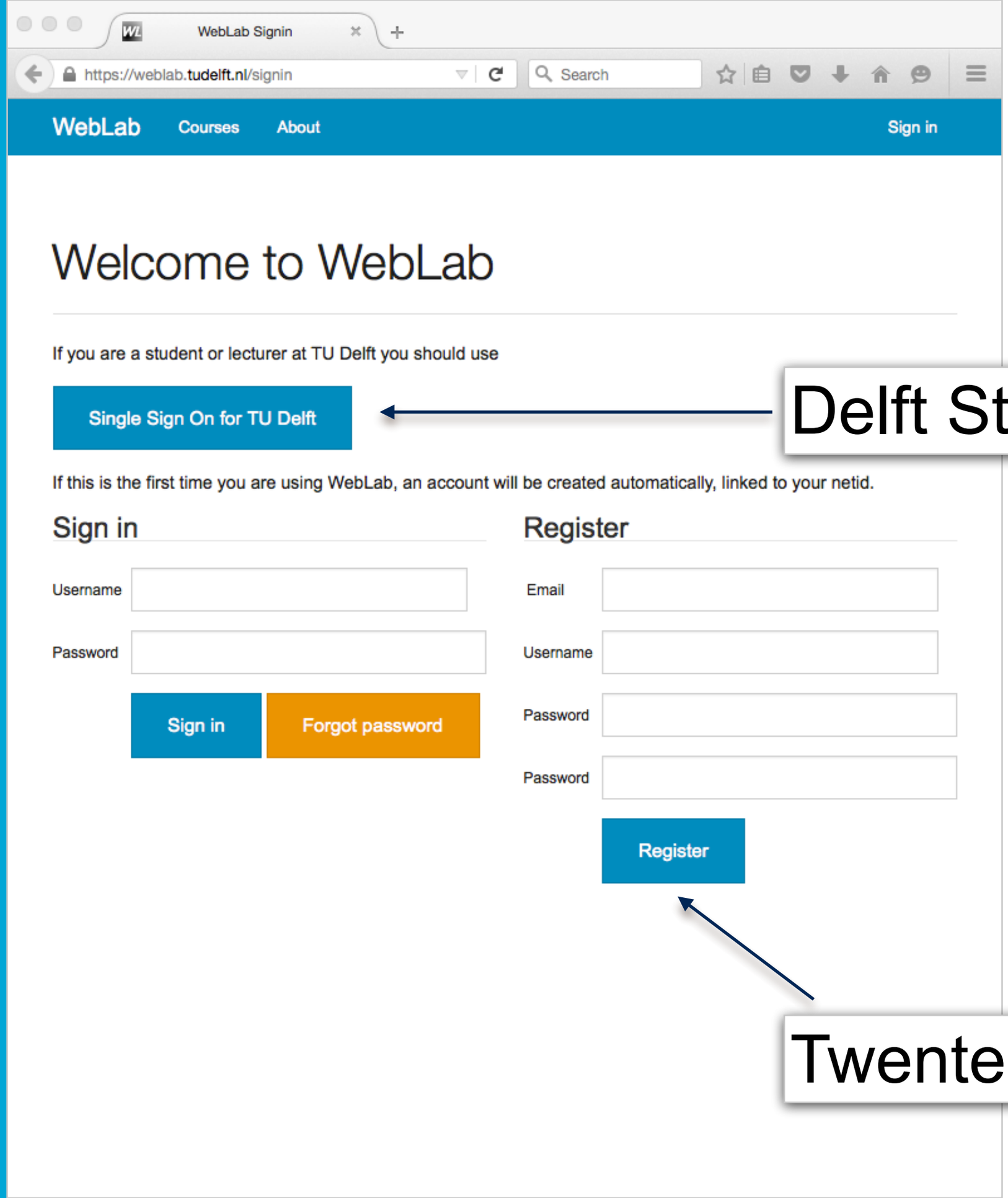
Course Catalog

Type to filter showing: 1 - 15 / 15 (15)

Code	Name	Institution
CS4105	Software Security	
DA-COPL	Concepts of Programming Languages	TU Darmstadt
DA-TSPL	Type Systems of Programming Languages	TU Darmstadt
DA-metaprog	Discussion Seminar on Metaprogramming	TU Delft
IN4303	Compiler Construction	TU Delft
IN4333	Language Engineering Project	
Informatica-VO	Algoritmiek voor informaticadocenten uit het VO	
PSU-CS558	Programming Languages	
TI1220	Concepts of Programming Languages	TU Delft
TI1316	Algoritmen en Datastructuren	TU Delft
TI2306	Algoritmiek	TU Delft
TI2606	Concepts of Programming Languages	
TI2726-B / TI2725-C	Embedded Software	TU Delft
WL101	WebLab Demo	WebLab
WL102	WebLab Test	WebLab

1 - 15 / 15 (15) 25

TU Delft



Delft Students

Twente Students

TU Delft

https://weblab.tudelft.nl/beta/

WebLab Courses About Admin E. Visser Sign out

WebLab - Course Catalog

Filter Courses

CS4105
Software Security

TI2726-B / TI2725-C
Embedded Software
TU Delft

TI2306
Algoritmiek
TU Delft

PSU-CS558
Programming Languages

IN4303
Compiler Construction
TU Delft

TI1316
Algoritmen en Datastructuren
TU Delft

DA-COPL
Concepts of Programming Languages
TU Darmstadt

TI2606
Concepts of Programming Languages

IN4333
Language Engineering Project

TU Delft

CS4105: Software Security

https://weblab.tudelft.nl/beta/cs4105/2015-2016/

Search

WebLab Courses About E. Visser Sign out

CS4105 /

2015-2016

Course Edition | News | Course Rules

Software Security

Course: CS4105 Edition: 2015-2016

From November 8, 2015 until January 31, 2016

Course Information

- Home
- All editions
- News archive
- Course rules

Enroll

One can enroll until **Mon, Nov 30, 2015 00:00**

Enroll

Course staff

Lecturers

- E. Visser

Assistants

- D. M. Groenewegen

About the Course

Many security problems in software systems are due to careless use of unsafe programming techniques. Preventing security problems should be an integral part of the software development process. The course studies the nature of security vulnerabilities in software systems, techniques to detect and prevent these problems, and the embedding of these techniques in a security-aware software development process.

News

Archive

Enroll

TU Delft

CS4105: Software Security

https://weblab.tudelft.nl/beta/cs4105/2015-2016/assignment/4445/view

Search

CS4105 / 2015-2016 /

Software Security

Submission | Assignment | Statistics/Dates

Description | Assignment Info

The course consists of five assignments that need to be submitted during the quarter and an exam. All need to be submitted through WebLab.

