# SECOMP

## Efficient Formally Secure Compilers to a Tagged Architecture

Cătălin Hriţcu

INRIA Paris

# Principal investigator: Cătălin Hriţcu

**[2005-2011]** MSc & PhD @ **Saarland University, Saarbrücken, Germany**
**[2011-2013]** Research Associate @ **University of Pennsylvania, USA**
**[2013-now]** Research Scientist @ **INRIA Paris, France**

- **Publications** (20+ papers, 500+ citations)

**Best venues in security** (2×Oakland S&P, CCS, 3×CSF, 2×JCS)

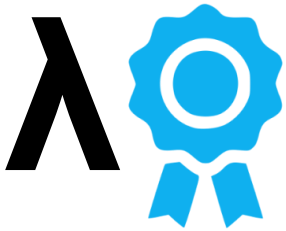
**and programming languages** (2×POPL, 2×ICFP, 2×JFP, ASPLOS, LMCS)

**Software Foundations** teaching programming languages & logic with Coq

- **Currently supervising 2 PhD and 3 MSc students**

- **General chair** of IEEE European Symposium on Security & Privacy 2017

- **PC member** for POPL 2017, CSF 2016, ITP 2016, CPP 2016 , POST 2017



# My Research

## Devising formal methods

- programming languages
- type systems, logics
- verification systems
- proof assistants
- property-based testing

## Solving security problems

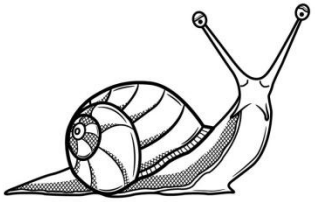
- formal attacker models
- provably secure systems
- stopping low-level attacks
- reference monitors
- security protocols

**Resulted in many innovative tools**

- Micro-Policies, F*, QuickChick, Luck, ...

# The problem: devastating low-level attacks

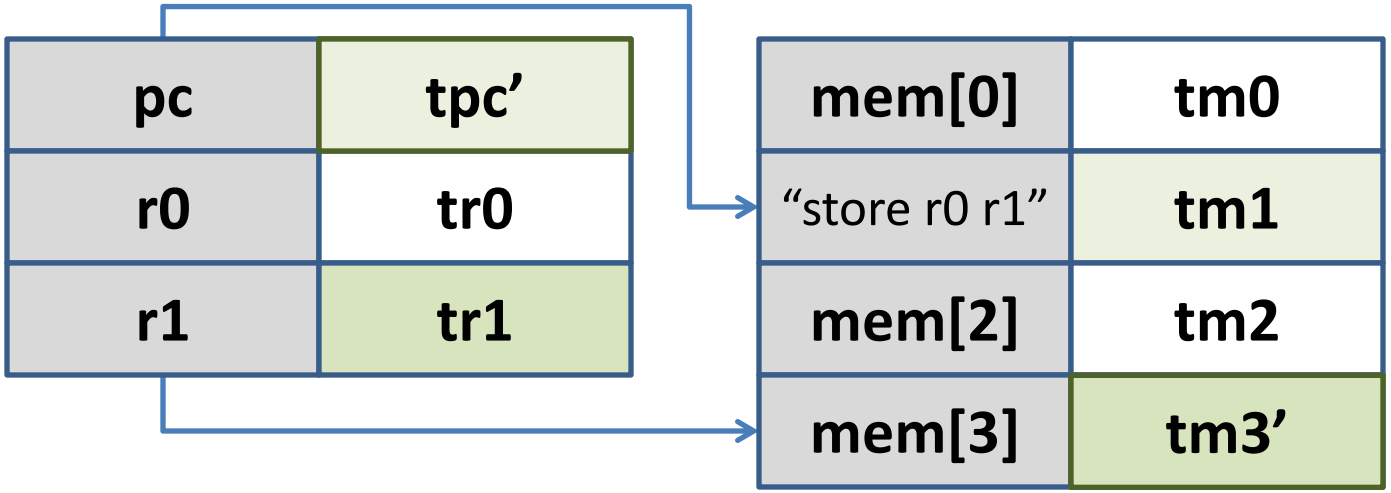
- **1. inherently insecure low-level languages (C, C++)**
  - **memory unsafe**: any buffer overflow can be catastrophic
    allowing remote attackers to gain complete control
- **2. unsafe interoperability with lower-level code**
  - even code written in **safer high-level languages (Java, C#, OCaml)** has to interoperate with **insecure low-level libraries (C, C++, ASM)**
  - **unsafe interoperability:** all high-level safety guarantees lost
- **Today's languages & compilers plagued by low-level attacks**
  - **main culprit: hardware provides no appropriate security mechanisms**
  - fixing this purely in software would be way **too inefficient**

