DRAPER

DOVER MICROSYSTEMS

A Security Policy Definition Language, Semantics, and Open Source Tools

Chris Casinghino, Draper
Greg Sullivan, Dover Microsystems

RISC-V Workshop, June 2019

This work is sponsored in part by DARPA's System Security Integrated Through Hardware and Firmware (SSITH) program under contract HR0011-18-C-0011.
The views, opinions, and/or findings contained in this talk are those of the author and should not be interpreted as representing the official views or policies, either expressed or implied, of the Department of Defense or the U.S. Government.

Overview

How can we concisely and uniformly express fine-grained security policies?

The plan in three parts:

1. Introduction to our domain-specific language (DSL) for security policies.

2. DSL simulation tools for RISC-V.

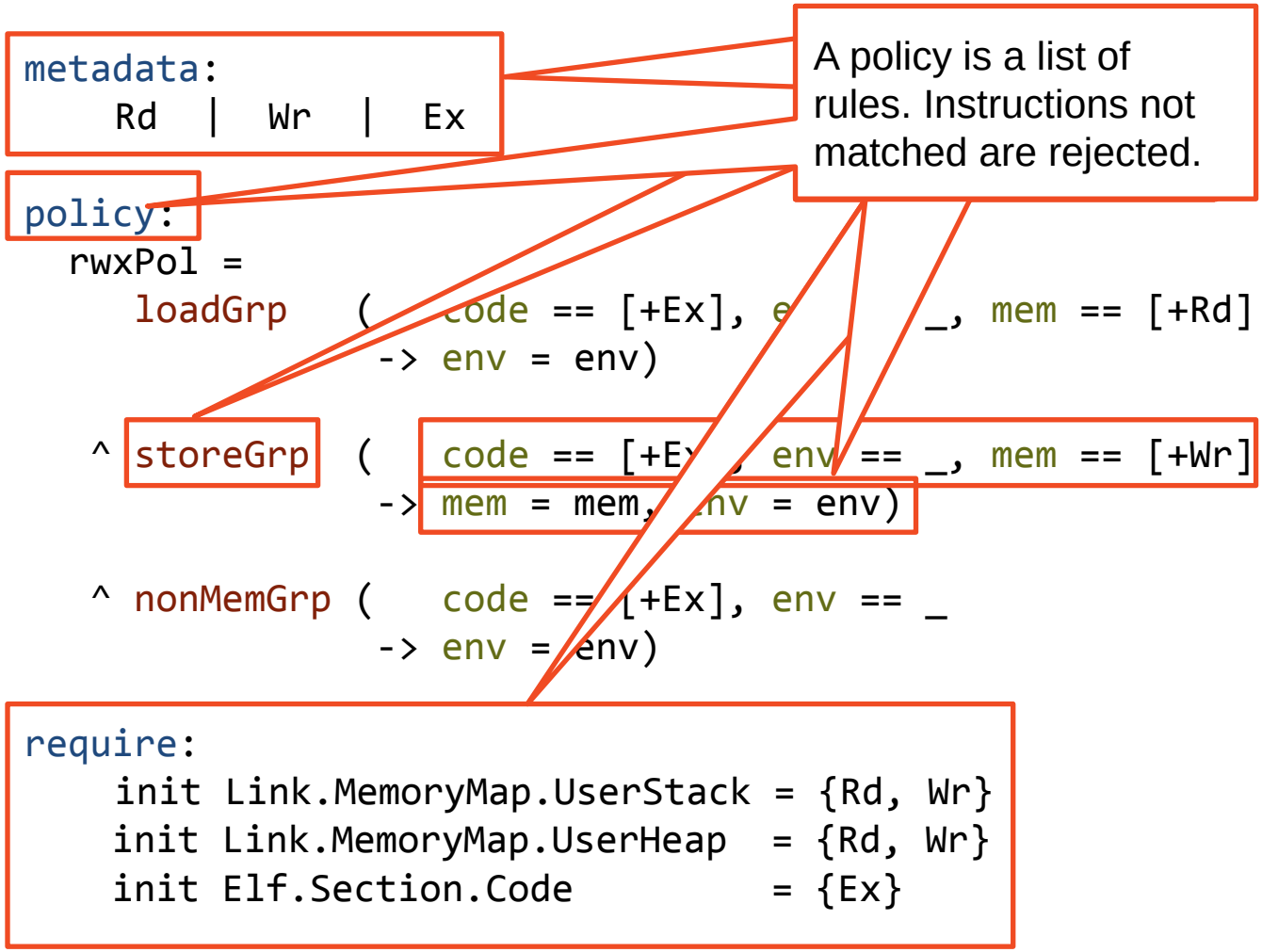
3. DSL hardware enforcement on RISC-V.