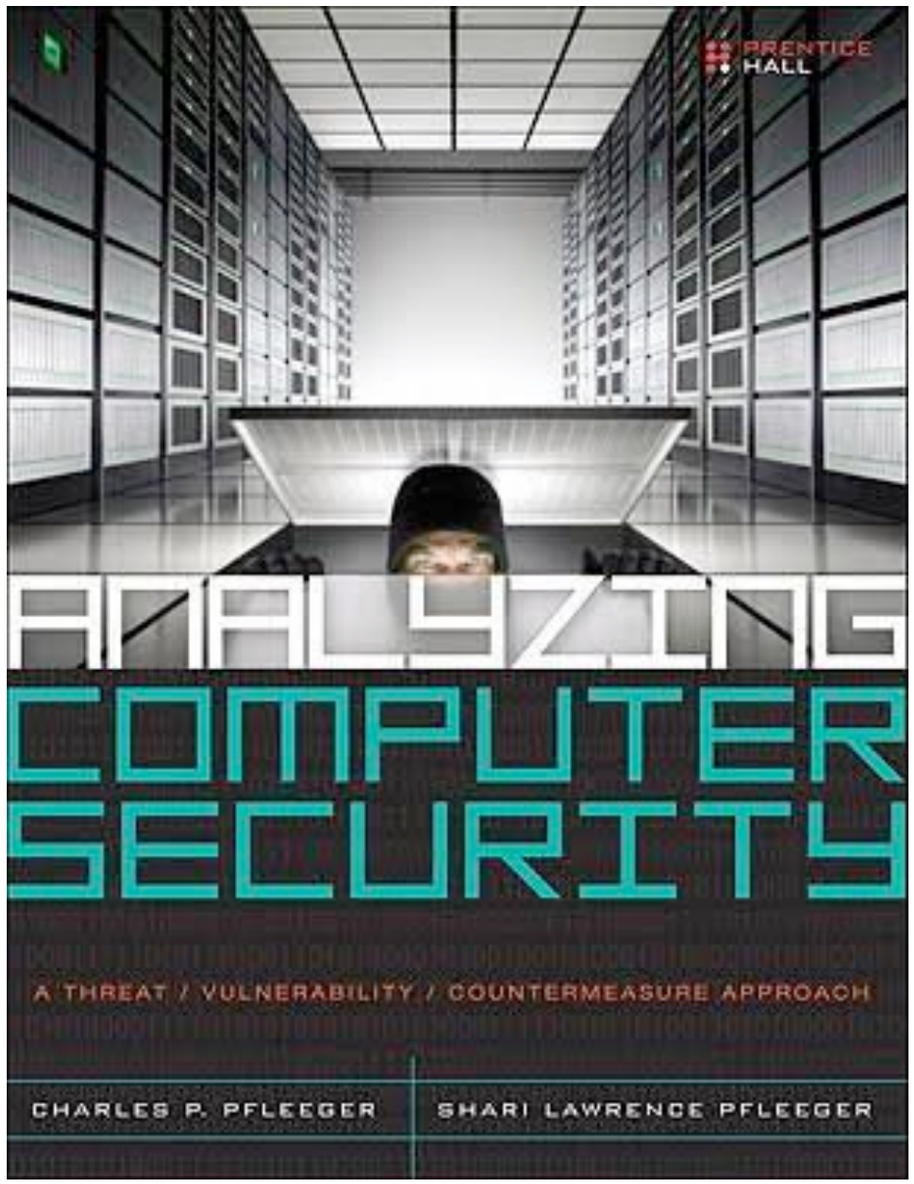
Assignment	Weight	Due
Software Security	100.0	
Lab	50.0 of total 100.0	
Exam	50.0 of total 100.0	

TU Delft

RECOMMENDED LITERATURE

Analyzing Computer Security

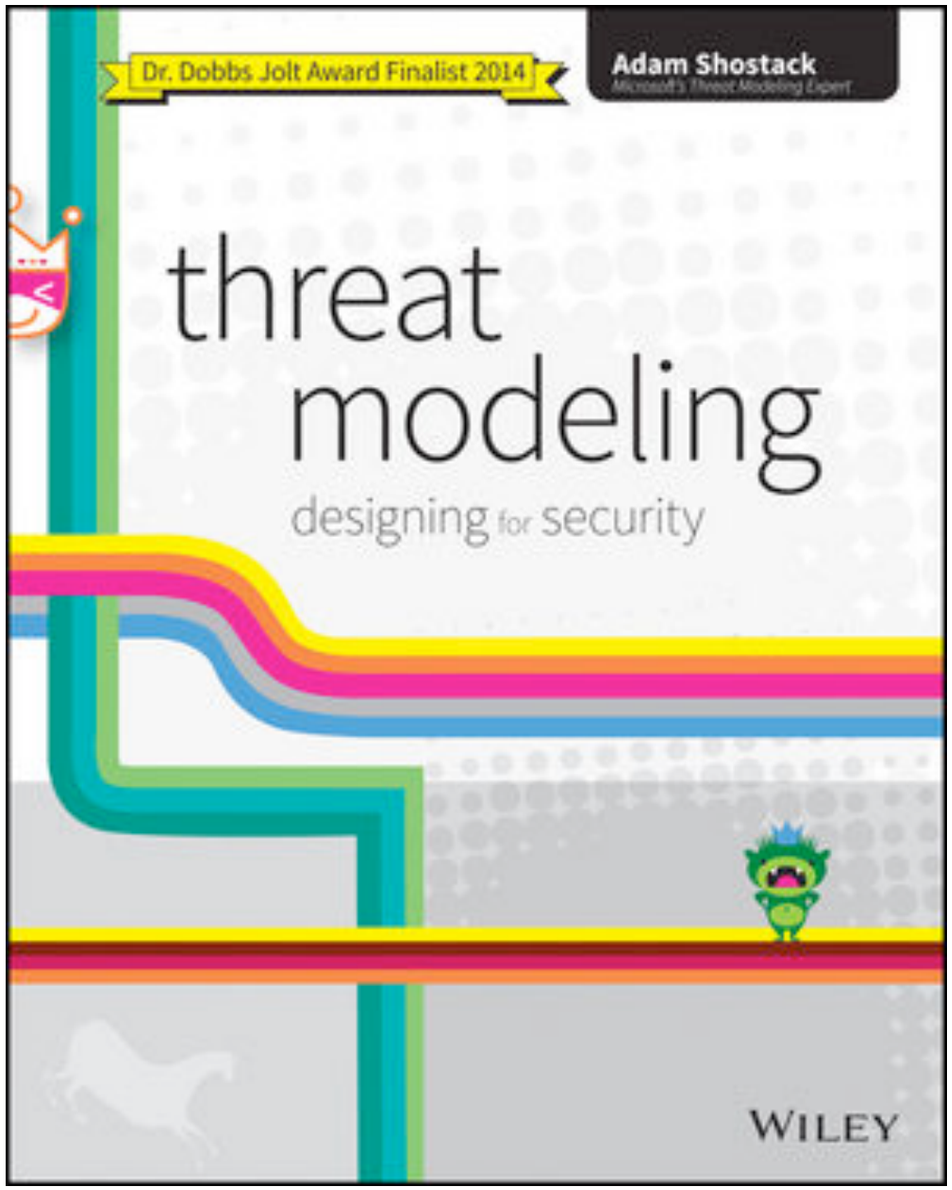
- Charles P. Pfleeger & Shari Lawrence Pfleeger
- Analyzing Computer Security: A Threat / Vulnerability / Countermeasure Approach
- Prentice Hall, 2011
- Available as Kindle book from Amazon