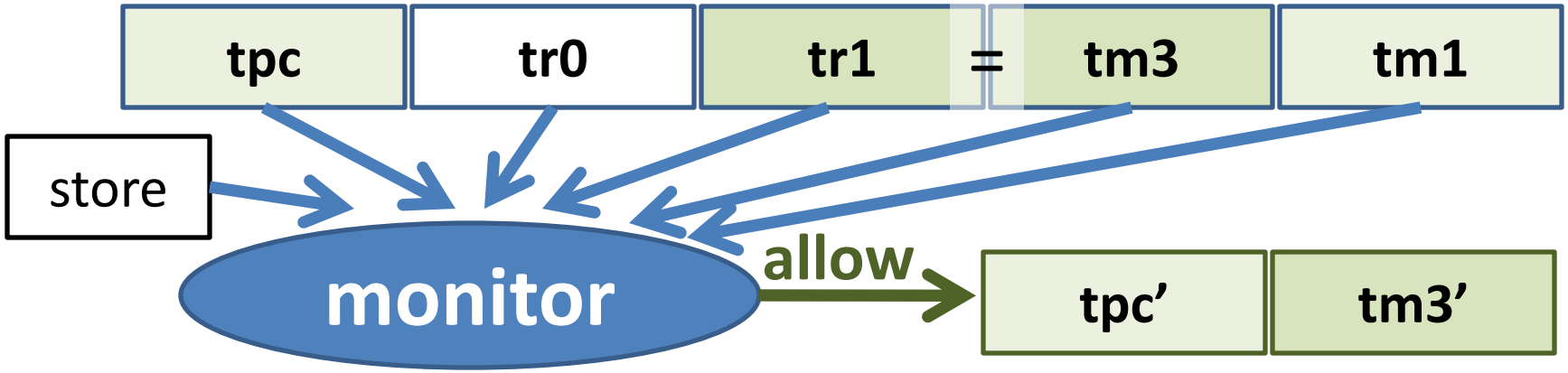


# Key enabler: Micro-Policies

[Oakland '13 & '15, POPL '14, ASPLOS '15]