DRAPER

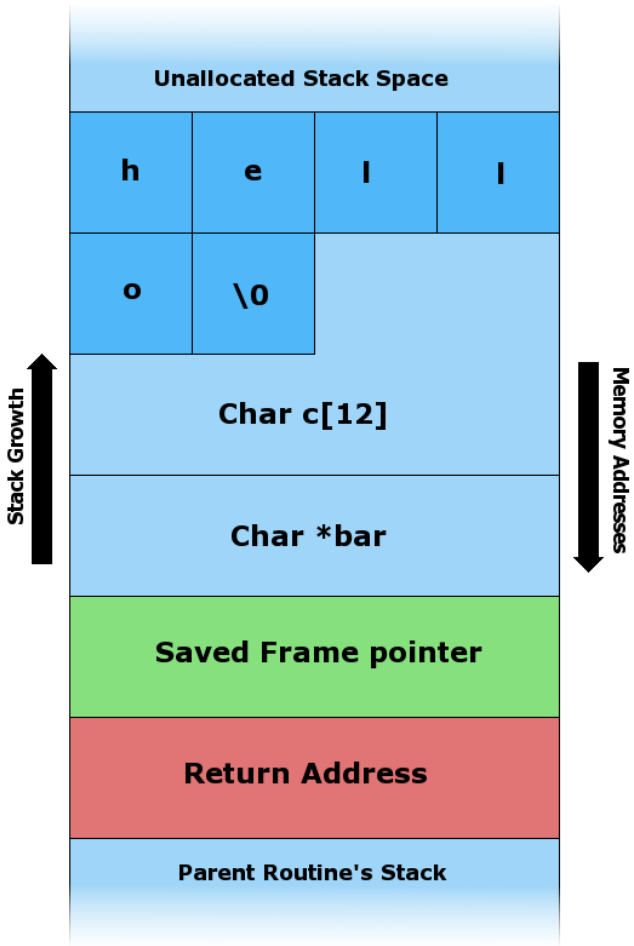
DOVER MICROSYSTEMS

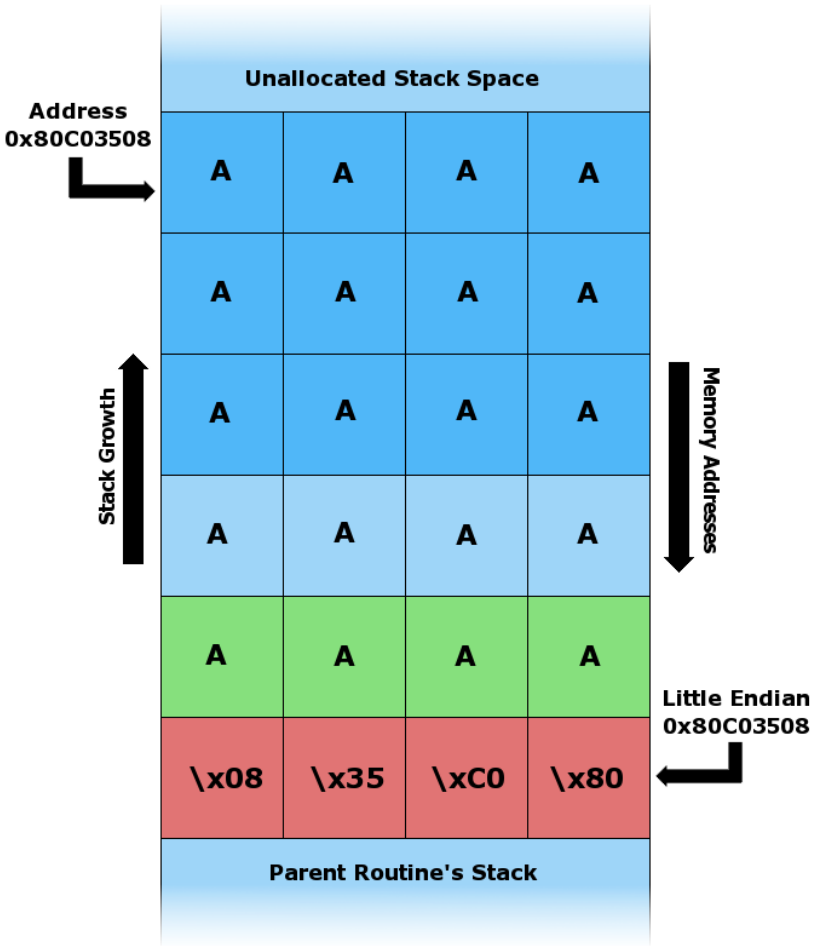
1) A Security Policy DSL

Example: A policy for RWX permissions



Example: Stack Buffer Overflow





Stack Protection Policy

- Tag memory containing function prologue and epilogue code.
- No other code may touch stack frame.

```
metadata:
    Frame  |  Prologue  |  Epilogue

policy:
  stackPol =
     // Prologue creates stack Frame
     storeGrp  (   code == [+Prologue], mem == _, env == _
                -> mem = {Frame})

     // Other code can't touch Frame
   ^ storeGrp (   code == [-Prologue, -Epilogue], mem == [+Frame]
               -> fail "Illegal stack access")

     // Epilogue clears Frame, allow other writes, ...
   ...
```

Example: Simple CFI

- Statically determine legal jump destinations.
- Dynamically check each jump.

```
metadata:
    Target  |  Jumping

policy:
  stackPol =
     // jump instructions set “Jumping”
     jumpGrp  (env == _ -> env = env[+Jumping])

     // Landing on legal target clears “Jumping”
   ^ allGrp   (   code == [+Target], env == [+Jumping]
                -> env = env[-Jumping])

     // Landing elsewhere is a violation
   ^ allGrp   (   code == [-Target], env == [+Jumping]
                -> fail "Illegal jump")
   ^ ...
```

DRAPER

DOVER MICROSYSTEMS

2) DSL Simulation Tools for RISC-V

Open Source Simulation Tools