TUDelft

Threat Modeling: Designing for Security

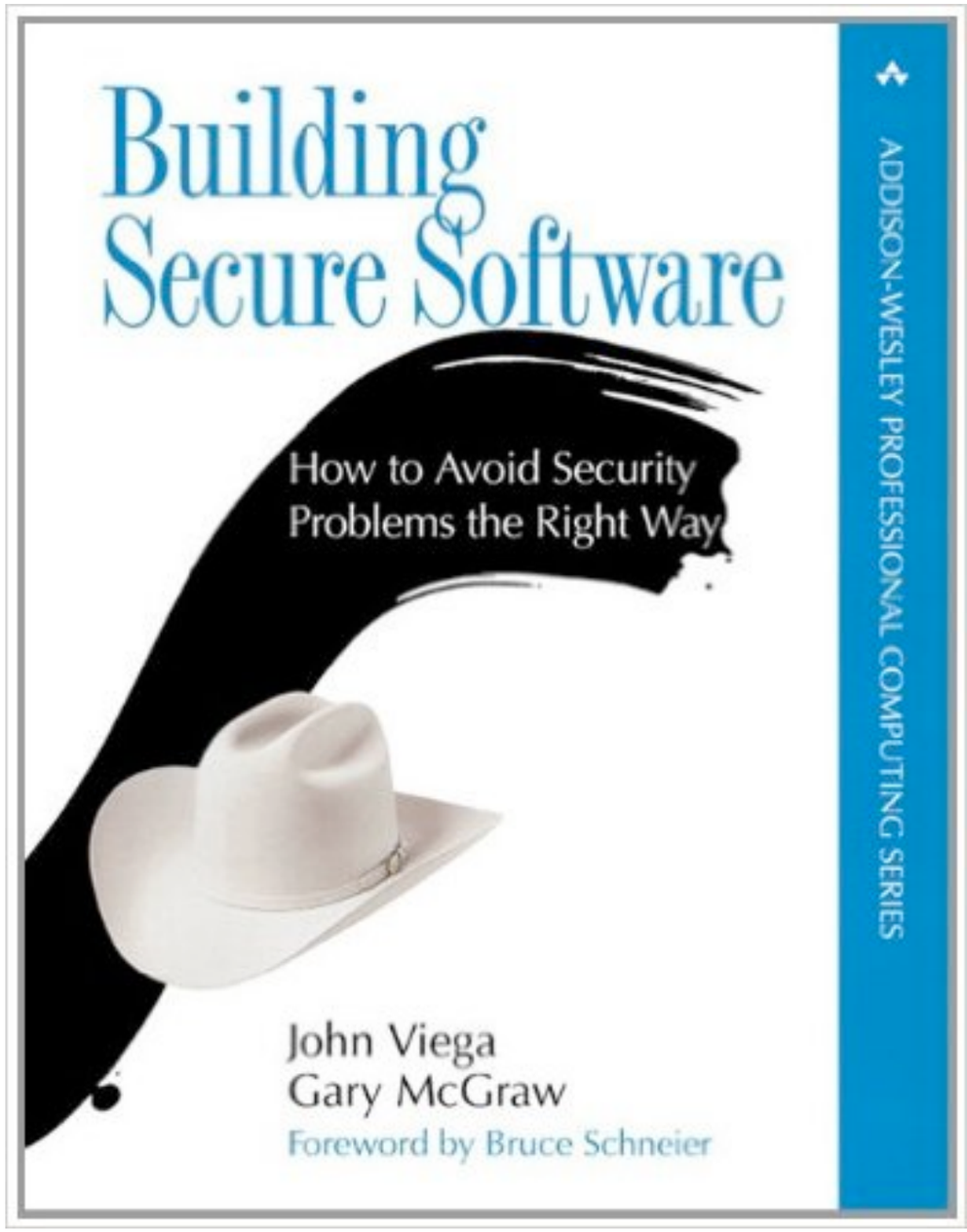
- Adam Shostack
- Wiley, 2014
- Available as Kindle book from Amazon



TU Delft

Building Secure Software

- John Viega & Gary McGraw
- Addison-Wesley, 2001



TUDelft

Literature

- Pfleeger & Pfleeger. Analyzing Computer Security: A Threat / Vulnerability / Countermeasure Approach
- Shostack. Thread Modeling: Designing for Security
- Viega & McGraw. Building Secure Software
- Academic papers (available online)
- Websites
- Software Security course on Coursera by Michael Hicks

What is Software Security?

TU Delft

Software Security in the News

Software security is about (notorious) attacks

- HeartBleed
- ShellShock
- DigiNotar
- Your PC in a botnet
- …

that are enabled by buffer overflow vulnerabilities in software

So, software security is: understanding buffer overflows and fix them …

But, why are (/ do we know that) these attacks problems?

Software Engineering

Software engineering is the application of engineering to the design, development and maintenance of software

Functional requirements
- functions of a system: inputs, behavior, outputs
- use cases
- what the system should do

Non-functional requirements
- performance
- reliability
- scalability
- robustness
- …

Process: to ensure we get a good quality software system
- requirements elicitation
- design
- implementation
- testing
- deployment