software-defined, hardware-accelerated, tag-based monitoring



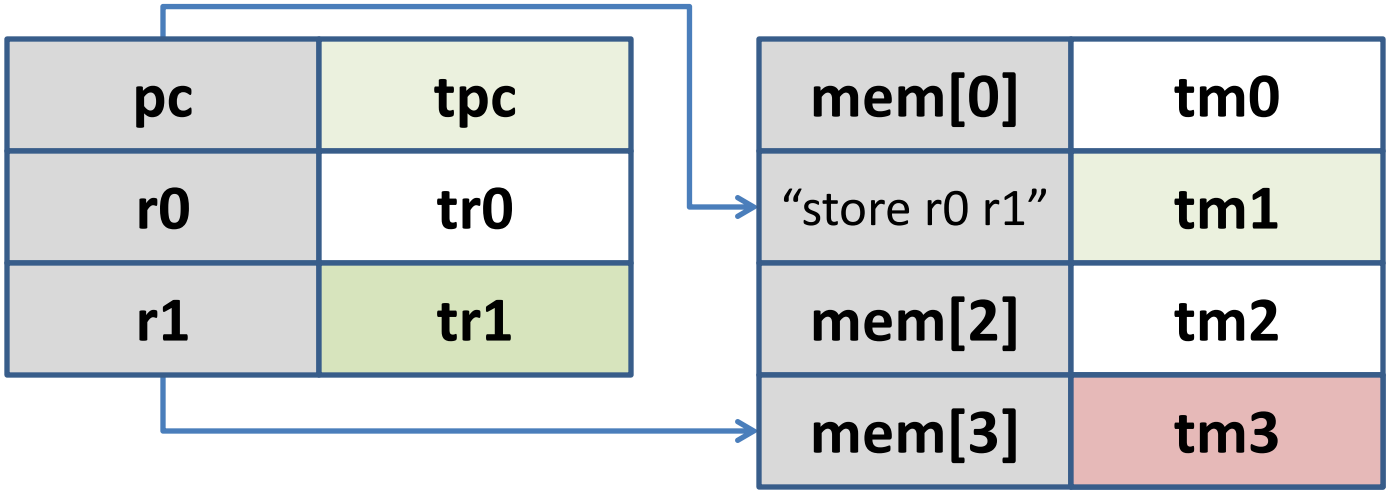



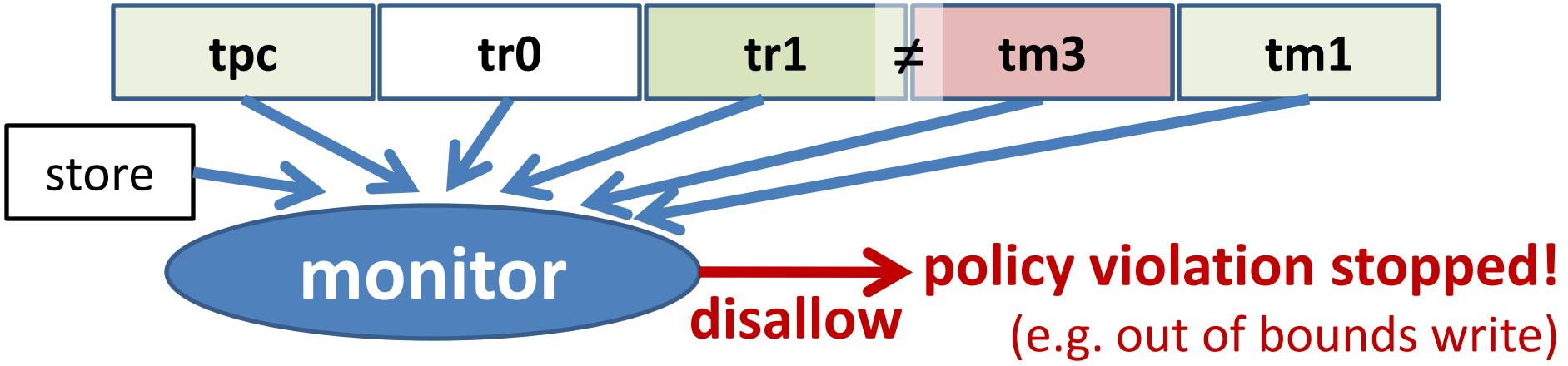

**software monitor's decision is hardware cached**

# Key enabler: Micro-Policies

[Oakland '13 & '15, POPL '14, ASPLOS '15]

