Micro-Policies
A Framework for Verified, Hardware-Assisted Security Monitors

Cătălin Hrițcu
INRIA Paris
Current collaborators on this project

• **Formal verification side**
  • Arthur Azevedo de Amorim (UPenn & INRIA Paris)
  • Maxime Dénès (UPenn)
  • Nick Giannarakis (INRIA Paris & NTU Athens)
  • Cătălin Hrițcu (INRIA Paris)
  • Benjamin Pierce (UPenn)
  • Antal Spector-Zabusky (UPenn)
  • Andrew Tolmach (Portland State)

• **Architecture side**
  • André DeHon, Udit Dhawan, Nikos Vasilakis, ... (UPenn)
Computer systems are insecure
Computer systems are insecure

- Today’s CPUs are mindless bureaucrats
  - “write past the end of this buffer” ... yes boss!
  - “jump to this untrusted integer” ... right boss!
  - “return into the middle of this instruction” ... sure boss!
Computer systems are insecure

• Today’s CPUs are mindless bureaucrats
  – “write past the end of this buffer” … yes boss!
  – “jump to this untrusted integer” … right boss!
  – “return into the middle of this instruction” … sure boss!

• Software bears most of the burden for security
  pervasive security enforcement impractical
  – security-performance tradeoff
    just write secure code … all of it!
Computer systems are insecure

• Today’s CPUs are mindless bureaucrats
  – “write past the end of this buffer” ... yes boss!
  – “jump to this untrusted integer” ... right boss!
  – “return into the middle of this instruction” ... sure boss!

• Software bears most of the burden for security
  pervasive security enforcement impractical
  – security-performance tradeoff
  just write secure code ... all of it!

• Consequence: vulnerabilities in every system
  – violations of known safety and security policies
Micro-policies

- general dynamic enforcement mechanism for
  - critical invariants of all machine code
  - high-level abstractions and programming models
Micro-policies

• general dynamic enforcement mechanism for
  – critical invariants of all machine code
  – high-level abstractions and programming models
• main idea: add **word-sized tag** to each machine word
  – “this word is an instruction, and this one is a pointer”
    “this word comes from the net, and this is private to A and B”
Micro-policies

• general dynamic enforcement mechanism for
  – critical invariants of all machine code
  – high-level abstractions and programming models

• main idea: add **word-sized tag** to each machine word
  – “this word is an instruction, and this one is a pointer”
    “this word comes from the net, and this is private to A and B”

• tags efficiently propagated on each instruction
  – tags and rules **defined by software** (miss handler; verified)
    **accelerated by hardware** (rule cache, near-0 overhead hits)
Micro-policies

• general dynamic enforcement mechanism for
  – critical invariants of all machine code
  – high-level abstractions and programming models
• main idea: add **word-sized tag** to each machine word
  – “this word is an instruction, and this one is a pointer”
    “this word comes from the net, and this is private to A and B”
• tags efficiently propagated on each instruction
  – tags and rules **defined by software** (miss handler; verified)
    **accelerated by hardware** (rule cache, near-0 overhead hits)
  low overhead: <10% runtime, <50% energy, <12% power
Micro-policies for ...

- information flow control (IFC)
Micro-policies for ...

- information flow control (IFC)
- monitor self-protection
- dynamic sealing
  compartmentalization
  memory safety
- control-flow integrity (CFI)
- hardware types (instr/ptr/...)
  taint tracking
- ...

12
Micro-policies for ...

- information flow control (IFC) [POPL 2014]
- monitor self-protection
- dynamic sealing
  compartmentalization
  memory safety
- control-flow integrity (CFI)
- hardware types (instr/ptr/...)
  taint tracking
- ...

Verified (in Coq)
Micro-policies for ...

- information flow control (IFC)
- monitor self-protection
- dynamic sealing
- compartmentalization
- memory safety
- control-flow integrity (CFI)
- hardware types (instr/ptr/...)
- taint tracking
- ...

Verified (in Coq)
 recent draft

Evaluated (simulations)

[POPL 2014]
Micro-policies for ...

- information flow control (IFC) [POPL 2014]
- monitor self-protection
- dynamic sealing
- compartmentalization
- memory safety
- control-flow integrity (CFI)
- hardware types (instr/ptr/...)
- taint tracking
- ...

