ASK 2024 논문집
Annual Symposium of KIPS 2024
www.kips.or.kr

신진학자 워크숍

Hardware-Supported Efficient Detection against Software Attacks

서지원 교수
(단국대학교)

Hardware-Supported Efficient Detection against Software Attacks

My Research Topic

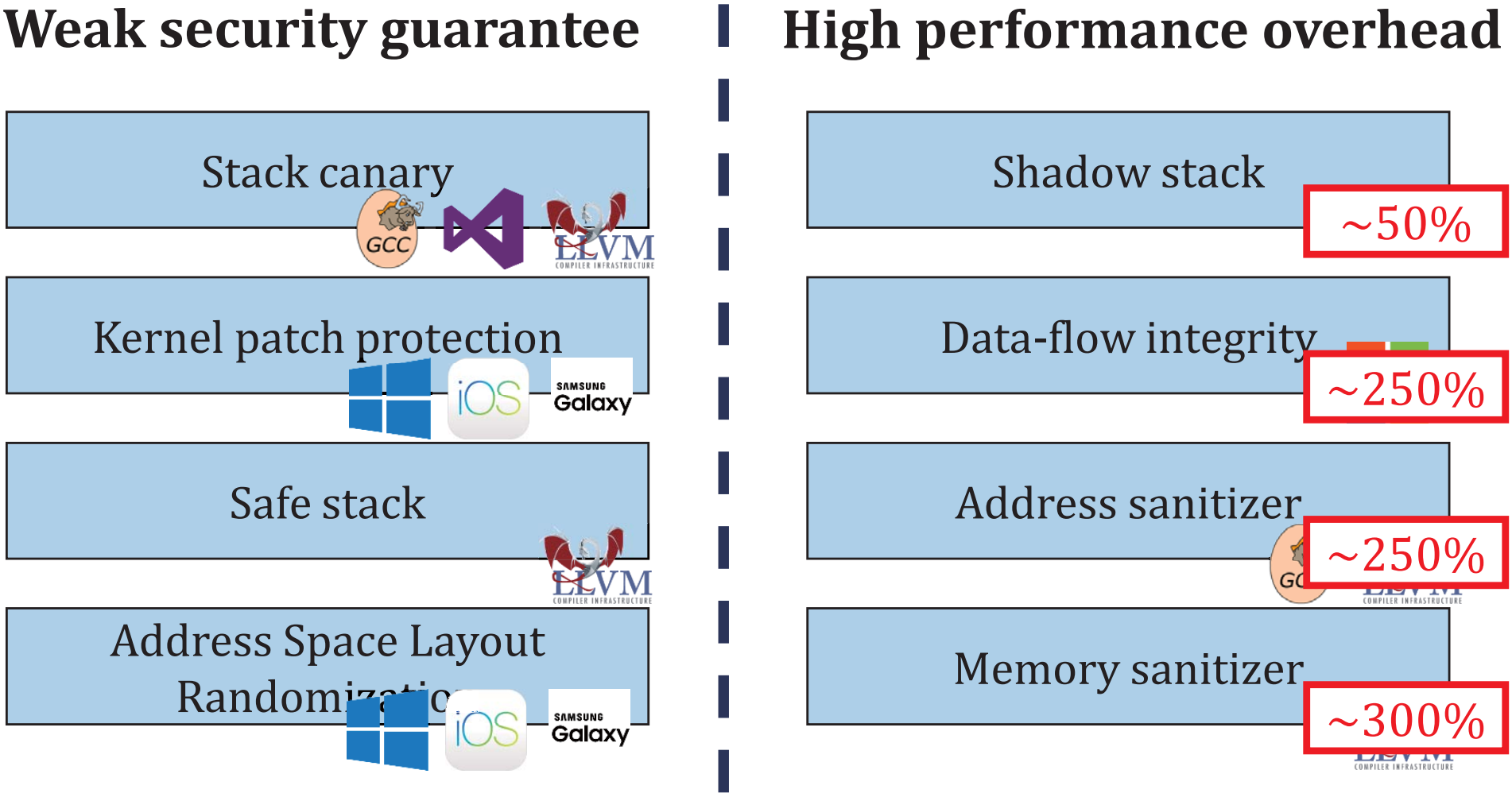
WHAT

Securing Computer Systems

HOW

Security Mechanisms

Software-only Security Mechanisms and their limitations



My Research Topic

WHAT

Securing Computer Systems

HOW

Hardware-assisted
Security Mechanisms

Representative work

- Efficient mitigation techniques using ARM MTE
- Code transformation techniques to enforce security policies for memory protection