software-defined, hardware-accelerated, tag-based monitoring



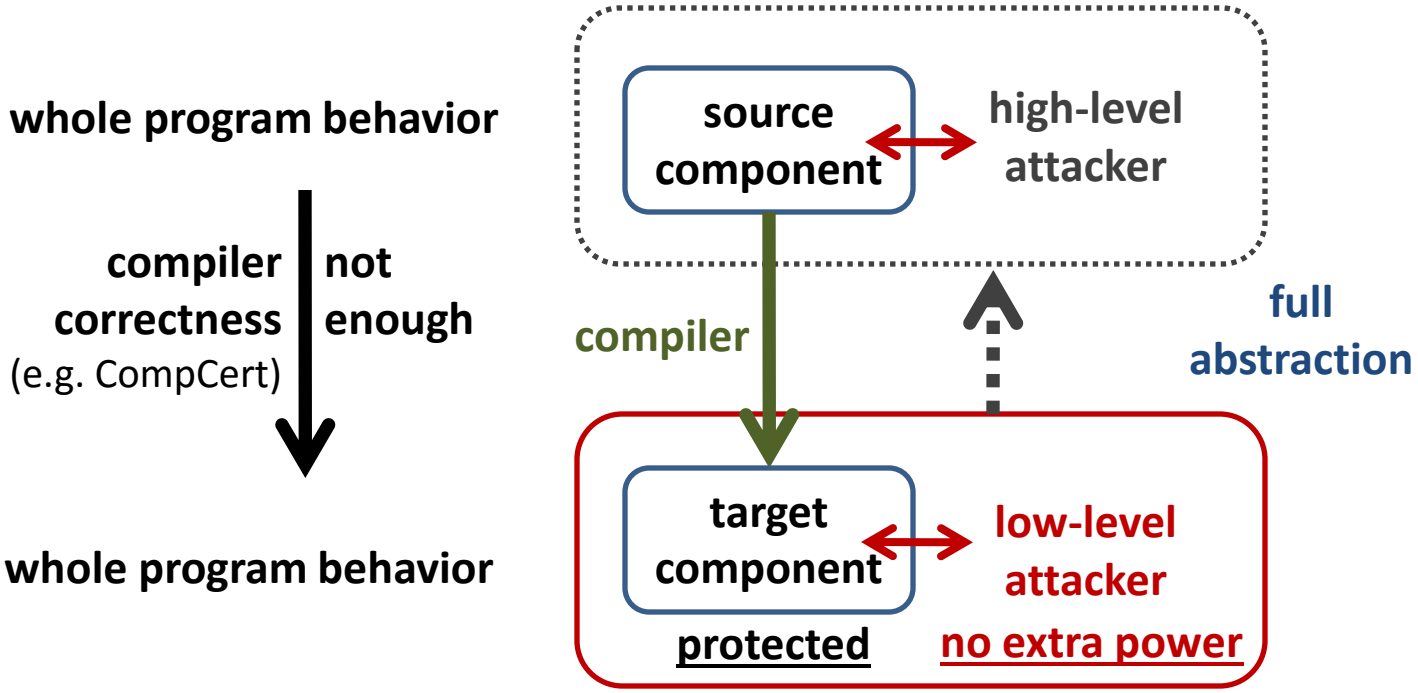
# SECOMP grand challenge

Build **the first efficient formally secure compilers** for **realistic programming languages**

1. **Provide secure semantics for low-level languages**
   - C with protected components and memory safety
2. **Enforce secure interoperability with lower-level code**
   - ASM, C, and F* [F* = ML + verification, POPL ’16]

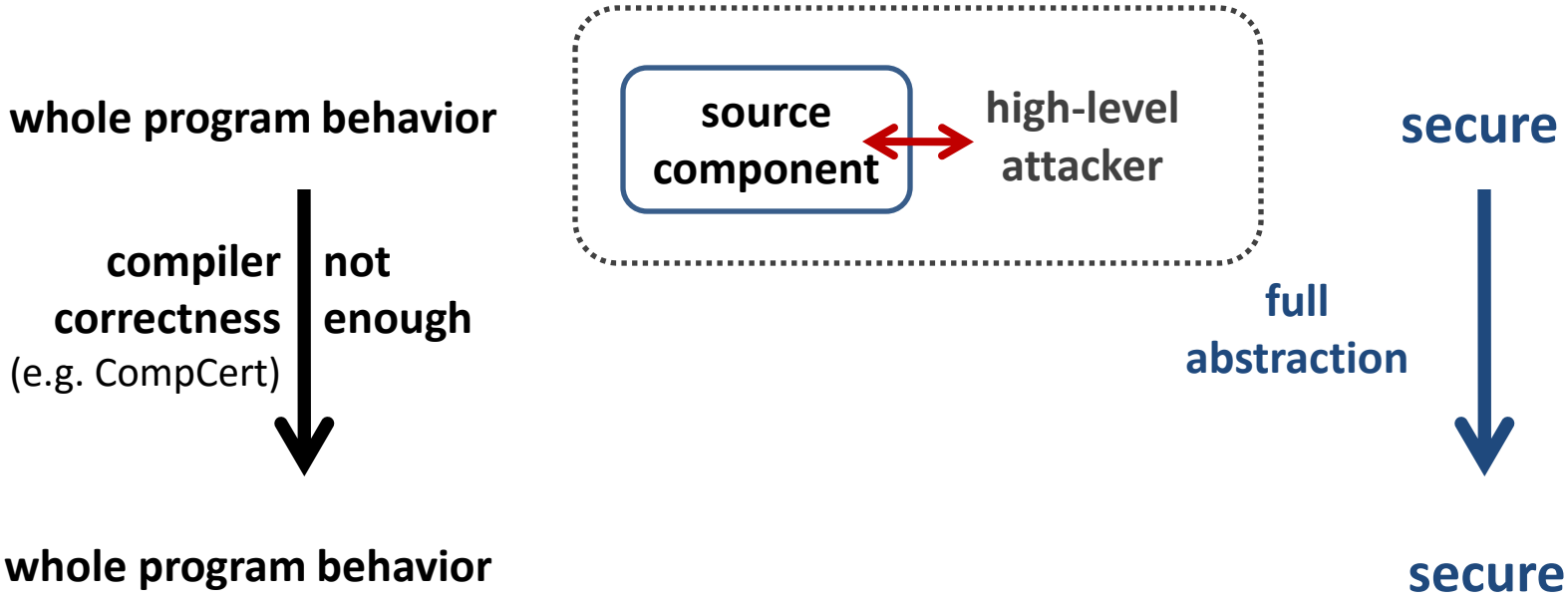
# Formally verify: full abstraction

holy grail of secure compilation, enforcing abstractions all the way down



# Formally verify: full abstraction

holy grail of secure compilation, enforcing abstractions all the way down




**Benefit**: **sound security reasoning in the source language**

forget about compiler chain (linker, loader, runtime system)

