frama C

Software Analyzers

Teaching Frama-C for Cybersecurity

Julien Signoles julien.signoles@cea.fr

23th May 2026 @ FMTea'26

CEA-List, Université Paris-Saclay, Software Safety and Security Lab

funded by the French National Research Agency through Project CMA - TCE (Train Cyber Experts)

cea list

université PARIS-SACLAY

FRANCE 2030

RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

anr agence nationale de la recherche

Outline

Duration and Places

- single 3-hour lecture and single 3-hour lab session
- shorter version available:
  - 90-minute lecture and 90-minute lab session
  - 1- or 2-hour lecture, without lab session
- created Q4 2022 for Telecom SudParis (engineering school), since:
  - Telecom SudParis (engineering school), 2022-2025
  - CentraleSupelec Rennes (engineering school), 2023-2025
  - Université Bretagne Sud (international master students), 2023-2025
  - École des Mines Paris (engineering school), 2025

Audience and Goals

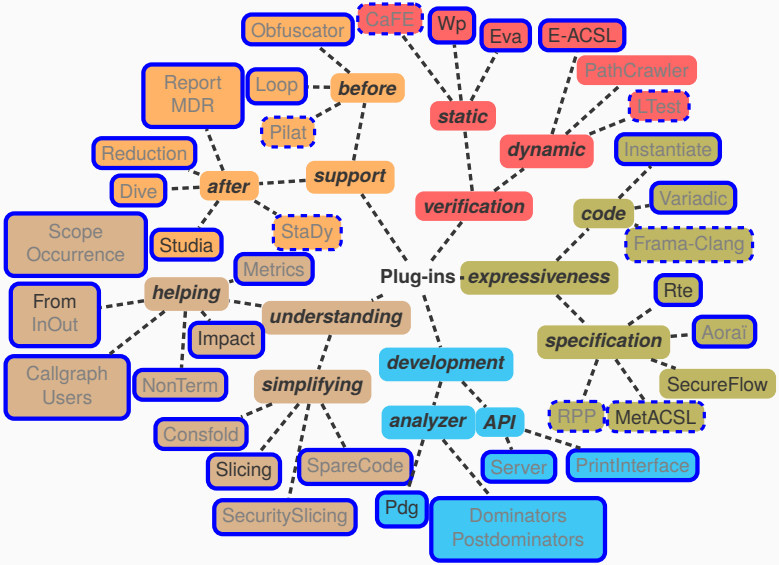
- target audience:
  - students at master level
  - embedded system or cybersecurity track
  - no knowledge in formal methods needed
- requirement: reading C code (bachelor level)
  - knowledge of integer and pointer arithmetic, dynamic allocation, etc
- goals:
  - introduce formal methods to students
  - emphasize practical aspects in cybersecurity
  - show what they are good, but do not hide practical difficulties

Framework for analyzing C source code

https://frama-c.com

- developed at CEA List since 2005: >20-year old
- open source (LGPL 2.1)
- based on formal methods
  - ACSL as formal specification language
- several code analysis tools provided as plug-ins
  - >30 plug-ins in the latest distribution
- plug-ins linked to the Frama-C kernel
  - uniform setting (shared user interface, etc)
  - information gathering (which analyzer proves which properties, etc)
  - analysis collaboration

Plug-in Gallery



distributed within Frama-C distribution | external, yet open-source | close-source

Outline

Lecture Outline

https://julien-signoles.fr/teaching/slides-frama-c-cyber.pdf

1 ⟩ Context
2 ⟩ Frama-C in a Nutshell
3 ⟩ Deductive Verification with ACSL and Wp
4 ⟩ Value Analysis with Eva
5 ⟩ Runtime Annotation Checking with E-ACSL
6 ⟩ Advanced Security Verification

breadth-based approach, instead of depth-based

- why formal methods are important:
  - for trusting software, particularly in critical systems
- why it is not easy: Rice theorem

  a sound, automatic, exact and sound analysis is not possible (undecidability)
- how to be pragmatic: different formal techniques, none being perfect
  - runtime verification (not static)
  - model checking (does not always scale)
  - abstract interpretation (not exact)
  - deductive verification (not automatic)

Structure of the Lecture

Similar Outline for every technique presented:

1 ⟩ general overview of the technique: no maths!

2 ⟩ examples of basic usage and demo on open-source code

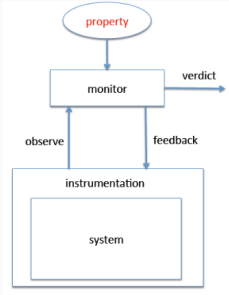
- live session on my laptop

3 ⟩ show practical difficulties and limitations

4 ⟩ industrial example

Example of Overview (1/2): Runtime Verification

- check whether a system under scrutiny satisfies a given property
- offline monitoring: after execution
- online monitoring: during execution
  - inline: the monitor is part of the system
  - outline: the monitor is outside the system
- most approaches focus on verifying temporal properties
- ... yet not E-ACSL



