RAFT: Hardware-assisted Dynamic Information Flow Tracking for Runtime Protection on RISC-V

Yu Wang[1], Jinting Wu[1], Haodong Zheng[1], Zhenyu Ning[2,1], Boyuan He[3], Fengwei Zhang[1]

[1] *Southern University of Science and Technology*
[2] *Hunan University*
[3] *Huawei Technologies Co., Ltd.*

October, 16, 2023

SUSTech Southern University of Science and Technology

COMPASS Lab COMPuter And System Security Lab

Dynamic Taint Analysis

```
1  struct SAMPLE{
2    char buffer[20];
3    int private_data;
4  };
5  int test()
6  {
7    struct SAMPLE s;
8    s.private_data = 5; // source ①
9    signed char input_char;
10   int i = 0;
11   while ((input_char = fgetc(fp)) != EOF) // source ②
12   {
13     s.buffer[i] = input_char;
14     i++;
15   }
16   int temp = s.private_data;
17   process(temp); // sink ③
18   return 0; // sink ④
19 }
```

Key Elements

- Taint source
- Taint propagation
- Taint sink

Applications

- Exploit detection
- Privacy leakage detection
- Data corruption detection

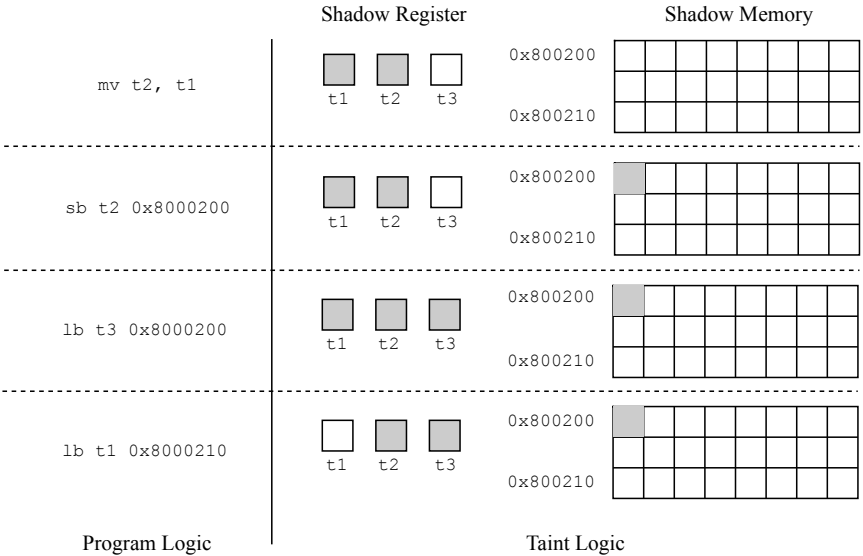
SUSTech
Southern University of Science and Technology

High Performance Overhead

- Software-based DTA tools often suffer from unbearably high performance overhead
  - Due to Dynamic Binary Instrumentation (DBI) or Virtual Machine (VM)
  - Imposing ~4x or even ~10x slowdown
- A research interest emerged in developing hardware accelerators to improve the performance of DTA

SUSTech
Southern University of Science and Technology

Hardware-assisted DTA

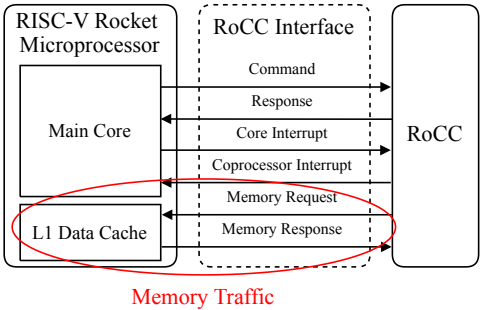


Hardware-assisted DTA on RISC-V

- Utilizing a coprocessor to perform analysis logic on RISC-V

Problem 1

- Non-negligible performance overhead (~20%)
  - Mainly coming from frequent memory operations
  - Reasonable for program analysis but is still unacceptable when protecting **time-critical applications** at runtime (e.g., medical applications and vehicle control units)