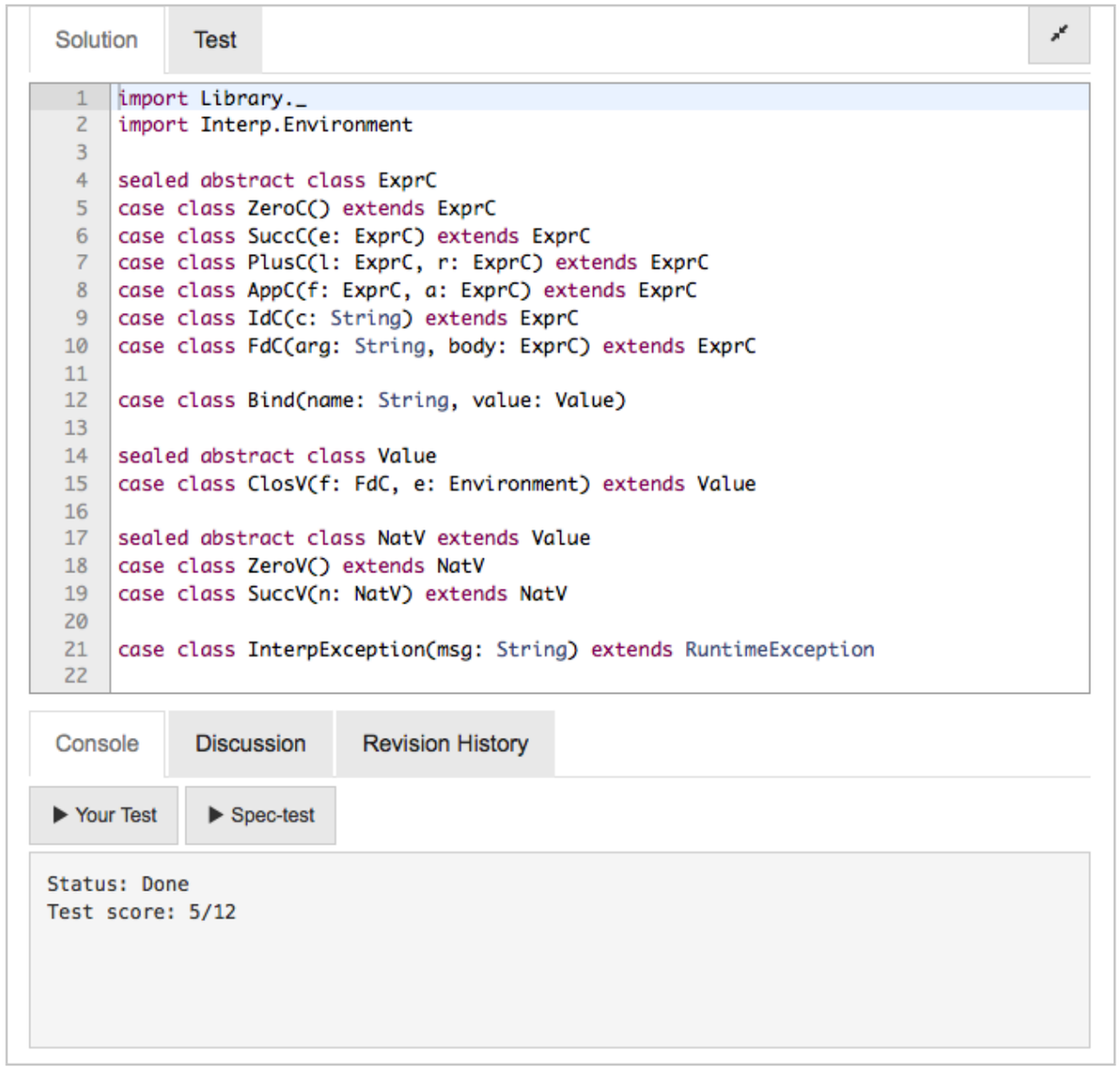
Example: WebLab

functional requirements

- view, edit assignments
- compute grades
- execute student programs
- design course
- navigate course
- …

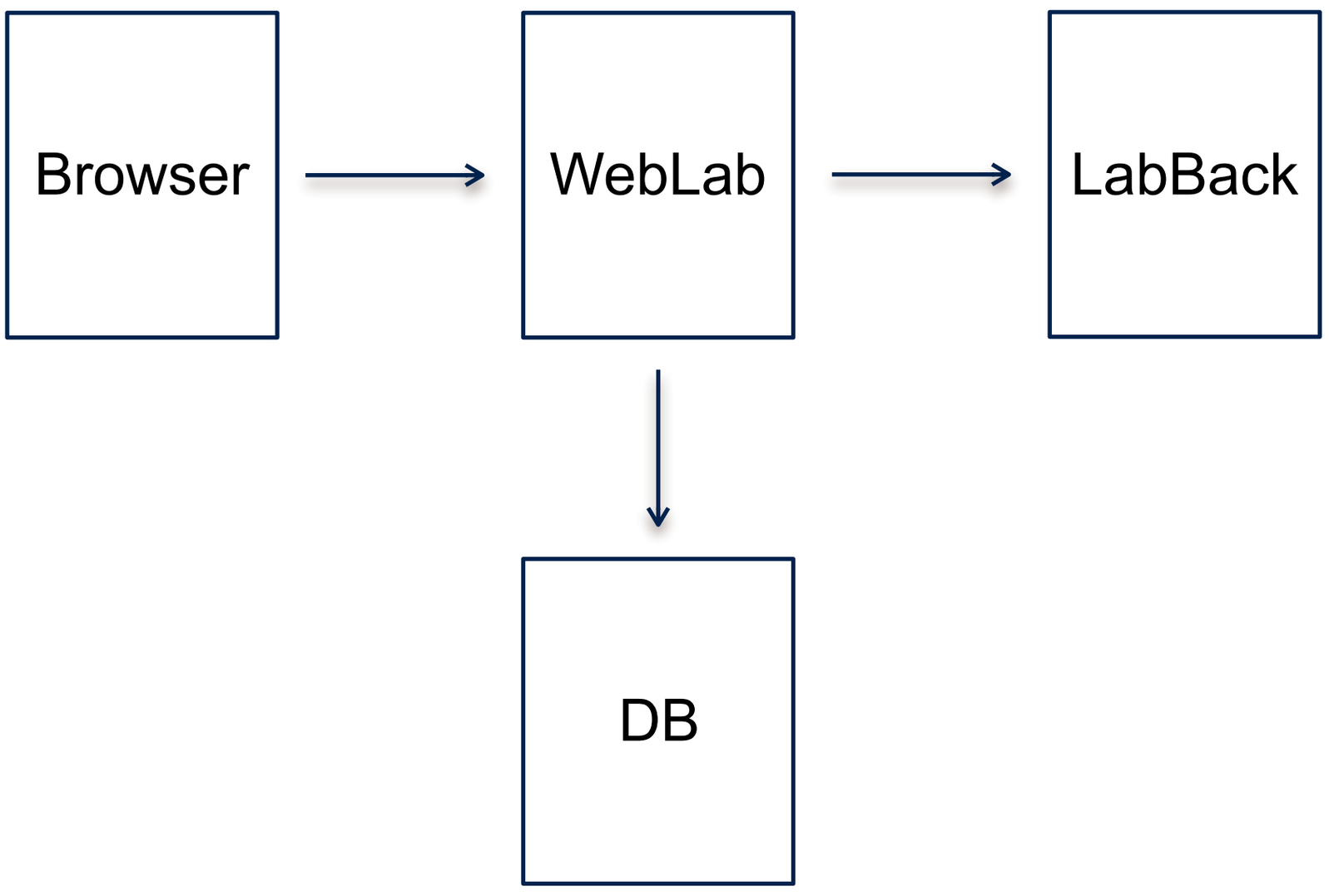
non-functional requirements

- response time
- …



TU Delft

WebLab Architecture



What role does security play
in the design of WebLab?

Understanding Software Security

Software security != patching security bugs

We need a framework for reasoning about software security

- What is the goal of software security?
- When is a software system secure?
- What does a secure system (not) do?
- How can we verify that a software system is secure?
- How can we integrate security in the software engineering process?

