SECOMP: Formally Secure Compilation of Compartmentalized C Programs

Cătălin Hrițcu, MPI-SP, Bochum

Joint work with
Carmine Abate, Sven Argo, Arthur Azevedo de Amorim, Roberto Blanco, Adrien Durier, Akram El-Korashy, Ana Nora Evans, Guglielmo Fachini, Deepak Garg, Aïna Linn Georges, Théo Laurent, Benjamin Pierce, Marco Stronati, Jérémy Thibault, Andrew Tolmach, ...

In part supported by ERC Starting Grant SECOMP
Secure Compilation of Vulnerable Source Programs
Secure Compilation of Vulnerable Source Programs

- Insecure languages like C enable devastating vulnerabilities
• Insecure languages like C enable devastating vulnerabilities
• Mitigate vulnerabilities by compartmentalizing the program
• Insecure languages like C enable devastating vulnerabilities
• Mitigate vulnerabilities by compartmentalizing the program
• We don't know which compartments will be compromised
• Insecure languages like C enable **devastating vulnerabilities**
• Mitigate vulnerabilities by compartmentalizing the program
• We don't know which compartments will be compromised
• Insecure languages like C enable **devastating vulnerabilities**
• Mitigate vulnerabilities by compartmentalizing the program
• We don't know which compartments will be compromised
  – protect **vulnerable** C compartments from each other
Secure Compilation of Vulnerable Source Programs

- Insecure languages like C enable devastating vulnerabilities
- Mitigate vulnerabilities by compartmentalizing the program
- We don't know which compartments will be compromised
  - protect vulnerable C compartments from each other
- We don't know when a compartment will be compromised
Insecure languages like C enable devastating vulnerabilities.
Mitigate vulnerabilities by compartmentalizing the program.
We don't know which compartments will be compromised.
- Protect vulnerable C compartments from each other.
We don't know when a compartment will be compromised.
Secure Compilation of Vulnerable Source Programs

• Insecure languages like C enable devastating vulnerabilities
• Mitigate vulnerabilities by compartmentalizing the program
• We don't know which compartments will be compromised
  – protect vulnerable C compartments from each other
• We don't know when a compartment will be compromised
Secure Compilation of Vulnerable Source Programs

- Insecure languages like C enable **devastating vulnerabilities**
- Mitigate vulnerabilities by compartmentalizing the program
- We don't know which compartments will be compromised
  - protect **vulnerable** C compartments from each other
- We don't know **when** a compartment will be compromised
  - every compartment should receive protection until compromised
Insecure languages like C enable devastating vulnerabilities. Mitigate vulnerabilities by compartmentalizing the program. We don't know which compartments will be compromised – protect vulnerable C compartments from each other. We don't know when a compartment will be compromised – every compartment should receive protection until compromised. Formalized this as a variant of robust safety preservation [CCS'18].
Security Enforcement

Large subset of C with compartments
Security Enforcement

Large subset of C with compartments

SECOMP: CompCert extended with secure compartments
Security Enforcement

Large subset of C with compartments

SECOMP: CompCert extended with secure compartments

CompCert RISC-V ASM with compartments

magically secure semantics
Large subset of C with compartments

SECOMP: CompCert extended with secure compartments
magically secure semantics

CompCert RISC-V ASM with compartments

Software-Fault Isolation

vanilla ASM
Large subset of C with compartments

SECOMP: CompCert extended with secure compartments

CompCert RISC-V ASM with compartments

magically secure semantics

Software-Fault Isolation

vanilla ASM

Micro-Policies: ASM with programmable tags

[POPL'14, S&P'15, ASPLOS'15, POST'18, CCS'18, CSF'23]

Hardware-accelerated enforcement
Security Enforcement

Large subset of C with compartments

SECOMP: CompCert extended with secure compartments
magically secure semantics

CompCert RISC-V ASM with compartments

Software-Fault Isolation

vanilla ASM
Done for simplified languages, yet to be ported to RISC-V

Micro-Policies: ASM with programmable tags
[POPL'14, S&P'15, ASPLOS'15, POST'18, CCS'18, CSF'23]

Hardware-accelerated enforcement
**Security Enforcement**

Large subset of C with compartments

SECOMP: CompCert extended with secure compartments

CompCert RISC-V ASM with compartments

magically secure semantics

Software-Fault Isolation

vanilla ASM

Done for simplified languages, yet to be ported to RISC-V

Micro-Policies: ASM with programmable tags

[POPL'14, S&P'15, ASPLOS'15, POST'18, CCS'18, CSF'23]

CHERI RISC-V capability machine

(inspiration for ARM Morello)

Hardware-accelerated enforcement
Large subset of C with compartments

SECOMP

RISC-V ASM with compartments

Software-Fault Isolation

vanilla ASM

Micro-Policies: ASM with programmable tags

CHERI RISC-V capability machine

Done for simpler languages, yet to be ported to RISC-V
Secure Compilation Proofs in Coq

