DD2460 Software Safety and Security

Introductory Lecture

Dilian Gurov

2. Software Safety and Security

· Software safety:

will not damage people or other systems
will not fail

· Software security:

- protected against malicious attack
- data integrity: only authorized access
- data confidentiality: does not leak information

Lecture Outline

- 1. The team
- 2. Introduction to the course
- 3. Course syllabus
- 4. Course objectives
- 5. Course organization

Software Safety

- A software specification error or design flaw can contribute to or cause a system failure or erroneous human decision
- Example areas:
 - embedded devices like a pacemaker or automotive brake control
 - software controlling nuclear power plants or aerospace rockets

• In summary: no illicit or undesired behaviour

1. Team

- First lecturer:
 - E-mail: – Phone:
- dilian@csc.kth.se 08-790 81 98 (office)

Dilian Gurov

- Office:
- E-bng, floor 4, room 4417
- Second lecturer: Gurvan Le Guernic - E-mail: gurvan@kth.se
- Course assistant: Musard Balliu
 E-mail: musard@kth.se

Software Security

- Engineering software so that it continues to function correctly under malicious attack
- Typical vulnarabilities:
 - memory leaks
 - buffer overflows
- Vulnarabilities give rise to threats

Information Security

- Network security:

 cryptographic protocols, firewalls, intrusion detection
- Secure information flow:
 confidentiality, integrity
- Access control:
 delegation, authorization, trust management

Why Formal Methods?

- Only formal methods can capture correctness *precisely*. Basis for *tools*.
- But: formal techniques are expensive
- Most needed for:
 - safety-critical systems
- commercially-critical systems (security)• Most succesful for: "small" systems
 - embedded systems
 communication protocols
 - on protocols

10

11

Language-based Security

- Application-level attacks:
 - Trojan horses, worms, buffer overrun attacks, exploit attacks, covert channels, and malicious code
- Language-based protection mechanisms:
 - static security analysis
 - program transformation
 - stack inspection

Formal Verification

- Various techniques
- Ingredients:
 - Property class
 - Modelling language
 - Property specification language
 - Verification method (decidability, scalability)
 - Tool support (degree of automation)

Formal Analysis

· Formal methods:

collection of formal notations and techniques (i.e. based on discrete mathematics and mathematical logic) for modelling and analysis of program behaviour.

• Common goal: The design of *correct* systems.

9

3. Course Syllabus

- We study three fundamental **techniques** for the analysis of programs, with focus on safety and security.
- The techniques are based on **types** and **logics** for programs, and allow to discover certain types of illicit behaviour or deduce the absence of such behaviour.
- We consider three successful **tools** implementing such techniques.

Part I. Temporal Logic and Model Checking

Props:Safety of state sequencesModels:Kripke structures, ProMeLaSpecs:Temporal logic formulas (LTL)Method:Model checkingTool:SPIN

13

14

15

4. Course Objectives

- **Aim**: provide working familiarity with three methods and tools for the analysis of safety and security of software, in theory and in practice.
- Grading: to pass the course, a student has to demonstrate the ability to apply the methods discussed in the course; for the highest grades he/she has also to be proficient in the theoretical underpinnings of these methods.

16

17

Part II. Hoare Logic and Program Verification

Props:	Safety of data manipulation
Models:	Source code (Java) (op. sem.)
Specs:	Hoare logic assertions (JML)
Method:	VCG, Symbolic execution
Tool:	VeriFast

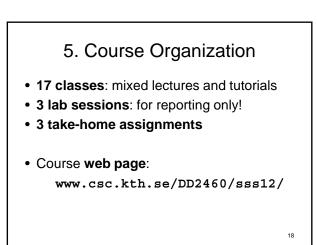
Intended Learning Outcomes

After the course, you should be able to:

- 1. Identify, specify and verify important safety and security properties using suitable automated tools.
- 2. Explain the underlying techniques and be able to argue for their correctness and limitations.
- 3. Correctly interpret and evaluate the results of the analysis.

Part III. Information Flow Analysis

Props:	Confidentiality + integrity of data
Models:	Source code (Java)
Specs:	Security levels
Method:	Type checking
Tool:	Jif



Course Literature

- Course book:
 "Logic in Computer Science"
 by Huth and Ryan (see Kårbokhandeln)
- Additional material: on the web page don't print without need!

Lab Reports

- All three lab assignments will be presented at dedicated lab sessions (in lab room Orange) on the basis of a written report
- The labs will be graded F-C based on – quality of work
 - quality of report

Auctioning System

- All three lab assignments are based on the same software system that you will develop: an **auctioning system** written in Java
- The three labs will analyse three different aspects of the system or selected components

Tools

- Three tools are installed on the Ubuntu machines in the lab rooms (Orange):
 - SPIN: ispin
 - VeriFast: vfide - Jif: jif

Lab Assignments

• SPIN lab:

safety of synchronization behaviour

- VeriFast lab: safety of shared data manipulation
- Jif lab: confidentiality and integrity of private data

21

19

20

Take-home Assignments

- Three take-home assignements will examine your understanding of the three analysis techniques and the theoretical underpinnings of the tools
- The assignments will be graded F-A

24

22

23