- Our simulator runs RISC-V programs with policy enforcement.
- Built on QEMU, plugin does policy checks.
- Many policies and test programs.
  - FreeRTOS webserver with application-specific policies.
  - Initial seL4 support.
- Open-source, MIT license, we'd love collaborators:
  - **https://github.com/draperlaboratory/hope-tools**

RISC-V Emulation with Policies

- Simulator detects policy violations in application.
- Provides user feedback on location and cause of violation.

```
cjc1527@leopard> cat pex.log
Policy Violation:
    PC = 20451ad0    MEM = 80502dd8
Metadata:
    Env   : {}
    Code  : {storeGrp}
    Op1   : {}
    Op2   : {}
    Op3   : -0-
    Mem   : {Frame}
Explicit Failure
Illegal stack access
MSG: End test.
cjc1527@leopard>
```

DRAPER DOVER MICROSYSTEMS

Metadata Support in GDB

- Simulator supports GDB.
- New commands to inspect metadata as application runs.

```
(gdb) metadata
Metadata related commnads:
   pvm       - print violation message
   env-m     - get the env metadata
   reg-m n   - get register n metadata
   csr-m a   - get csr metadata at addr a
   mem-m a   - get mem metadata at addr a
Watchpoints halt simulation when metadata changes
   env-mw    - set watch on the env metadata
   reg-mw n  - set watch on register n metadata
   csr-mw a  - set watch on csr metadata at addr a
   mem-mw a  - set watch on mem metadata at addr a
(gdb) mem-m 0x20417688
 {loadGrp, notMemGrp}
(gdb)
```