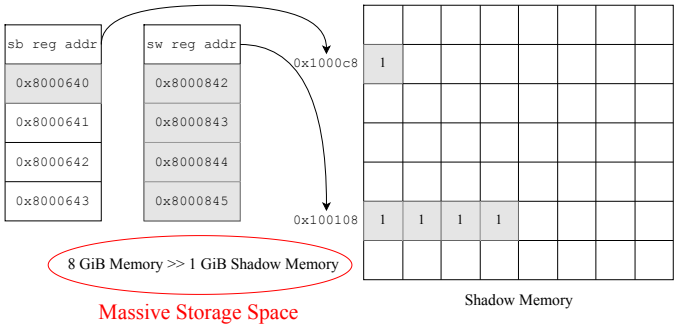


Hardware-assisted DTA on RISC-V

- Utilizing a coprocessor to perform analysis logic on RISC-V

Problem 2

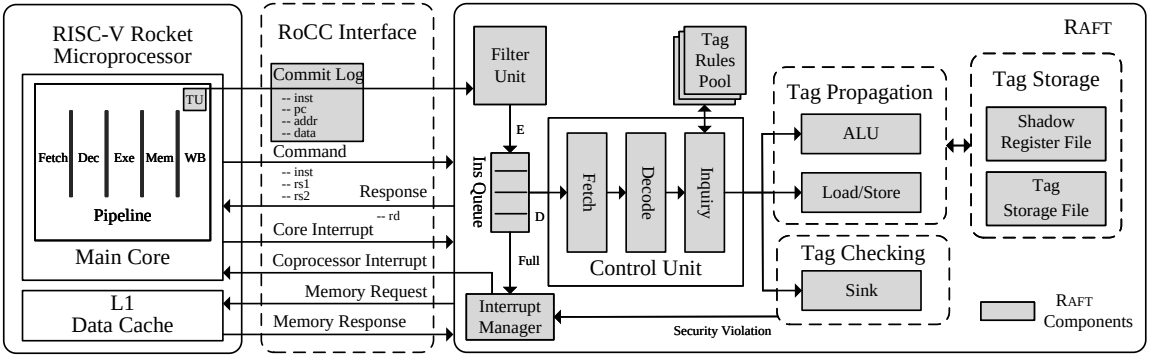
- High memory overhead
  - For example, a flat, fixed-size structure: directly mapping the tags of every memory address into shadow memory requires massive storage space



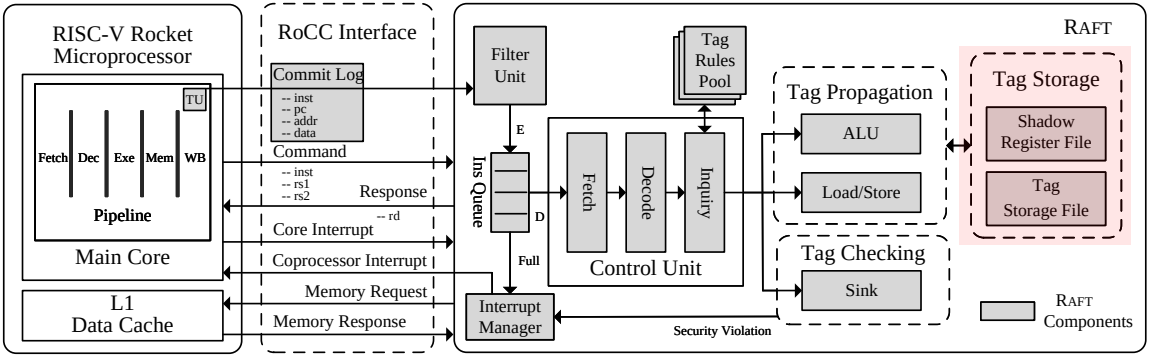
Goals

- A flexible hardware-assisted DTA framework on RISC-V to provide runtime protection for embedded applications **without delay to the programs**
- A new tag-storage mechanism with hybrid byte/variable granularity to **reduce the size of tag storage**