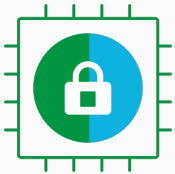
- E-ACSL is a Frama-C plug-in that generates inline monitors for (E-)ACSL properties on C programs [Signoles et al, 2017]
  - takes as input an (E-)ACSL-annotated C program
  - generates a new C program
  - that behaves as the original C program if all the annotations are valid; or
  - fails on the first invalid annotation (by default)

```
1  int div(int x, int y) {
2    /*@ assert y-1 != 0; */
3    return x / (y-1);
4  }
```

E-ACSL →

```
1  int div(int x, int y) {
2    /*@ assert y-1 != 0; */
3    e_acsl_assert(y-1 != 0L);
4    return x / (y-1);
5  }
```

Example of Demo: PolarSSL with Eva



PolarSSL

(now known as Mbed-TLS)

`https://git.trustedfirmware.org/mirror/mbed-tls.git/about/`

- C implementation of TLS (aka SSL)
- not as complex as openSSL

Showcasing Difficulties: What do you (not) prove with Wp?

- what do you prove?
  - the source code matches the spec
  - under the assumptions of your toolset
    - check the manuals, the emitted warnings, ...
- what do you **not** prove?
  - the executed code does what you have in mind
    - same mistake in the code and in the spec
    - missed assumptions
    - ... *we'll come back at the end*

Industrial Example: Common Criteria Certification

- Formal verification of a JavaCard Virtual Machine [Djoudi et al, 2021]
- Common Criteria's EAL7 certificate
- Example of properties
  - header integrity
    - allocated object's header cannot be modified during a run
  - data integrity
    - allocated object's data can be modified only by the owner
  - data confidentiality
    - allocated object's data can be read only by the owner
- generate $\approx 400,000$ ACSL annotations from ≈ 500 MetACSL properties
  - all proved with Wp

THALES

Lab Session (1/2)

https://julien-signoles.fr/teaching/tp-cyber/tp.md

- Frama-C in a VM
- based on Eva (value analysis by abstract interpretation)
- target: ciphering and deciphering functions of Bacon code
  - from *Rosetta Code* project
  - extended with a few (simple) vulnerabilities introduced
- technical goals:
  1. find all the vulnerabilities with Eva et fix them
  2. demonstrate there is no vulnerabilities anymore, i.e., 0 alarm
     - for a large class of input messages

Lab Session (2/2)

- principles:
  - let students work independently as much as possible
  - help them and provide explanation when necessary
- difficult to apply this principle for a lab session on automatic analysis
  - if the analysis is conclusive (no alarm or bug found), one has no clue why it concludes this way
  - otherwise, hard to understand why it is not able to conclude
- solution: step-by-step problem wording
  - preliminary exercise on a simple example from the Juliet test suite
  - slow development
  - many written explanations about the analyzer's output
- not (yet) solvable by any LLM
  - the students must often describe then explain what they see on the Frama-C GUI

Outline

Feedback: Lecture

- usually not a lot of feedback in French classes, yet a bit, e.g.:
  "The students said again at the review meeting that your class was great!"

  J.-F. Lalande, 15th April 2025 (private mail, translated from French)
- live examples and demos make the class more engaging
  - also help for time management, e.g. skipping a demo if late is easy
- industrial applications demonstrate usefulness of formal methods
- pointing out limitations makes the explanations more credible

Feedback: Lab Session

- they like working on a real open-source code
- good level of difficulty
  - not too easy, yet doable alone (if they take care to all the written advice)
  - fine-tuning over several years: added questions and explanations when necessary
- teacher's advice still needed from time to time
  - point out what is written in the text
  - explain some Frama-C outputs and/or behaviors
- unintented positive side-effects: improve their C programming knowledge a bit
  - computer arithmetics and integer overflows
  - dynamic arrays and pointer arithmetics
  - malloc may fail and return `null`
  - error handling
- some students "fix" code in the wrong way, leading to unremovable alarms
- sparse VM issue (VM size, no ARM support, etc)

Feedback: Short Version

- not as mature as the standard 3-hour version
- shortening the lesson was easy
  - same outline and global message
  - mainly less demos and examples (which take time)
  - yet message harder to deliver: less time to repeat the key insights regularly
- shortening the lab session was more difficult
  - remove preliminaries and move its relevant questions to the main exercice
  - repetitive questions have also been removed
  - now even a bit too easy for the best students
  - also lack repetition for good learning
  - critical to solve any VM issue very quickly

- 3-hour lecture and 3-hour lab session about Frama-C for cybersecurity
- practical introduction to formal methods and their applications to cybersecurity
- reusable material for teachers familiar with Frama-C
  - already successfully done once, without modifying the material
- adaptable to others formats
  - shorter lecture and shorter lab session
  - 2-hour lecture in a doctoral summer school
  - 1-hour seminar to master students

Thank You!

- Teachers who have invited me to teach Frama-C for cybersecurity:


O. Levillain
Telecom SP


J.-F. Lalande
CS Rennes


P. Wilke


S. Sadou
U. Bretagne Sud


G. Gogniat


O. Hermant
Mines Paris

- Original idea of the lab session:


V. Prevosto

- Some examples:


N. Kosmatov