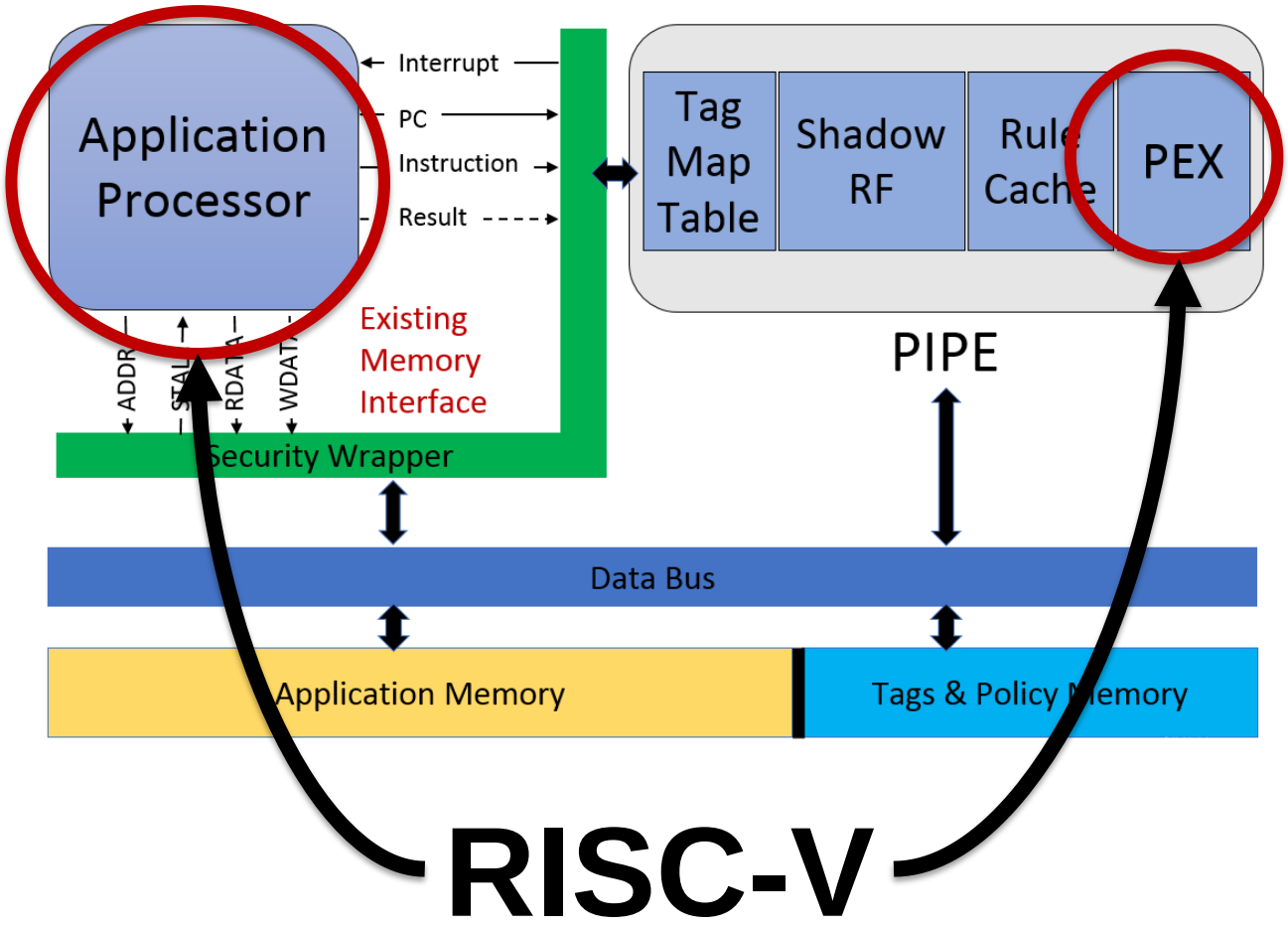
DRAPER

DOVER MICROSYSTEMS

3) DSL Hardware Enforcement on RISC-V

PIPE Hardware Diagram

- PIPE: Processor Interlocks for Policy Enforcement



HW Availability

- Dover and Draper have collaborated on the design of PIPE.
- Dover offers a performant, commercial version, called CoreGuard.

- Draper integrates PIPE for government customers in defense systems.

Conclusion

- Summary:
  - DSLs can make security policies easier to write, understand, and verify.
  - You can play with our open simulation tools today:
    - https://github.com/draperlaboratory/hope-tools
  - Talk to us about performant HW implementations for RISC-V (and other architectures)

  - Greg Sullivan : gregs@dovermicrosystems.com
  - Chris Casinghino: ccasinghino@draper.com

Thanks!