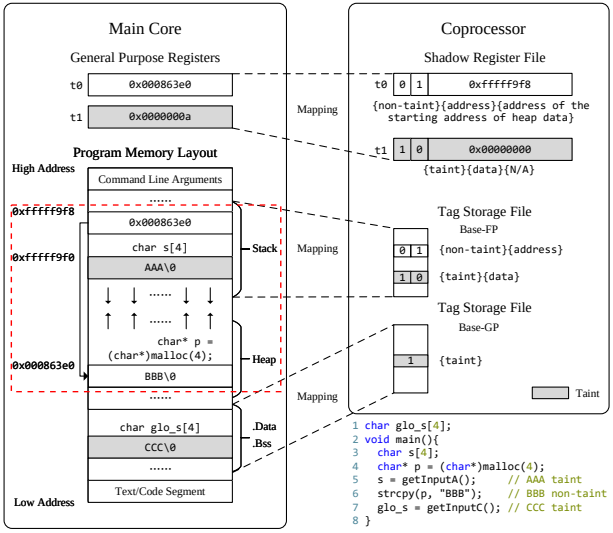
Architecture Overview



Architecture Overview



Tag-storage Mechanism



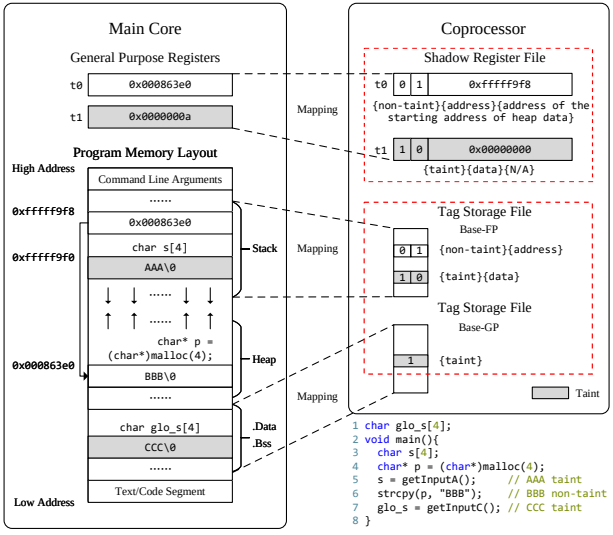
Observation

- The way to access an allocated heap memory is usually through the start address stored on the stack

Design

- Utilizing the information on the stack to represent the tags of heap data

Tag-storage Mechanism



Tag Storage File (TSF)

Tagging program memory

- 2-bit tag:
  - 1 bit: taint or non-taint
  - 1 bit: address or data

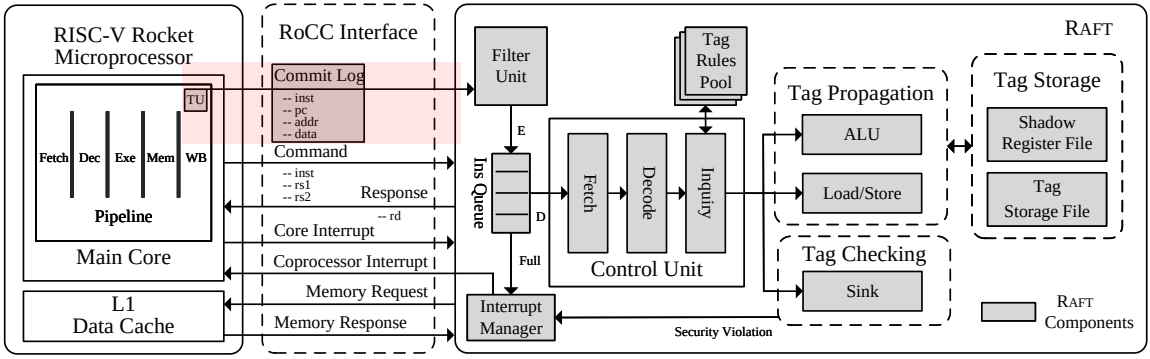
Shadow Register File (SRF)

Tagging general-purpose registers

- 64-bit tag:
  - 1 bit: taint or non-taint
  - 1 bit: address or data
  - **62 bits: the location of the address stored on the stack**