TU Delft

Security is (subset of) Reliability

- Software security is not a feature
- Security is preventing bad things from happening
- Cannot observe security
- Can observe lack of security … when it goes wrong
- “The objective of a secure system is to prevent all unauthorized use of information, a negative kind of requirement. It is hard to prove that this negative requirement has been achieved, for one must demonstrate that every possible threat has been anticipated.” Saltzer & Schroeder

TU Delft

SECURITY REQUIREMENTS

TU Delft

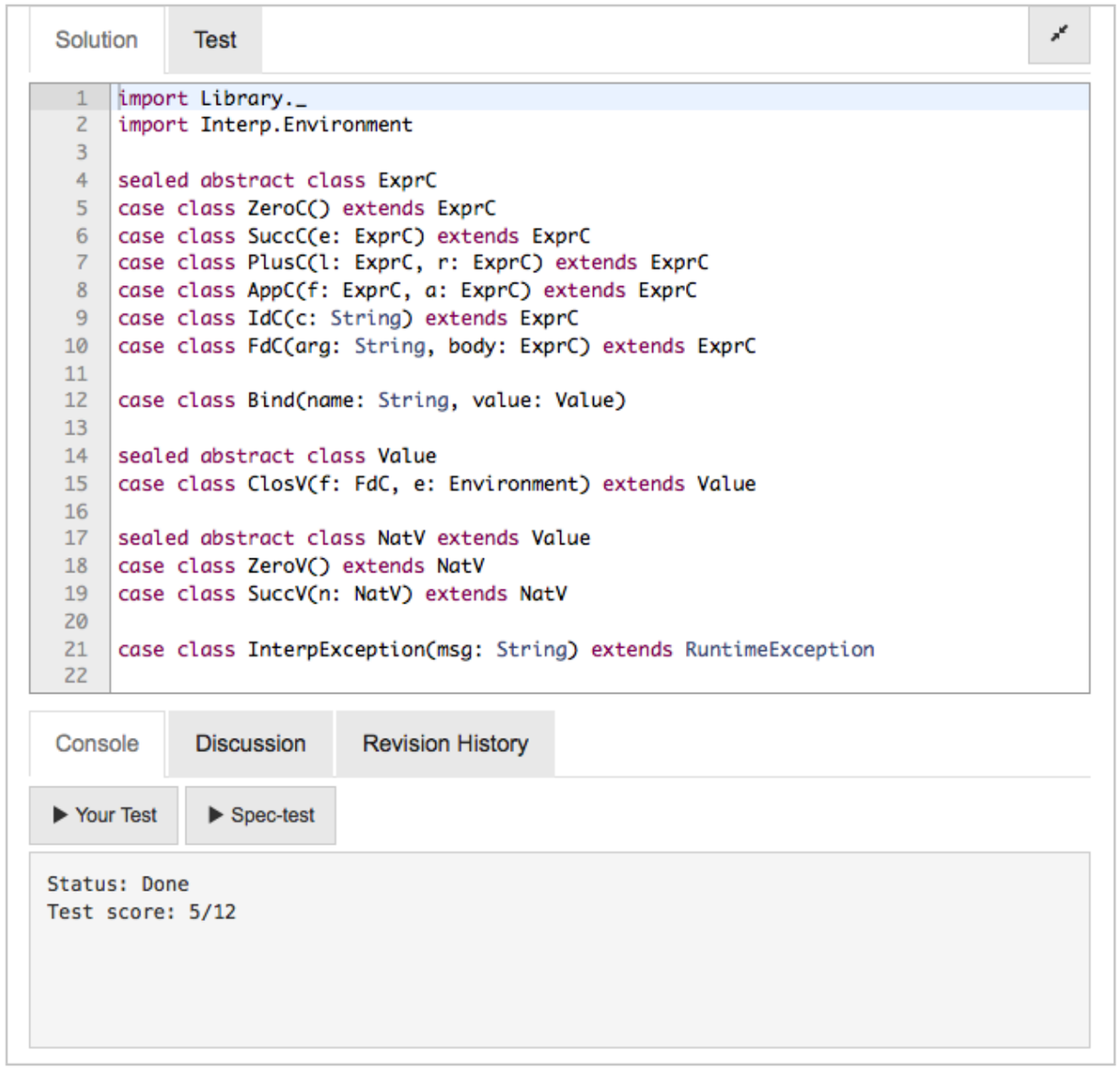
Security is Protection of Assets

- “The term “security” describes techniques that control who may use or modify the computer or the information contained in it.“ *Saltzer & Schroeder*
- “Computer security is the protection of items you value, called the assets of a computer or computer system” *Pfleeger & Pfleeger*
- To determine what to protect, we must first identify what has value and to whom

TU Delft

WebLab Assets

- Assignments: lab assignments, exam assignments
- Reference solutions to assignments (model answers)
- Student submissions to assignments
- Student grades
- Account data, especially passwords



What are the assets in a learning management system?

Security Requirements (CIA)

What security properties should a software system have?

- Confidentiality
- Integrity
- Availability
- Privacy
- Nonrepudiation

TU Delft

Confidentiality

- The ability of a system to ensure that an asset is viewed only by authorized parties
- Sensitive information is not leaked to unauthorized parties
- Privacy for individuals, confidentiality for data

Examples

- Confidentiality: exam assignments should not be published (at least not until the exam is over)
- Privacy: grades should only be visible to instructor and student involved

Anonymity

- Specific kind of privacy

Example

- Find out what kind of courses are available without being tracked by job advertisers / universities

TU Delft

Integrity

- The ability of a system to ensure that an asset is modified only by authorized parties
- Sensitive information is not damaged by unauthorized parties

Examples

- Submissions are not edited by anyone other than student
- Submissions are not edited by anyone after the deadline
- Grades are determined only by instructor or auto-grader
- Assignments, submissions, grades are not removed / only by instructor

TU Delft

Availability

- The ability of a system to ensure that an asset can be used by any authorized parties
- A system is responsive to requests

Example

- Should execute program tests fast (in reasonable time)
- Should be responsive when editing text/code

Nonrepudiation / Accountability

- The ability of a system to confirm that a sender cannot convincingly deny having sent something
- Note: opposite of privacy/anonymity; requires a balance

Examples

- Student cannot deny to have edited submission after the deadline

TU Delft

SECURITY MECHANISMS

TU Delft

Security Mechanisms

Provided by a system to enforce its security requirements

- Authentication
- Authorization
- Audit

TU Delft

Authentication

- “verify the identity of a person (or other external agent) making a request of a computer system”
- The ability of a system to confirm the identity of a sender
- Determine subject of security policy
- Principal: an entity that can be authenticated
- Authentication factors
  - something the user knows; password
  - something the user has; smart phone, card
  - something the user is; fingerprint
  - *multi-factor* authentication uses several of these

TU Delft

Authorization

- Defines when a principal may perform an action

Examples

- WebLab roles
  - Manager may edit assignments
  - Grader can view and grade submissions
- Student can edit own submissions
- Student can view exam assignments when signed into the exam and the exam has started (and not ended)

TU Delft

Auditing

- The ability of a system to trace all actions related to a given asset
- Retain enough information to be able to determine the circumstances of a breach or misbehavior (or establish one did not occur)
- Such information, often stored in log files, must be protected from tampering, and from access that might violate other policies