forget that libraries are written in a lower-level language

# SECOMP: achieving full abstraction at scale

**F* language**
(ML + verification)

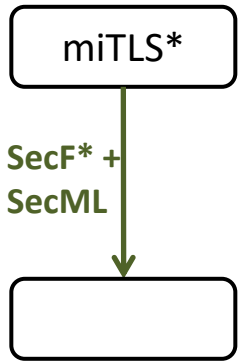
miTLS*

**C language**
+ memory safety
+ components

# SECOMP: achieving full abstraction at scale

**F* language**
(ML + verification)

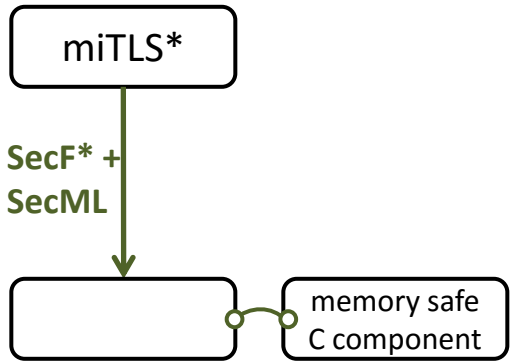
**C language**
+ memory safety
+ components



# SECOMP: achieving full abstraction at scale

**F* language**
(ML + verification)

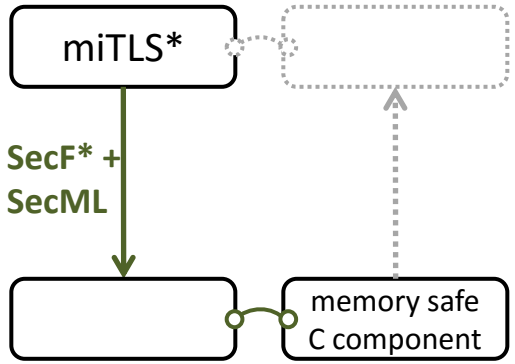
**C language**
+ memory safety
+ components