Verified (in Coq)
recent draft

Evaluated (simulations)
Memory safety

• Prevent
  – **spatial violations**: reading/writing out of bounds
  – **temporal violations**: use after free, invalid free
Memory safety

• Prevent
  – spatial violations: reading/writing out of bounds
  – temporal violations: use after free, invalid free

• Pointers become unforgeable capabilities
  – can only obtain a valid pointer to a memory region by allocating that region or
    • by copying/offsetting an existing pointer to that region
Memory safety micro-policy

$$T_v ::= i \mid \text{ptr}(c) \quad \text{tags on values}$$

$$T_m ::= \text{M}(c,T_v) \mid F \quad \text{tags on memory}$$
Memory safety micro-policy

\[ T_v ::= \text{i | ptr(c)} \quad \text{tags on values} \]
\[ T_m ::= M(c,T_v) \mid F \quad \text{tags on memory} \]
Memory safety micro-policy

\[ T_v ::= i \mid \text{ptr}(c) \quad \text{tags on values} \]
\[ T_m ::= \text{M}(c, T_v) \mid \text{F} \quad \text{tags on memory} \]
Memory safety micro-policy

\[
p \leftarrow \text{malloc } k
\]

\[
T_v ::= i \mid \text{ptr}(c) \quad \text{tags on values}
\]

\[
T_m ::= M(c, T_v) \mid F \quad \text{tags on memory}
\]
Memory safety micro-policy

\[ p \leftarrow \text{malloc } k \]

\[ \begin{array}{cccc}
0 & 1 & \ldots & k-1 \\
\end{array} \]

\[ T_v ::= i \mid \text{ptr}(c) \quad \text{tags on values} \]
\[ T_m ::= M(c,T_v) \mid F \quad \text{tags on memory} \]
Memory safety micro-policy

\( p \leftarrow \text{malloc} \ k \)

\( p = \text{A8F0} \)

\[ T_v ::= i \mid \text{ptr}(c) \quad \text{tags on values} \]

\[ T_m ::= M(c,T_v) \mid F \quad \text{tags on memory} \]
Memory safety micro-policy

\[ p \leftarrow \text{malloc } k \]

\[ \text{fresh } c \]

\[ p = \text{A8F0} \]

\[ T_v ::= i \mid \text{ptr}(c) \]  
\[ T_m ::= M(c,T_v) \mid F \]  

tags on values  

tags on memory
Memory safety micro-policy

\[ p \leftarrow \text{malloc } k \]

fresh \( c \)

\[ p = A8F0 \]

\[ T_v ::= i \mid \text{ptr}(c) \quad \text{tags on values} \]

\[ T_m ::= M(c,T_v) \mid F \quad \text{tags on memory} \]
Memory safety micro-policy

\[ p \leftarrow \text{malloc } k \]

fresh \( c \)

\[ p = A8F0 @ \text{ptr}(c) \]

\[ T_v ::= i \mid \text{ptr}(c) \quad \text{tags on values} \]

\[ T_m ::= M(c,T_v) \mid F \quad \text{tags on memory} \]
Memory safety micro-policy

\[ p = \text{A8F0} \] @ptr(c) \]

\[ q \leftarrow p + 1 \]

\[ T_v ::= i \mid \text{ptr}(c) \quad \text{tags on values} \]

\[ T_m ::= M(c, T_v) \mid F \quad \text{tags on memory} \]
Memory safety micro-policy

\[ q \leftarrow p + 1 \]

\[ p = \text{A8F0}@\text{ptr}(c) \]
\[ \text{A8F1}@\text{ptr}(c) = q \]

\[ T_v := i \mid \text{ptr}(c) \quad \text{tags on values} \]
\[ T_m := M(c, T_v) \mid F \quad \text{tags on memory} \]
Memory safety micro-policy

\[ p = A8F0 \_\@ptr(c) \]

\[ A8F1 \_\@ptr(c) = q \]

\[ q \leftarrow p + k \]

\[ T_v ::= i \mid \text{ptr}(c) \quad \text{tags on values} \]

\[ T_m ::= M(c,T_v) \mid F \quad \text{tags on memory} \]
Memory safety micro-policy

\[ p = A8F0 \@\text{ptr}(c) \]

\[ A8FK \@\text{ptr}(c) = q \]

\[ q \leftarrow p + k \]

\[ T_v ::= i \mid \text{ptr}(c) \quad \text{tags on values} \]

\[ T_m ::= M(c, T_v) \mid F \quad \text{tags on memory} \]
Memory safety micro-policy

\[ p = A8F0 \]  
\[ q \leftarrow p + k \]

\[ T_v ::= i \mid \text{ptr}(c) \] tags on values

\[ T_m ::= M(c,T_v) \mid F \] tags on memory

\[ !p \leftarrow 7 \]
Memory safety micro-policy

\[ p = A8F0 @ \text{ptr}(c) \]
\[ !p \leftarrow 7 \]

\[ q \leftarrow p + k \]

\[ T_v ::= i \mid \text{ptr}(c) \quad \text{tags on values} \]
\[ T_m ::= M(c, T_v) \mid F \quad \text{tags on memory} \]
Memory safety micro-policy