ZOMETAG	**Spatial error detection** using ARM MTE
SFITAG	**Kernel driver isolation** using ARM MTE

ZOMETAG: Zone-Based Memory Tagging for Fast, Deterministic Detection of Spatial Memory Violations on ARM

IEEE Transactions on Information Forensics and Security

IF: 7.231

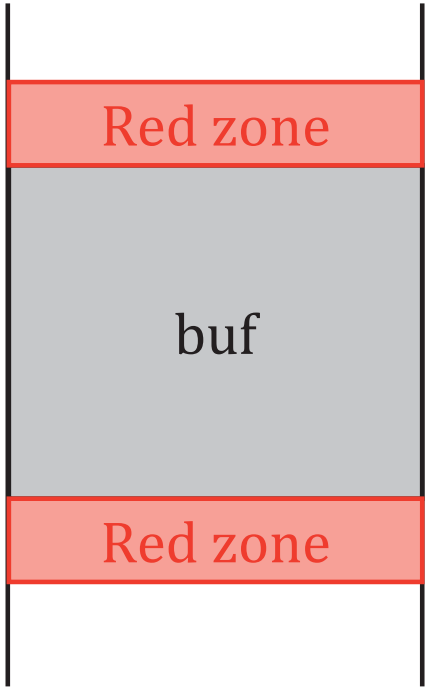
A large number of legacy codebases written in **unsafe C/C++**

- **Problem:** Spatial memory safety violations have been a serious threat to software security for several decades

- Software-based Solutions
  - Mitigating overflows with red zones
  - Explicitly checking bounds

Mitigating overflows with red zones

- Generalization of stack canaries
- Address sanitizer
  - Reserve some virtual pages as red zone
  - Install all store/load to check if they hit the red zone

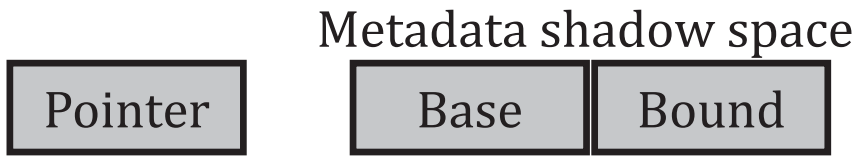


➔ **High performance overhead**

Explicitly checking bounds

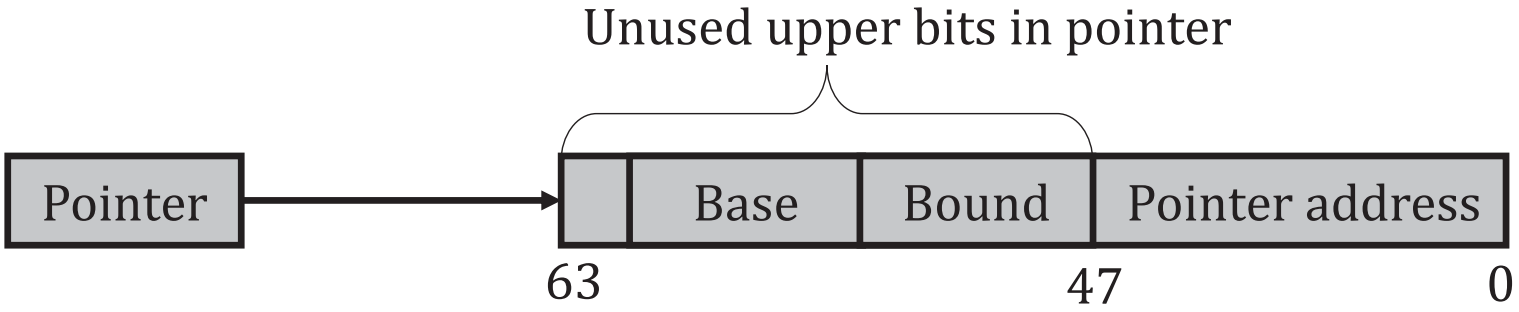
- Check pointers with base and bound information