Examples

- Maintain versions of edits to program submission
- Maintain access log

TU Delft

Summary: Security Design

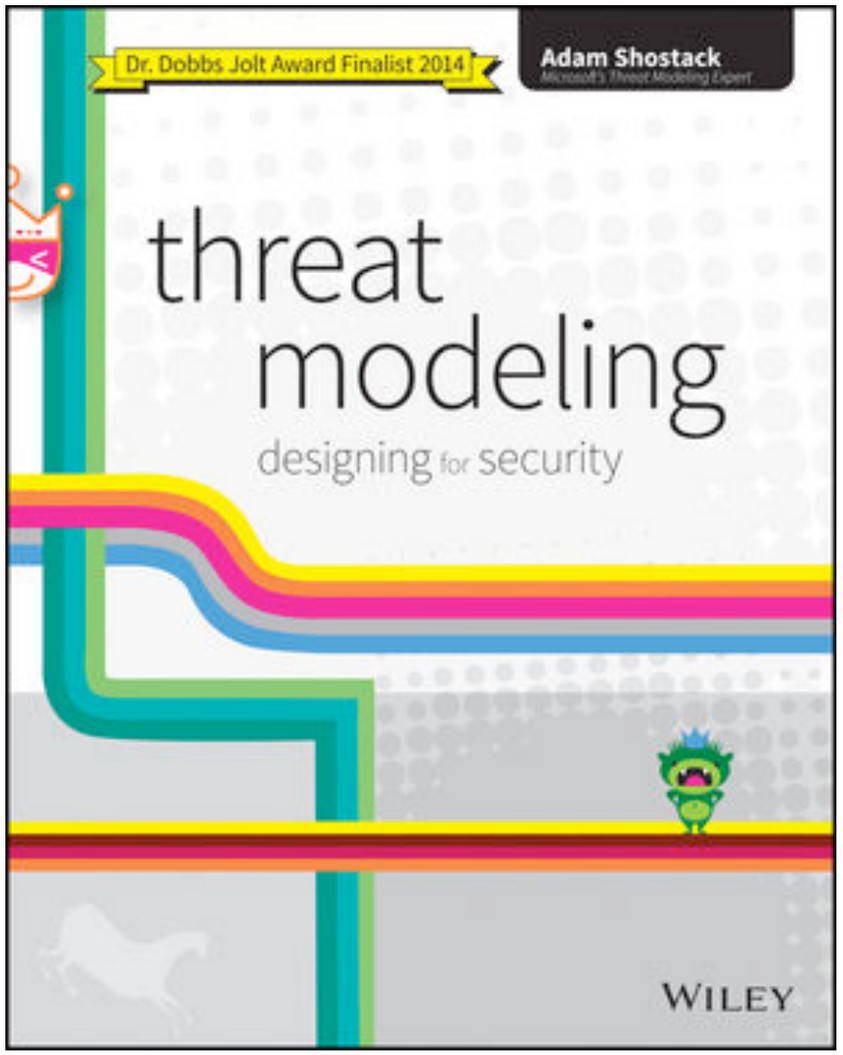
Specify functional requirements
- use cases

Specify security requirements
- what are the assets to protect?
- confidentiality
- integrity
- availability
- which is more important?

Define security mechanisms
- authentication
- authorization
- auditing

How do you know these are adequate?
- assurance

THREAT MODELING



TU Delft

Threats

- What can attackers do that violates these properties?
- How do these threats violate security requirements?

TU Delft

Trust Boundaries

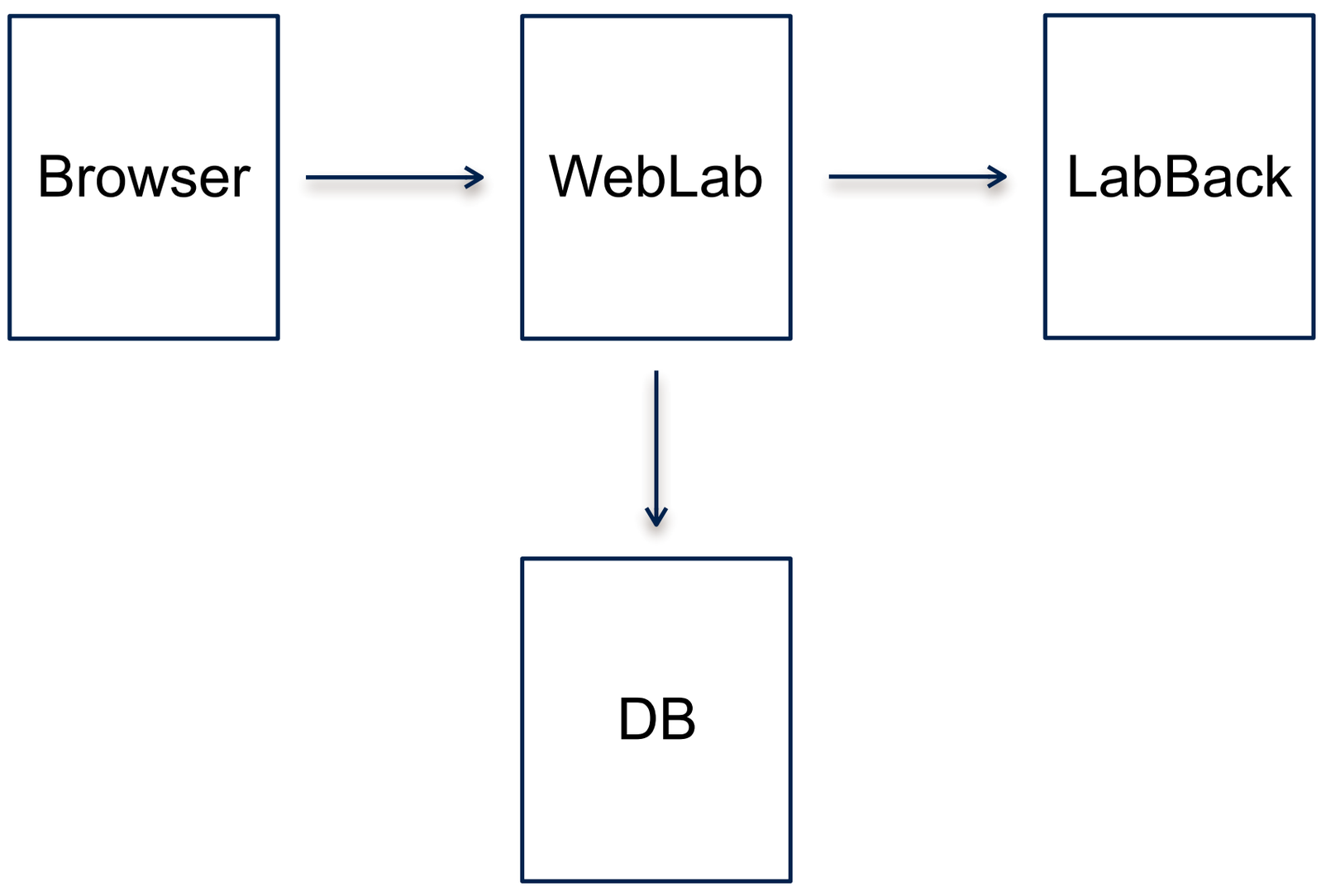
- Who controls what?
- Data-flow diagram
- Boxes around components indicate trust boundaries