# SECOMP: achieving full abstraction at scale

**F* language**
(ML + verification)

**C language**
+ memory safety
+ components

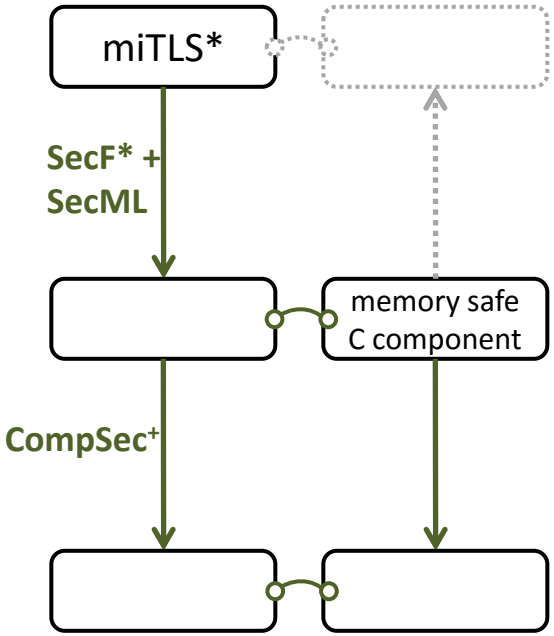


# SECOMP: achieving full abstraction at scale

**F* language**
(ML + verification)

**C language**
+ memory safety
+ components

**ASM language**
(RISC-V + micro-policies)

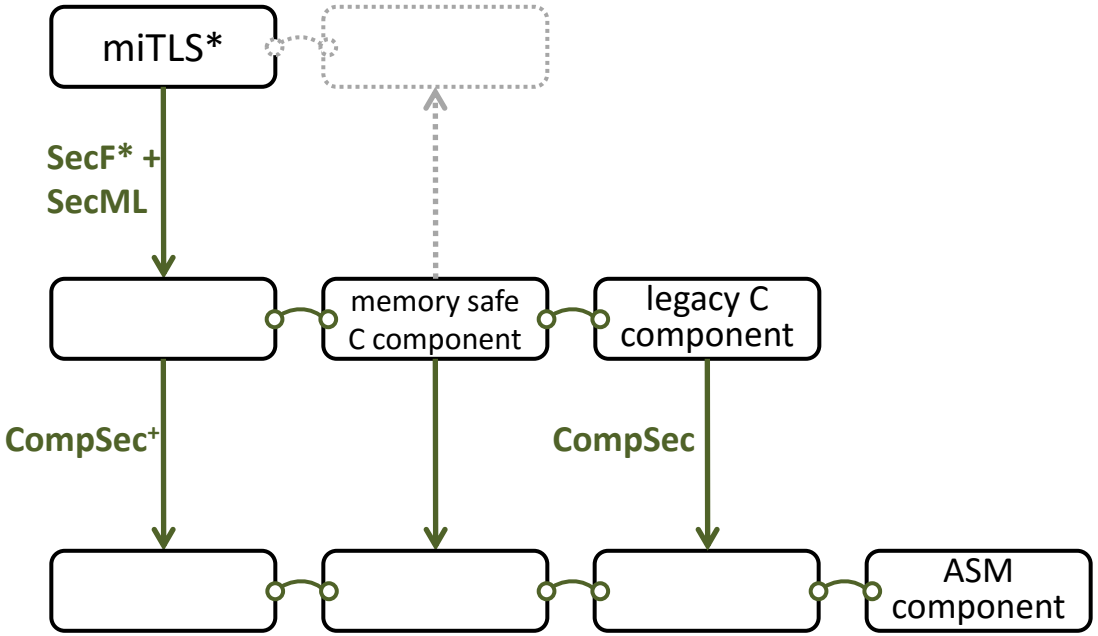


# SECOMP: achieving full abstraction at scale

**F* language**
(ML + verification)

**C language**
+ memory safety
+ components

**ASM language**
(RISC-V + micro-policies)

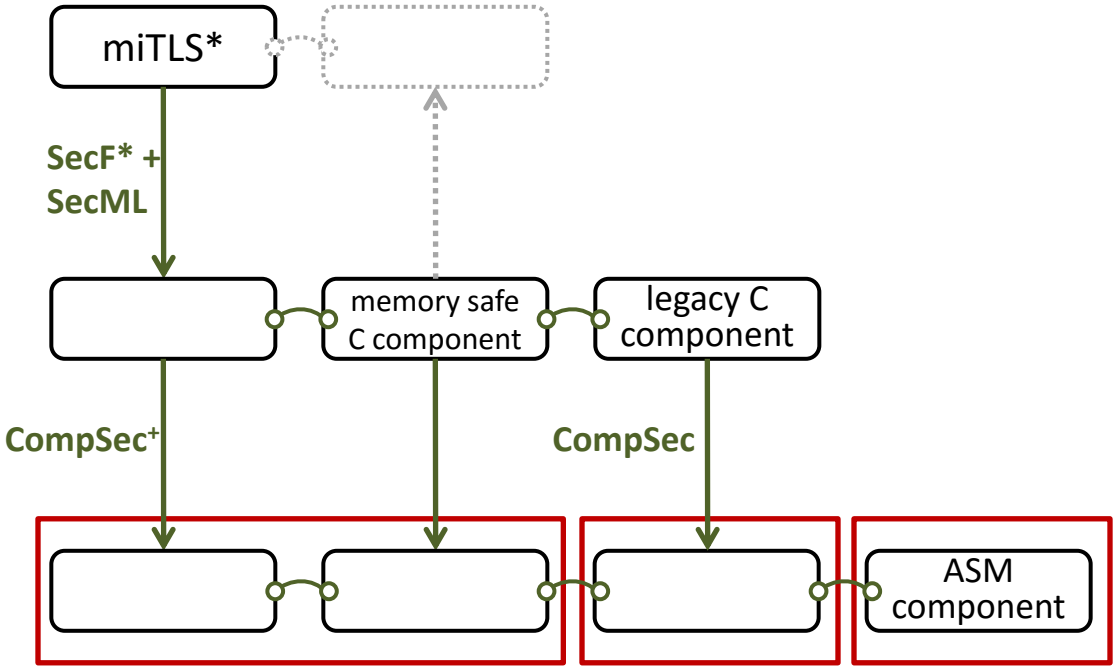


# SECOMP: achieving full abstraction at scale

**F* language**
(ML + verification)

**C language**
+ memory safety
+ components

**ASM language**
(RISC-V + micro-policies)



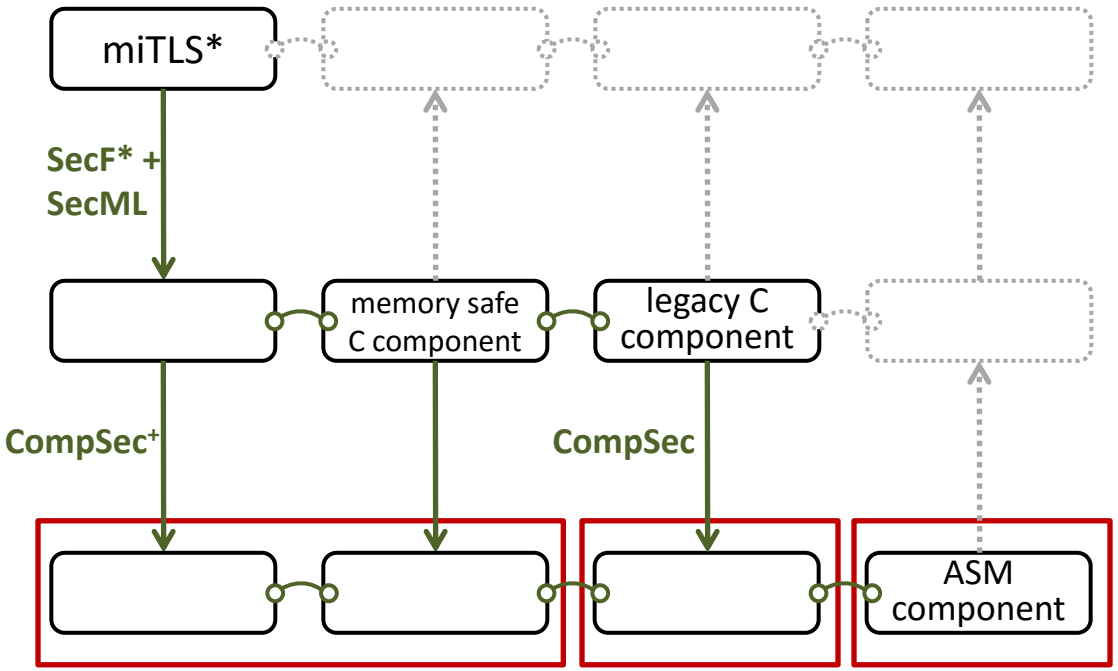

protecting component boundaries

# SECOMP: achieving full abstraction at scale

**F* language**
(ML + verification)

**C language**
+ memory safety
+ components

**ASM language**
(RISC-V + micro-policies)



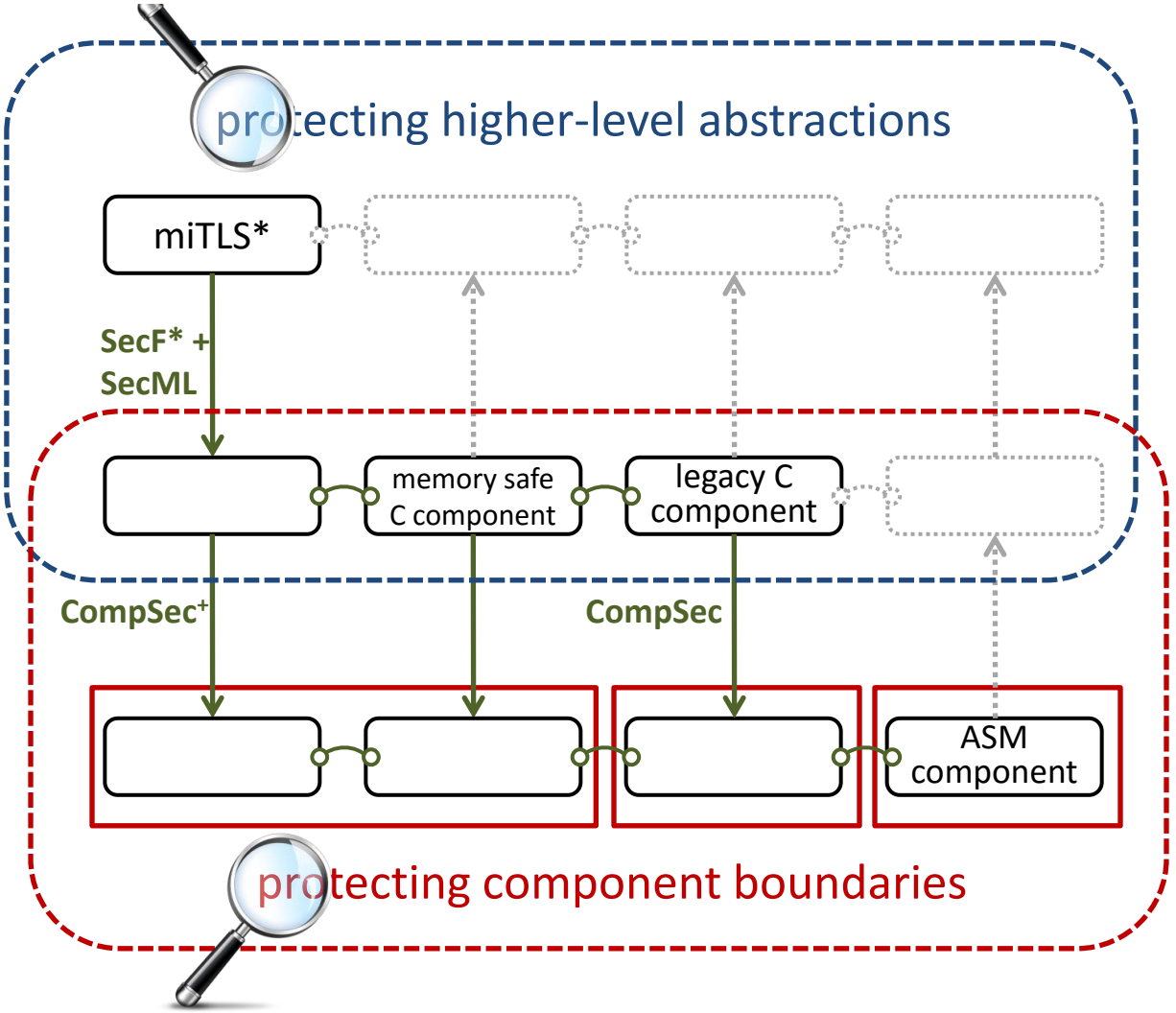

protecting component boundaries

# SECOMP: achieving full abstraction at scale

**F* language**
(ML + verification)

**C language**
+ memory safety
+ components

**ASM language**
(RISC-V + micro-policies)



# Protecting component boundaries

- **Add mutually distrustful components to C**