\[ p = A8F0 \@ptr(c) \]

\[ q \leftarrow p + k \]

\[ !p \leftarrow 7 \]

\[ T_v ::= i \mid \text{ptr}(c) \quad \text{tags on values} \]

\[ T_m ::= M(c,T_v) \mid F \quad \text{tags on memory} \]
Memory safety micro-policy

\[ p = A8F0 \]@ptr(c) \]

\[ \text{A8FK}@ptr(c) = q \]

\[ q \leftarrow p + k \]

\[ !q \leftarrow 42 \]

\[ T_v ::= i \mid \text{ptr}(c) \quad \text{tags on values} \]

\[ T_m ::= M(c,T_v) \mid F \quad \text{tags on memory} \]
Memory safety micro-policy

\[ p = A8F0@\text{ptr}(c) \]

\[ q \leftarrow p + k \]

\[ c \neq c' \]

\[ !q \leftarrow 42 \]

\[ T_v ::= i | \text{ptr}(c) \quad \text{tags on values} \]

\[ T_m ::= M(c,T_v) | F \quad \text{tags on memory} \]
Memory safety micro-policy

\[
p = A8F0@\text{ptr}(c)
\]

\[
q \leftarrow p + k
\]

\[
c \neq c'
\]

\[
!q \leftarrow 42
\]

out of bounds

\[
T_v ::= i \mid \text{ptr}(c) \quad \text{tags on values}
\]

\[
T_m ::= M(c,T_v) \mid F \quad \text{tags on memory}
\]
Memory safety micro-policy

\[
p = A8F0@\text{ptr}(c)
\]

\[
q \leftarrow p + k
\]

\[
!q \leftarrow 42
\]

Out of bounds

\[
T_v ::= i \mid \text{ptr}(c) \quad \text{tags on values}
\]

\[
T_m ::= M(c,T_v) \mid F \quad \text{tags on memory}
\]
Memory safety micro-policy

\[
p = A8F0_{\text{ptr}(c)}
\]

A8FK_{\text{ptr}(c)} = q

out of bounds

\[
q \leftarrow p + k
\]

\[
!q \leftarrow 42
\]

free \( p \)

\[
T_v ::= i \mid \text{ptr}(c) \quad \text{tags on values}
\]

\[
T_m ::= M(c, T_v) \mid F \quad \text{tags on memory}
\]
Memory safety micro-policy

\[ p = A8F0 \]  
\[ q \leftarrow p + k \]

\( \text{free } p \)

\( x \leftarrow \text{!}p \)

\[ T_v ::= i | \text{ptr}(c) \quad \text{tags on values} \]

\[ T_m ::= M(c, T_v) | F \quad \text{tags on memory} \]
Memory safety micro-policy

\[ p = A8F0 \text{@ptr}(c) \]

\[ q \leftarrow p + k \]

\[ !q \leftarrow 42 \text{ out of bounds} \]

free p

x \leftarrow !p

\[ T_v ::= i \mid \text{ptr}(c) \text{ tags on values} \]

\[ T_m ::= M(c,T_v) \mid F \text{ tags on memory} \]
Memory safety micro-policy

\[ p = A8F0@\text{ptr}(c) \]

\[ \text{A8FK}@\text{ptr}(c) = q \]

\[ q \leftarrow p + k \]

\[ \neg q \leftarrow 42 \] out of bounds

\[ \text{free } p \]

\[ \times x \leftarrow !p \] use after free

\[ T_v ::= i \mid \text{ptr}(c) \] tags on values

\[ T_m ::= M(c,T_v) \mid F \] tags on memory
Memory safety micro-policy
1. Sets of tags

\[ T_v ::= i \mid \text{ptr}(c) \]

\[ T_m ::= M(c,T_v) \mid F \]

\[ T_{pc} ::= T_v \]
## Memory safety micro-policy

### 1. Sets of tags
- $T_v ::= i \mid \text{ptr}(c)$
- $T_m ::= M(c,T_v) \mid F$
- $T_{pc} ::= T_v$

### 2. Transfer function
- **Record IVec** := \{ op:opcode ; $t_{pc}:T_{pc}$ ; $t_i:T_m$ ; ts: ... \}
- **Record OVec** (op:opcode) := \{ $t_{rpc} : T_{pc}$ ; $t_r : ... $ \}
- transfer : (iv:IVec) -> option (OVec (op iv))
1. Sets of tags
   \[ T_v ::= i \mid \text{ptr}(c) \]
   \[ T_m ::= M(c,T_v) \mid F \]
   \[ T_{pc} ::= T_v \]