Example

- Trust boundaries for WebLab

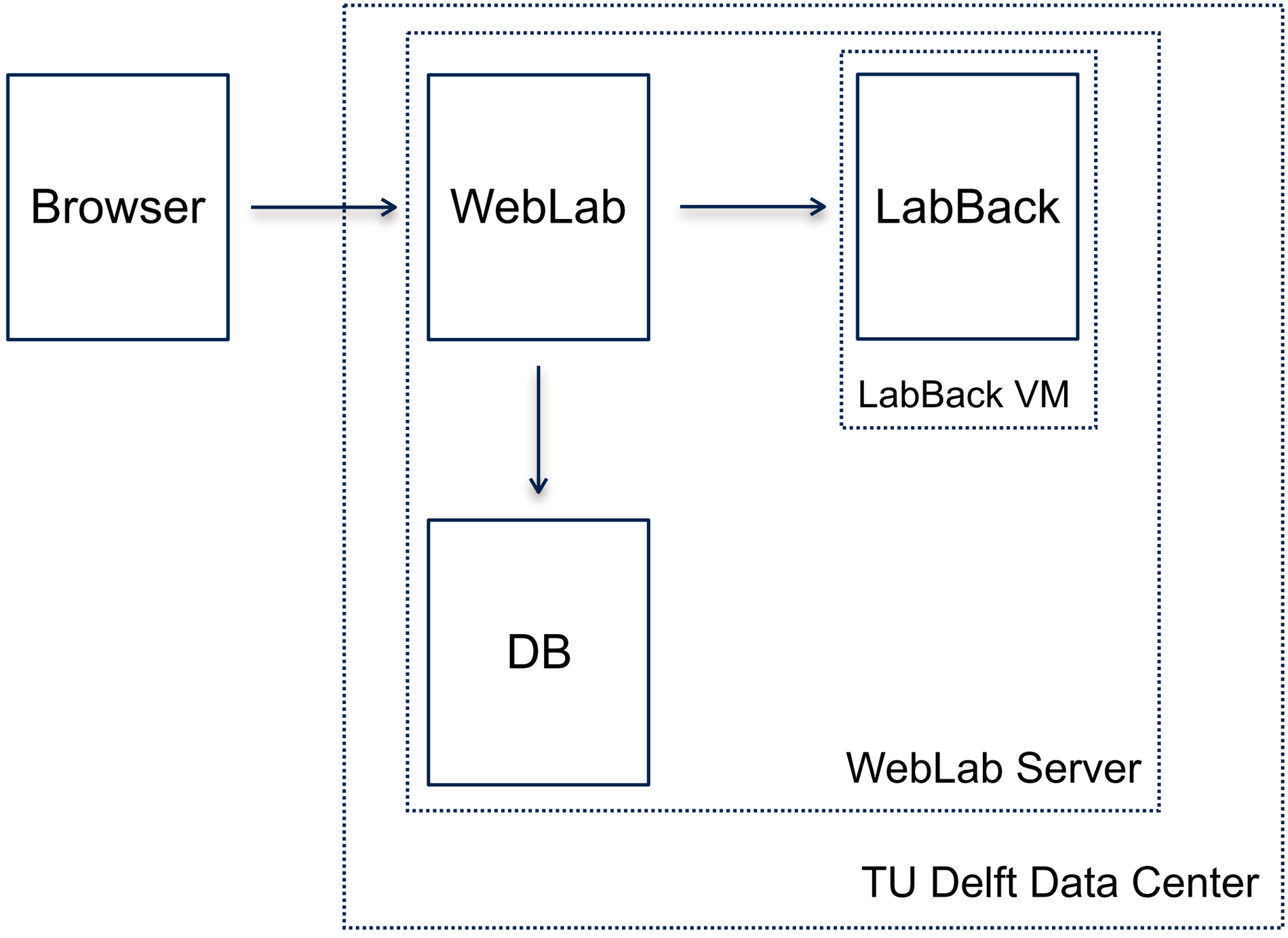
TU Delft

WebLab Architecture



Who can access / control
which components?

WebLab Trust Boundaries



TU Delft

Abuse Cases

- Opposite of use cases
- Illustrate security requirements
- Identify things system should *not* do

Example use case

- Student logs in and edits submission

Example abuse cases

- Student edits submission after the deadline
- Student changes grade of own submission
- Student executes program that modifies/destroys database

TU Delft

Finding Threats

How to identify threats for you system?

- Think like an attacker
- Taxonomy of threats
  - Kohnfelder & Garg. The threats to our products. MSDN 1999 (STRIDE)
- Attack trees
- Attack libraries

STRIDE

- **S**poofing
- **T**ampering
- **R**epudiation
- **I**nformation disclosure
- **D**enial of service
- **E**levation of privilege

TU Delft

Spoofing

Violates

- Authentication

Definition

- Pretending to be something or someone other than yourself

Abuse case

- Pretending to be course manager

TU Delft

Tampering

Violates

- Integrity

Definition

- Modifying something on disk, on a network, or in memory

Abuse cases

- Changing grade for a submission
- Change code of a submission
- Delete a submission, assignment, course

TU Delft

Repudiation

Violates

- non-repudiation

Definition

- Claiming that you didn't do something, or were not responsible

Abuse cases

- student claiming that they did not
  - copy solution
  - change grade
  - etc.

TUDelft

Information disclosure

Violates

- Confidentiality

Definition

- Providing information to someone not authorized to see it

Abuse cases

- Publishing answer to assignment
- Publishing exam questions before the exam

TU Delft

Denial of service

Violates

- Availability

Definition

- Absorbing resources needed to provide service

Abuse cases

- make system do many things: request flooding
- give system a big job: submitting program that runs a long time and consumes a lot of resources
- make system: submitting program solution that crashes the server

TU Delft

Elevation of privilege

Violates

- Authorization (integrity of security mechanism)