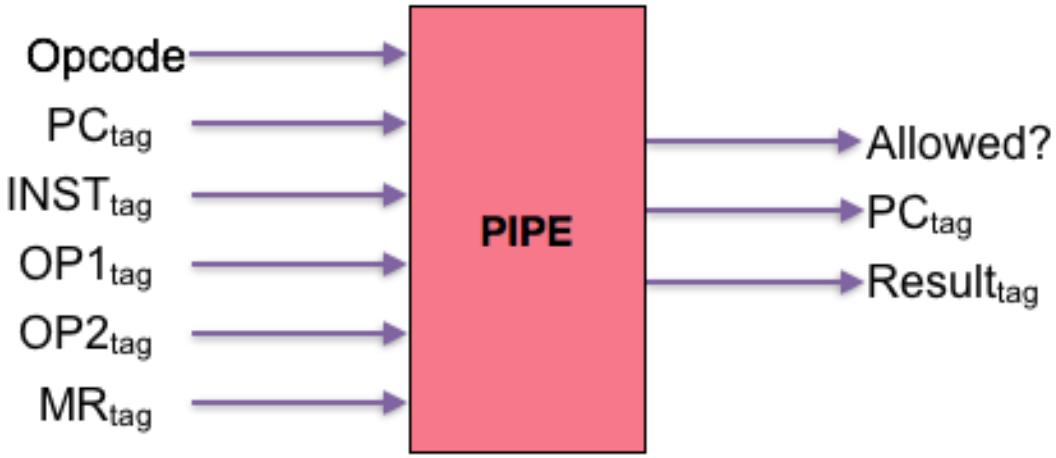
DRAPER

DOVER MICROSYSTEMS

Backup

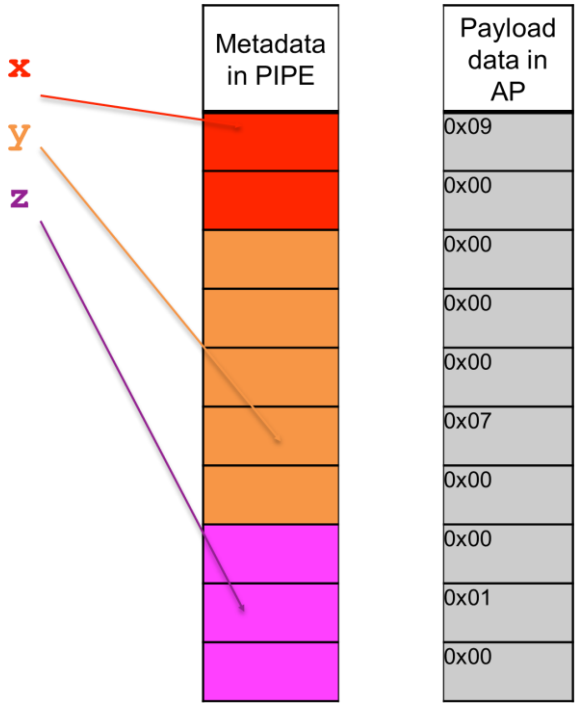
PIPE hardware

- PIPE: Processor Interlocks for Policy Enforcement
- Runs in parallel with application processor.
- Generates application processor interrupt if policy violation occurs.
- Creates unassailable hardware interlock blocking execution of bad instructions.
- Cache keeps performance high.



Example: Heap Memory Safety

- **Policy Goal:** enforce spatial and temporal safety
- **Method:** “color” pointers and memory
  - On allocation, add colors to the region and the pointer.
  - Recolor on free.
- **Policy:**
  - On access, pointer and memory colors must match.




```
z = malloc(3);
z[1] = 0x01;
y = malloc(5);
y[3] = 0x07;
x = malloc(2);
x[0] = 0x09;
x[2] = 0xbad; //FAIL
```

DRAPER DOVER MICROSYSTEMS

Heap Safety Policy

- Tags can carry data values, allowing us to implement this color scheme.

```
type:
    data Color = Int (20)

metadata:
    ApplyColor | NewColor | Pointer Color | Cell Color | ...

policy:
   // Generate new color during malloc
     storeGrp(   code == [ApplyColor], mem == [NewColor]
              -> mem = mem[(Pointer new)])

   // Allow load if colors match
   ^ loadGrp(   mem == [Cell color], addr == [Pointer color]
             -> res = mem[-(Cell _)])

   ...
```

Policy Language Continued

- Users may define new “opgroups”.
  - Group instructions in the way that makes sense for your policy.
  - Keeps policy rules somewhat ISA-agnostic.

- Language implemented in Haskell.
  - We generate C and compile to RISC-V binaries.
  - Lots of room left to optimize.

- Language has formal operational semantics.
  - Future goal: prove implementation correct.