- Where to put the metadata
  - Outline: keep base/bound in a designated table



➔ **High performance overhead**

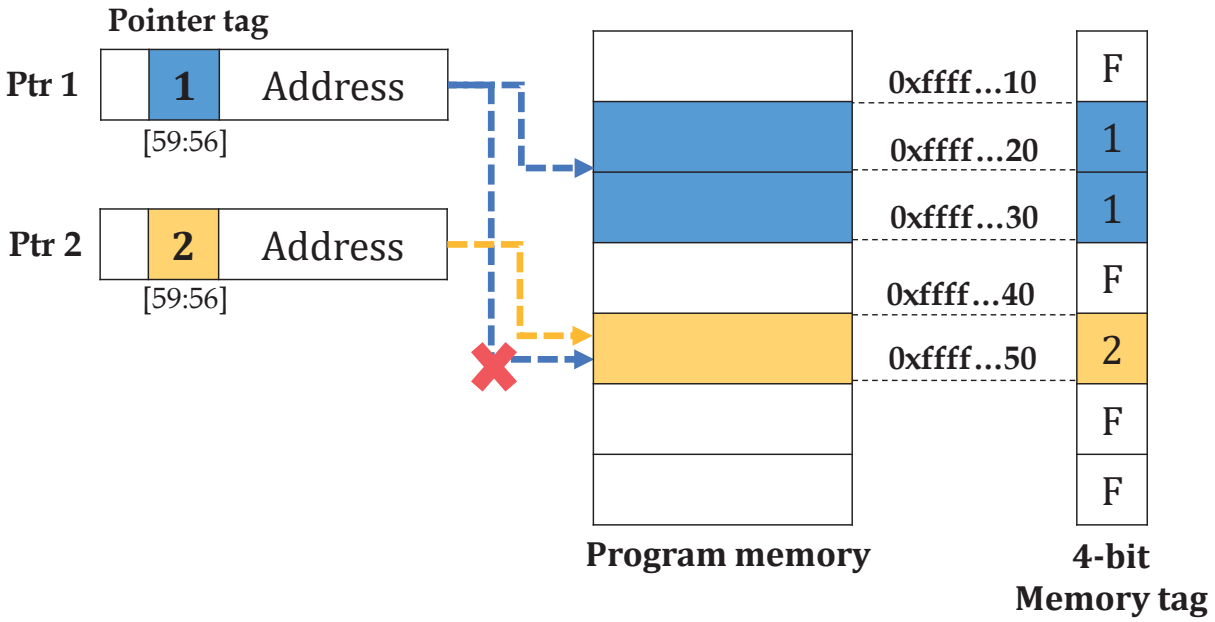
  - Inline: expand each pointer



➔ **Weak security guarantee**

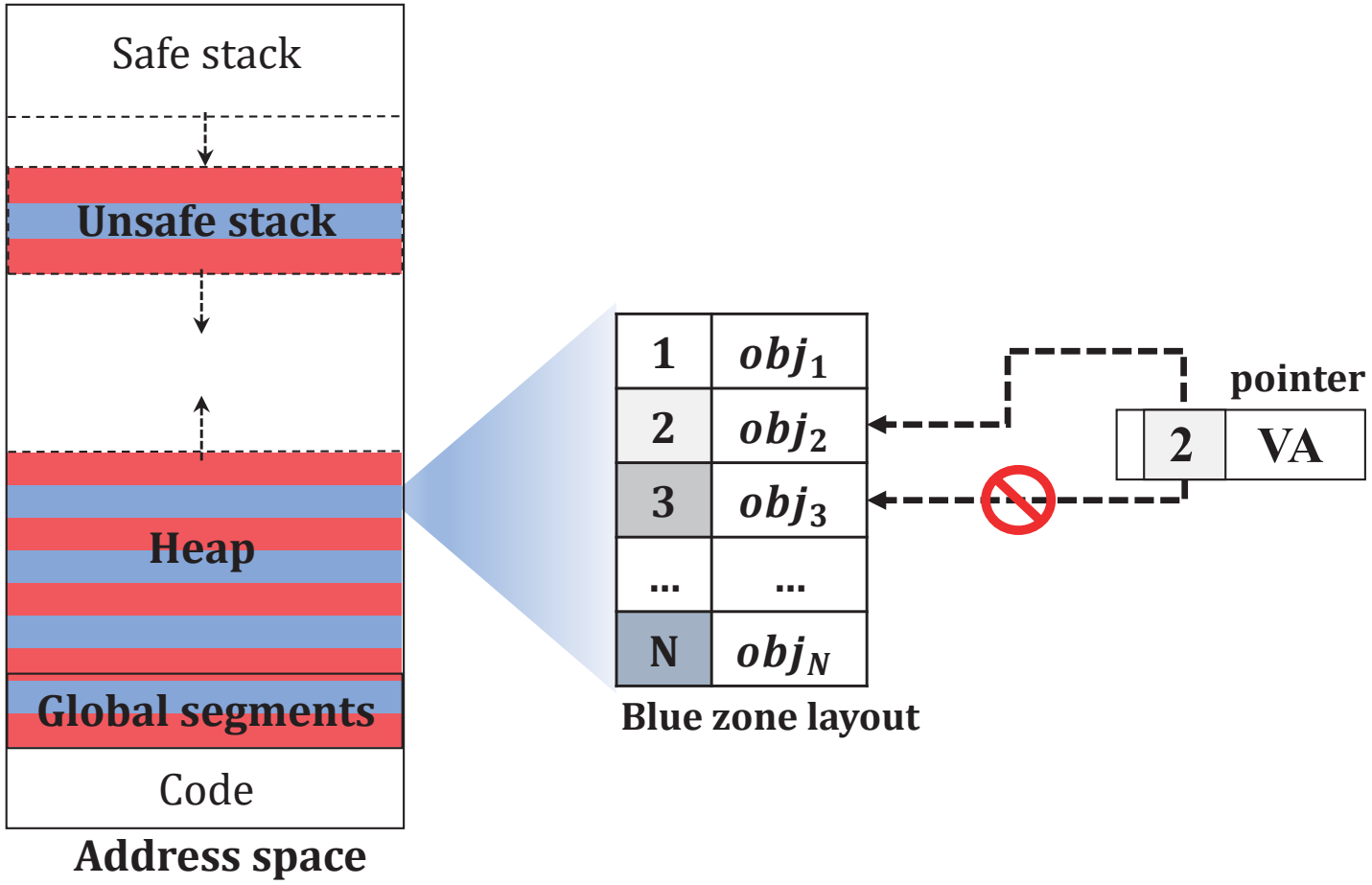
Solution: strong and low overhead security with hardware support