Machine-checked proofs in Coq

Large subset of C with compartments

RISC-V ASM with compartments

SECOMP

RISC-V ASM with compartments

Software-Fault Isolation

vanilla ASM

Micro-Policies: ASM with programmable tags

CHERI RISC-V capability machine

Done for simpler languages, yet to be ported to RISC-V
Secure Compilation Proofs in Coq

Machine-checked proofs in Coq

Large subset of C with compartments

SECOMP

RISC-V ASM with compartments

Scalable proof technique for secure compilation
• first applied to simpler languages [CCS'18, CSF'22]

Software-Fault Isolation

vanilla ASM

Micro-Policies: ASM with programmable tags

CHERI RISC-V capability machine

Done for simpler languages, yet to be ported to RISC-V
Secure Compilation Proofs in Coq

Machine-checked proofs in Coq

Large subset of C with compartments

SECOMP

RISC-V ASM with compartments

Software-Fault Isolation

vanilla ASM

Micro-Policies: ASM with programmable tags

Scalable proof technique for secure compilation
- first applied to simpler languages [CCS'18, CSF'22]
- then scaled up to C compartments [CCS'24]
  - this reuses extended CompCert correctness proof
  - verified strong full-abstraction-like property (~38K LoC)

Done for simpler languages, yet to be ported to RISC-V

CHERI RISC-V capability machine
Secure Compilation Proofs in Coq

Large subset of C with compartments

SECOMP

RISC-V ASM with compartments

Scalable proof technique for secure compilation
• first applied to simpler languages [CCS'18, CSF'22]
• then scaled up to C compartments [CCS'24]
  – this reuses extended CompCert correctness proof
  – verified strong full-abstraction-like property (~38K LoC)
• milestone in terms of realism!

Software-Fault Isolation

vanilla ASM

Micro-Policies: ASM with programmable tags

cheri RISC-V capability machine

Done for simpler languages, yet to be ported to RISC-V
Scalable proof technique for secure compilation
• first applied to simpler languages [CCS'18, CSF'22]
• then scaled up to C compartments [CCS'24]
  – this reuses extended CompCert correctness proof
  – verified strong full-abstraction-like property (~38K LoC)
• milestone in terms of realism!
  – optimizing C compiler with 19 passes

Software-Fault Isolation

- vanilla ASM
- Micro-Policies: ASM with programmable tags
- CHERI RISC-V capability machine

Done for simpler languages, yet to be ported to RISC-V
Secure Compilation Proofs in Coq

Machine-checked proofs in Coq

Large subset of C with compartments

SECOMP

RISC-V ASM with compartments

Scalable proof technique for secure compilation
• first applied to simpler languages [CCS'18, CSF'22]
• then scaled up to C compartments [CCS'24]
  – this reuses extended CompCert correctness proof
  – verified strong full-abstraction-like property (~38K LoC)
• milestone in terms of realism!
  – optimizing C compiler with 19 passes

Software-Fault Isolation

vanilla ASM

Micro-Policies: ASM with programmable tags

Done for simpler languages, yet to be ported to RISC-V

Systematic testing

CHERI RISC-V capability machine
Large subset of C with compartments

SECOMP

RISC-V ASM with compartments

Scalable proof technique for secure compilation
- first applied to simpler languages [CCS'18, CSF'22]
- then scaled up to C compartments [CCS'24]
  - this reuses extended CompCert correctness proof
  - verified strong full-abstraction-like property (~38K LoC)
- milestone in terms of realism!
  - optimizing C compiler with 19 passes

Software-Fault Isolation

vanilla ASM

Micro-Policies: ASM with programmable tags

Done for simpler languages, yet to be ported to RISC-V

CHERI RISC-V capability machine

Big verification challenge for the future

Systematic testing
Future Plans on Formally Secure Compilation

Verify capability backend
Future Plans on Formally Secure Compilation

Better Proof Techniques

Verify capability backend
Future Plans on Formally Secure Compilation

Better Proof Techniques

Capability passing

Verify capability backend
Future Plans on Formally Secure Compilation

- Preserve data confidentiality
- Capability passing
- Better Proof Techniques
- Verify capability backend
Future Plans on Formally Secure Compilation

Stronger Security Goals

Preserve data confidentiality

Better Proof Techniques

Capability passing

Verify capability backend
Stronger Security Goals

Preserve data confidentiality against micro-architectural side-channel attacks, for compartmentalized programs in F*, C, or Wasm

Better Proof Techniques

Capability passing

Verify capability backend
Future Plans on Formally Secure Compilation

Stronger Security Goals

Preserve data confidentiality against micro-architectural side-channel attacks, for compartmentalized programs in F*, C, or Wasm

Realistic Enforcement

Better Proof Techniques

Capability passing

Verify capability backend
Future Plans on Formally Secure Compilation

Stronger Security Goals

- Preserve data confidentiality against micro-architectural side-channel attacks, for compartmentalized programs in F*, C, or Wasm.

Realistic Enforcement

- ARM Morello capability machine
- Capability passing

Better Proof Techniques

- Verify capability backend