  - interacting only via **strictly enforced interfaces**
- **CompSec compiler chain** (based on CompCert)
  - propagate interface information to produced binary
- **Micro-policy simultaneously enforcing**
  - component separation
  - type-safe procedure call and return discipline

- **Fundamental challenge: Proper attacker model**

NEW extending full abstraction to mutual distrust + unsafe source

# Protecting higher-level abstractions

- **Enforcing more interesting abstractions** with micro-policies
  - ML: stronger types, value immutability, GC vs malloc/free, ...
  - F*: strong specifications (via dynamic boundary checks)
- **Fundamental challenge: Micro-policies for C and ML**
  - consequence: put micro-policies in the hands of programmers
- **Fundamental challenge: Secure micro-policy composition**
  - one micro-policy's behavior can break another's guarantees

# SECOMP research team

- **Cătălin Hriţcu (principal investigator, 75%)**
- **ERC: 1 Junior Researcher, 2 PostDocs, 3 PhD students**
- 1 already funded PhD student: Yannis Juglaret

| WP | Year 1 | Year 2 | Year 3 | Year 4 | Year 5 |
|---|---|---|---|---|---|
| 1. CompSec | Yannis + JR | | JR | | |
| 2. CompSafe | | JR + PhD 2 | | PhD 2 | |
| 3. CompSec+ | | | JR + PhD 2 | PhD 2 + PostDoc 2 | |
| 4. Compose µP | PhD 1 + JR | | | | |
| 5. C/ML + µP | PhD 1 | PhD 1 + PostDoc 1 | | | |