- ARM processors provides a **hardware feature** that enhance memory safety using tag management
  - MTE utilizes a 4-bit memory tag at the granularity of 16 bytes to tag the memory
  - Pointers are allowed to access the memory with the matching tag



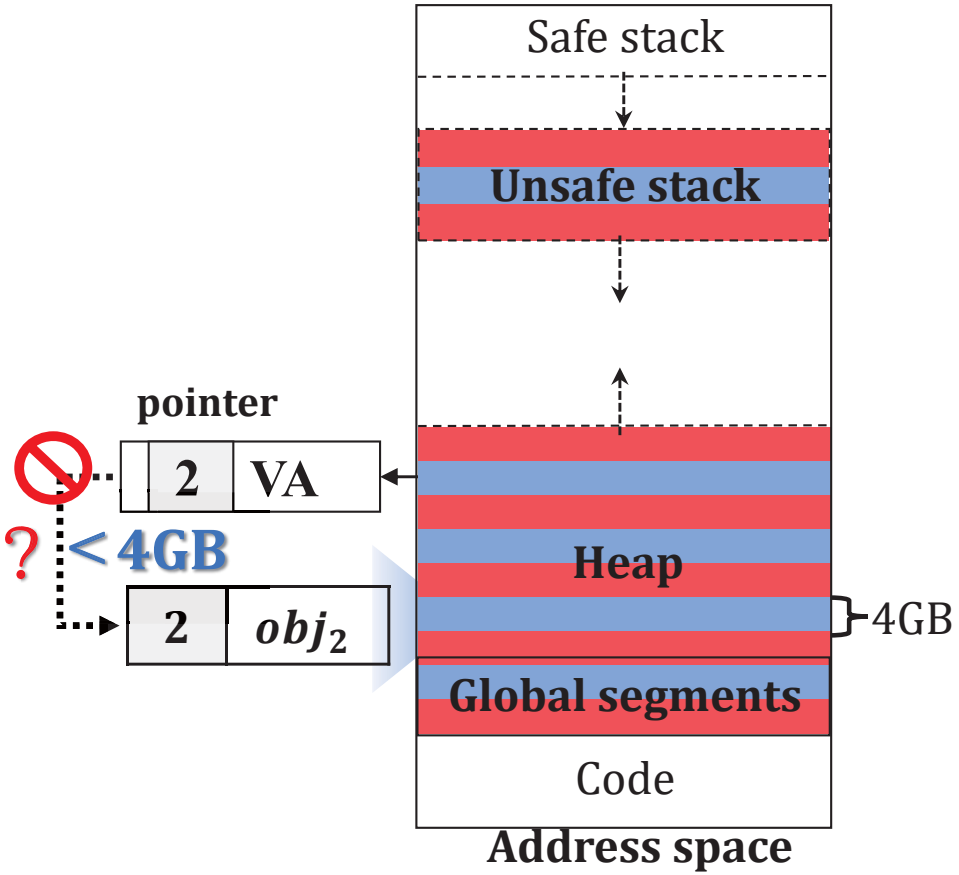
ZOMETAG: strong and low overhead security with hardware support