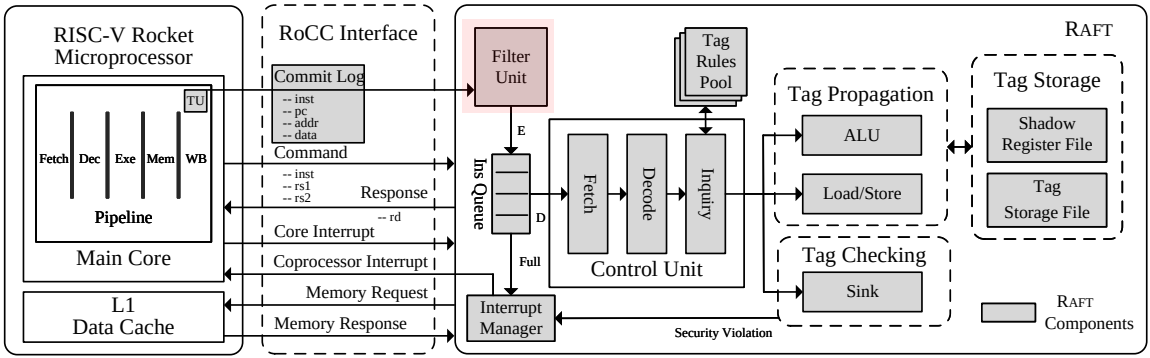
SUSTech

Trace Unit



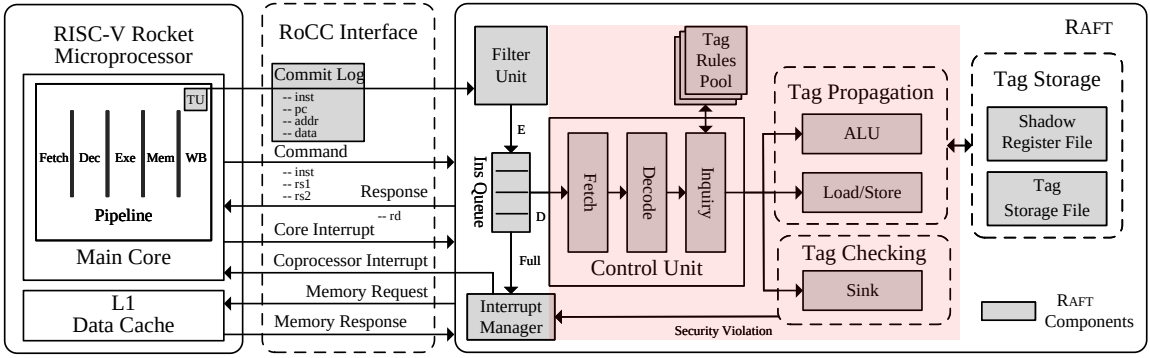
- Collecting the runtime information about the main core and committing it to the coprocessor

Filter Unit



- Filtering out DTA-unrelated instructions that do not involve tag propagation
  - Manually instrumenting two custom instructions before and after the block of unrelated instructions

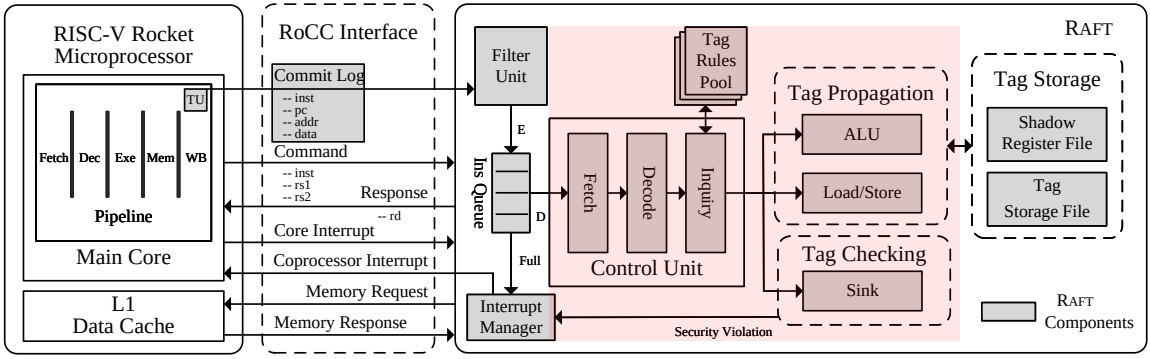
Control Unit



- Controlling analysis logic
  1. Dequeue an instruction and Decode it
  2. Query tag propagation rules
  3. Perform DTA operations

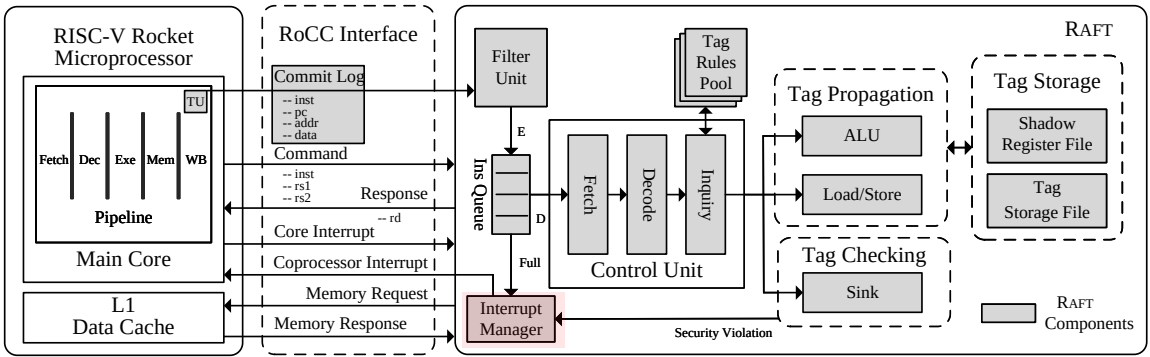
$$Tag(rd) \leftarrow Tag(rs1);$$
$$Tag(rd) \leftarrow Tag(rs1) \vee Tag(rs2);$$
$$Tag(rd) \leftarrow Tag(Mem[rs1 + offset]);$$
$$Tag(Mem[rd + offset]) \leftarrow Tag(rs1);$$