Definition

- Allowing someone to do something they're not authorized to do

Examples

- regular user executes code / database queries

Abuse cases

- missing / incomplete authentication
- missing authorization checks

TU Delft

Attack Trees

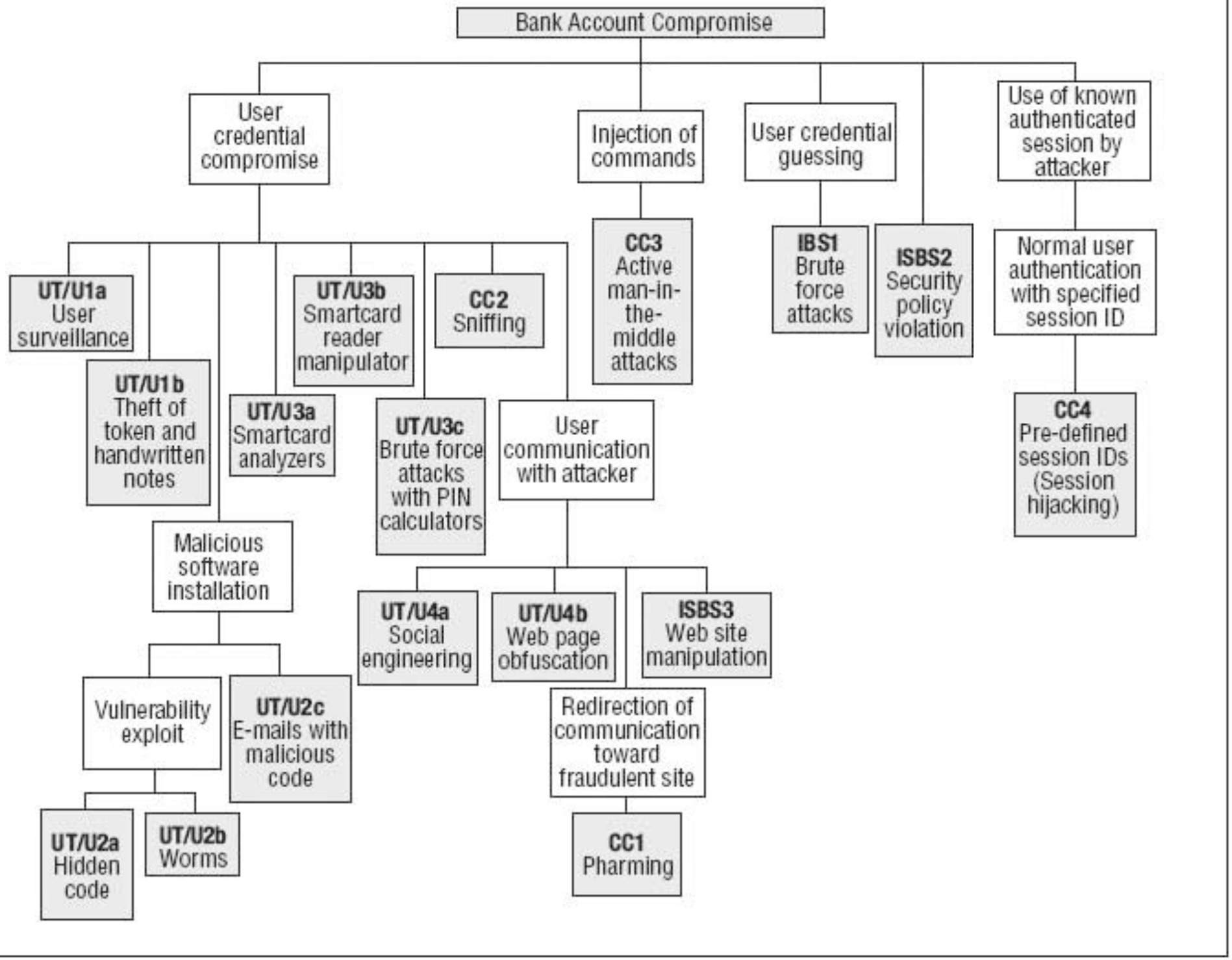
“Attack trees provide a formal, methodical way of describing the security of systems, based on varying attacks. Basically, you represent attacks against a system in a tree structure, with the goal as the root node and different ways of achieving that goal as leaf nodes” (Schneier, 1999)

Using attack trees

- Use existing attack tree to generate threats / abuse cases for your system
- Construct new attack tree to systematically enumerate possible attacks for your system
- Create attack tree for a class of systems (e.g. attacks for learning management systems)

TU Delft

Figure 1—Attack Tree



Attack Libraries

TU Delft

PRINCIPLES OF SECURE SOFTWARE DESIGN

Read:

Saltzer & Schroeder (1975)
The Protection of Information in Computer Systems
http://web.mit.edu/Saltzer/www/publications/protection/

Chapter 5 in Viega & McGraw:
Guiding Principles for Software Security

TU Delft

Categories

Prevention

- eliminate software defects entirely

Mitigation

- reduce harm from exploitation of unknown defects

Detection and recovery

- identify and understand an attack and undo damage

TU Delft

Design Principles (Saltzer & Schroeder)

- Economy of mechanism
- Fail-safe defaults
- Complete mediation
- Open design
- Separation of privilege
- Least privilege
- Least common mechanism
- Psychological acceptability
- Work factor
- Compromise recording
- Defense in depth
- …

TU Delft

Summary

- Software security is a subset of software **reliability**

- Security is about **protection of assets**
  - specified in terms of security requirements
  - confidentiality, integrity, availability, accountability

- Security is realized through **security mechanisms**
  - authentication, authorization, auditing

- **Threat modeling** used to identify threats against security
  - trust boundaries, attack surface
  - attack taxonomies (STRIDE), attack trees, attack libraries

- **Principles** of secure software design
  - best practices to avoid known pitfalls

NEXT

TU Delft

Lectures

- Lecture 1: What is Software Security? (Nov 11)
  - Eelco Visser in Delft
- Lecture 2: Memory-Based Attacks (Nov 18)
  - Sandro Etalle in Twente
- Lecture 3: Language-Based Security (Nov 25)
  - Eelco Visser in Delft
- Lecture 4: Vulnerabilities in Web Applications (Dec 2)
  - Sandro Etalle in Twente
- Lecture 5: Language-based Security for the Web (Dec 9)
  - Danny Groenewegen, Mark Jansen in Delft
- Lecture 6: Information Flow and Access Control (Dec 16)
  - Eelco Visser in Delft
- Lecture 7: Security Testing (Jan 6)
  - Eelco Visser in Delft

TU Delft

Assignment D
Security Design and Analysis

D1: Threat Modeling

- Select an existing software system or imagine one
- Describe its functional design using standard modeling techniques
  - class diagrams
  - data-flow diagrams
  - use cases
- Apply threat modeling to the design
  - abuses cases
  - attack trees

D2: Threat Model Peer Review

D3: Designing Security Policies

- Formulate a security design, including authentication, authorization, and auditing policies for the D1 system and argue why your design is safe

D4: Security Policies Peer Review

TU Delft

Assignment I: Security Bugs and Language-Based Security

I1: Buffer Overflows

- Construct an attack by exploiting a buffer overflow vulnerability

I2: Safety by Construction

- Implement a translation from a high-level language to a low-level language that ensures safety properties

I3: Web Security

- Implement a small web application with vanilla use of a web programming language / framework
- Examine security vulnerabilities in the result
- What do you need to do to prevent these bugs?
- Examine the counter measures in a WebDSL implementation of the same application

TU Delft

The Secret Life of Mobile Applications
Julia Rubin (MIT)

Wed, Nov 18, 2015 at 9:15 | TU Delft, EWI, Chip

Mobile applications have access to and leak user-sensitive information, such as device id, location and the user's email address. Several static and dynamic program analysis techniques were recently proposed to identify such information leakages. Yet, distinguishing between "acceptable" and "unacceptable" leakages is still a tedious manual task: an application could send the user's location to its own server in order to provide accurate navigation instructions, which would likely be acceptable by the majority of users. Shadowing the same information to a third-party server without any effect on the observable application behavior would likely be unacceptable.

In this talk, I will present a set of techniques that attempt to automate the distinction between "acceptable" and "unacceptable" leakages. Specifically, I will describe our approach for detecting communication that does not affect the delivered application experience. As the vast majority of mobile applications are user-centric, one would expect that such communication is rare. To our surprise, that was not the case in many highly popular Android applications from Google Play.

TU Delft