- Zone construction
  - The number of objects allocated in a blue zone must not be more than that of available MTE tags



ZOMETAG: strong and low overhead security with hardware support

- Zone-based isolation
  - Pointer arithmetic is instrumented to have 32-bit registers as operands
  - Objects allocated in the same blue zone must not exceed the zone size (4GB)



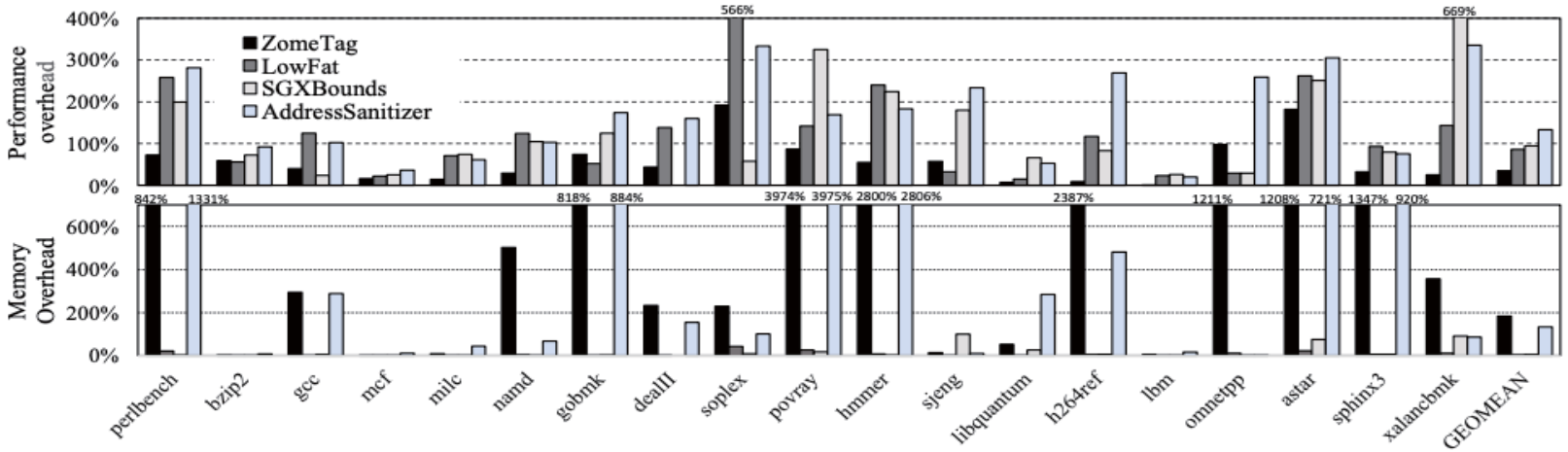

```
<foo>:                                  <foo>:
   movz   x0, #0x0, lsl #48                movz   x0, #0x0, lsl #48
   ......                                  ......
   ldr    x0, [x1, x2]                     ldr    x0, [x1, w2]          ①
   mov    w9, w1                           mov    w9, w1
   ......                                  ......
   ldr    x4, [sp, #16]                    ldr    x4, [sp, #16]
   str    x3, [x4, x5]                     str    x3, [x4, w5]          ②
   ......                                  ......
   ldr    x7, [sp, #32]                    ldr    x7, [sp, #32]
   add    x7, x7, #100                     add    x7, x7, #100
                                           tbz    x7, #32, <overflow>   ③
```