Control Unit



- Pipelining the above tasks
- Moving tags from shadow memory to the coprocessor

Interrupt Manager



- Managing the coprocessor interrupt
  - Security violation (e.g., return address comes from the system input)
  - Instruction Queue is full

Implementation

- Implementing our prototype based on PHMon (USENIX Security'20)
  - An efficient programmable hardware monitor to enforce an event–action monitoring model
  - Utilizing it to flexibly configure tag propagation rules and trace the runtime information
- RAFT is deployed on the RISC-V Rocket emulator and Xilinx Kintex-7 FPGA KC705 evaluation board
- Custom instructions (RISC-V ISA)

Instruction	**Usage**
taint rs1, rs2	Mark the taintedness
src rs1, rs2	Mark heap pointer variables
arg rd, rs1	Assist tag propagation
open/close zero, zero	Filter out DIFT-unrelated instructions
sink rs1, rs2	Perform security checks
base fp/gp, rs2	Pass the frame pointer and global pointer to the coprocessor

- Toolchain (LLVM 12.0.1) and OS support (Linux v5.4)

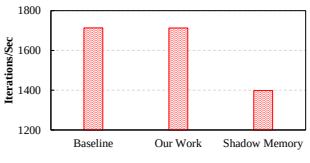
SUSTech

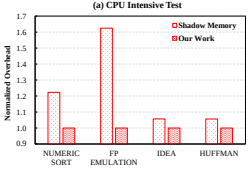
Performance Evaluation

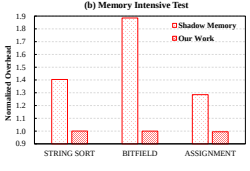
Comparison against Traditional Storage

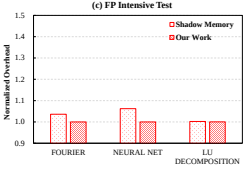
- Shadow Memory: 22.54% overhead on CoreMark and 26.37% overhead on NBench
- RAFT: <0.1% overhead on CoreMark and NBench

- RAFT effectively cuts down the performance overhead from >20% to <0.1% with our new tag-storage mechanism





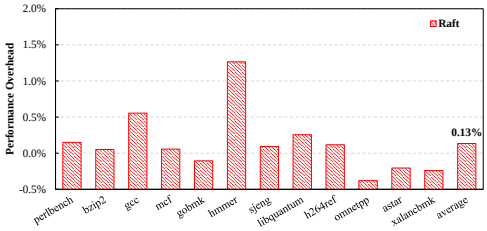




SUSTech
Southern University of Science and Technology

Performance Evaluation

- SPEC CINT 2006: 0.13% overhead on average



- We set up a main core with a peak instruction processing rate 2x and 3x that of the coprocessor and rerun the CoreMark
  - RAFT still introduces negligible performance overhead

Hardware Resource Cost

- An extra usage of 34.04% LUTs and 93.17% FFs
- 2.31% power overhead

	Whole System		Power
	Slice LUTs	Slice Registers	
Without RAFT	58,442	29,445	3.46 W
With RAFT	78,355	56,879	3.54 W
%	+ 34.07%	+ 93.17%	+ 2.31%

SUSTech
Southern University of Science and Technology

Functionality Evaluation

- A medical embedded application OpenSyringePump
  - Detecting a non-control data attack by exploiting a buffer overflow vulnerability
- 5 known CVEs

ID	CVE ID	Program	Vulnerability	Detection
1	CVE-2009-4496	Boa	Information Leakage	✓
2	CVE-2014-8503	Size	Buffer Overflow	✓
3	CVE-2016-3186	Gif2tiff	Buffer Overflow	✓
4	CVE-2018-17100	Ppm2tiff	Integer Overflow	✓
5	CVE-2010-0001	Gzip	Integer Underflow	✓

Conclusion

- A flexible hardware-assisted DTA framework that provides runtime protection for embedded applications without delay to the programs
- A new storage mechanism with hybrid byte/variable granularity
- A prototype on the RISC-V Rocket emulator and FPGA development board

https://github.com/Compass-All/Raft

Thanks!
You can reach me at 12032879@mail.sustech.edu.cn for follow-up questions

SUSTech Southern University of Science and Technology

COMPASS Lab COMPuter And System Security Lab

SUSTech