2. Transfer function
   
   Record IVec := \{ op:opcode ; t_{pc}:T_{pc} ; t_i:T_m ; ts: ... \}
   
   Record OVec (op:opcode) := \{ t_{rpc} : T_{pc} ; t_r : ... \}
   
   transfer : (iv:IVec) -> option (OVec (op iv))

   Definition transfer iv :=
   
   match iv with
   
   | {op=Load; t_{pc}=\text{ptr}(c_{pc}); t_i=M(c_{pc},i); ts=[\text{ptr}(c); M(c,T_v)]}
     => \{ t_{rpc} = \text{ptr} _r(c_{pc}); t_r = T_v \}
Memory safety micro-policy

1. Sets of tags
   \[ T_v ::= i \mid \text{ptr}(c) \]
   \[ T_m ::= M(c,T_v) \mid \text{F} \]
   \[ T_{pc} ::= T_v \]

2. Transfer function
   Record IVec := \{ op: opcode ; t_{pc}: T_{pc} ; t_i: T_m ; ts: \ldots \}
   Record OVec (op: opcode) := \{ t_{rpc}: T_{pc} ; t_r: \ldots \}
   transfer : \text{(iv:IVec)} \rightarrow \text{option (OVec (op iv))}

Definition transfer iv :=
match iv with
| \{op=Load; t_{pc}=\text{ptr}(c_{pc}); t_i=M(c_{pc},i); ts=[\text{ptr}(c); M(c,T_v)]\}
  => \{t_{rpc}=\text{ptr}(c_{pc}); t_r=T_v\}
| \{op=Store; t_{pc}=\text{ptr}(c_{pc}); t_i=M(c_{pc},i); ts=[\text{ptr}(c); T_v; M(c,T_v')]\}
  => \{t_{rpc}=\text{ptr}(c_{pc}); t_r=M(c,T_v)\}
...

...
### Memory safety micro-policy

#### 1. Sets of tags
- \( T_v ::= i \mid \text{ptr}(c) \)
- \( T_m ::= M(c,T_v) \mid F \)
- \( T_{pc} ::= T_v \)

#### 2. Transfer function
- Record IVec := \{ op:opcode ; t_{pc}:T_{pc} ; t_i:T_m ; ts: ... \}
- Record OVec (op:opcode) := \{ t_{rpc} : T_{pc} ; t_r : ... \}
- transfer : (iv:IVec) \rightarrow \text{option} \ (OVec \ (op \ iv))

#### 3. Monitor services
- Record service := \{ addr : word; sem : state \rightarrow \text{option} \ state; ... \}
- Definition mem_safety_services : list service := [malloc; free; size; base; eq].
memory safety
micro-policy
Symbolic machine

Micro-policy

memory safety micro-policy
Symbo lic machine

Concrete machine

Micro-policy

memory safety micro-policy

Rule cache

Monitor

memory safety monitor

ASM
Verification

Symbolic machine

Concrete machine

Rule cache

Monitor

Micro-policy

refines

memory safety micro-policy

memory safety monitor

ASM
Verification

Symbolic machine

Concrete machine

Micro-policy

memory safety micro-policy

correctly implements*

memory safety monitor

ASM

*only proved for IFC [POPL 2014]
Verification

Memory safe abstract machine

Symbolic machine

Concrete machine

Micro-policy

Rule cache

Monitor

memory safety micro-policy

correctly implements*

memory safety monitor

*only proved for IFC [POPL 2014]

ASM
Memory safe abstract machine

Symbolic machine

Concrete machine

Micro-policy

Rule cache

Monitor

Verification

ASM

*only proved for IFC [POPL 2014]
\[ P \in \{IFC, CFI\} \]
\[ P \in \{ IFC, CFI \} \]
Concrete machine

Abstract machine for P

Symbolic machine

Micro-policy

P

Secure

refinement (data)

refinement (data)

P ∈ \{IFC, CFI\}

Concrete machine

Rule cache

Monitor

monitor for P
Abstract machine for $P$

Symbolic machine

Concrete machine

Monitor

Rule cache

Micro-policy

$P \in \{IFC, CFI\}$

secure

preserved by

refinement (data)

refinement (data)
Abstract machine for $P$

Symbolic machine

Concrete machine

Micro-policy

Rule cache

Monitor

Monitor for $P$

$P \in \{IFC, CFI\}$

Prepared by secure refinement (data)

refinement (data)

secure
Future verification challenges

1. Proofs for real RISC architecture (e.g. ARM)
2. Verify all monitors down to machine-code level
3. Formally study micro-policy composition
4. Devise generic meta-language for micro-policies
5. Study more micro-policies (e.g. stack protection, ...)
6. Formally study expressive power of micro-policies
7. Interaction with loader and compiler (static + dynamic)
8. ... and operating system (e.g. protect the OS itself)