| 6. SecML | | | PhD 3 | PhD 3 + PostDoc 2 | |
| 7. SecF* | | PostDoc 1 | | | |
| 8. miTLS* | | PostDoc 1 | | PostDoc 2 | |

# Collaborators & Community

- **Ongoing projects**
  - **Micro-Policies:** INRIA, UPenn, MIT, Portland State, Draper Labs
  - **F* and miTLS*:** INRIA, Microsoft Research
  - **CompCert:** INRIA, Princeton
- **New potential collaborators**
  - Several other researchers working on **secure compilation**
    - Deepak Garg (MPI-SWS), Frank Piessens (KU Leuven), Martin Abadi (Google), Amal Ahmed (Northeastern)
- **Secure compilation workshop** @ INRIA Paris, August 2016
  - **build larger research community, identify open problems, bring together communities** (hardware, systems, security, languages, verification, ...)

# SECOMP in a nutshell

- **We need more secure languages, compilers, hardware**
- **Key enabler: micro-policies** (software-hardware protection)
- **Grand challenge: the first efficient formally secure compilers** for **realistic programming languages** (C, ML, F*)
- **Answering challenging fundamental questions**
  - attacker models, composition, micro-policies for C
- **Achieving, testing, and proving full abstraction**
- **Very ambitious and risky milestone project, but ...**
  - experience, preliminary results, team, collaborations, community
- **Impact: unprecedented security, could become mainstream**