(a) Before **(b) After**

Evaluation

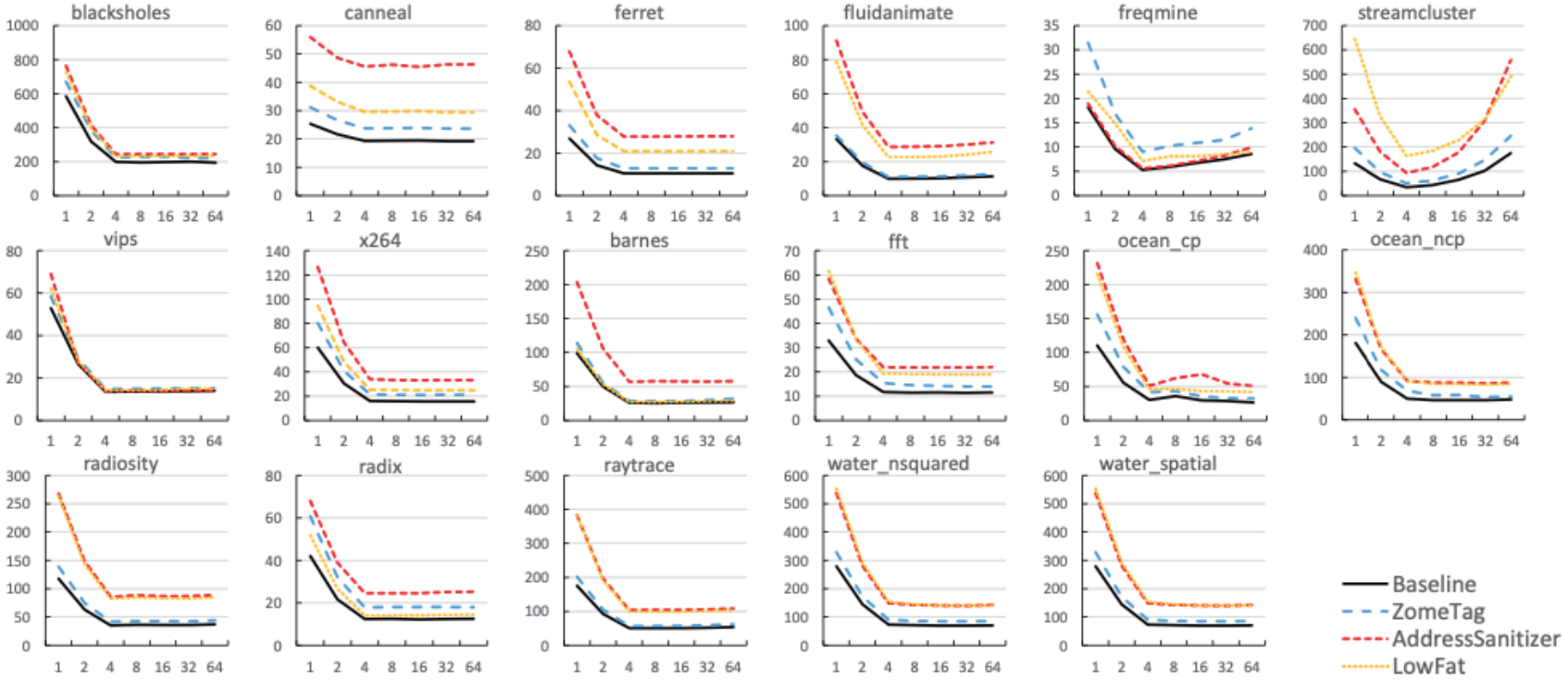
- Experimental Setup
  - ODROID-C4 with Cortex A-55 quad-core CPU @ 2.0 GHz and 4 GB RAM
  - LLVM 9.0.0

- SPEC 2006 benchmark (single-threaded overhead)
  - **ZOMETAG: 35%,** SGXBounds: 94%, LowFat: 86%, ASAN: 133 %



Evaluation

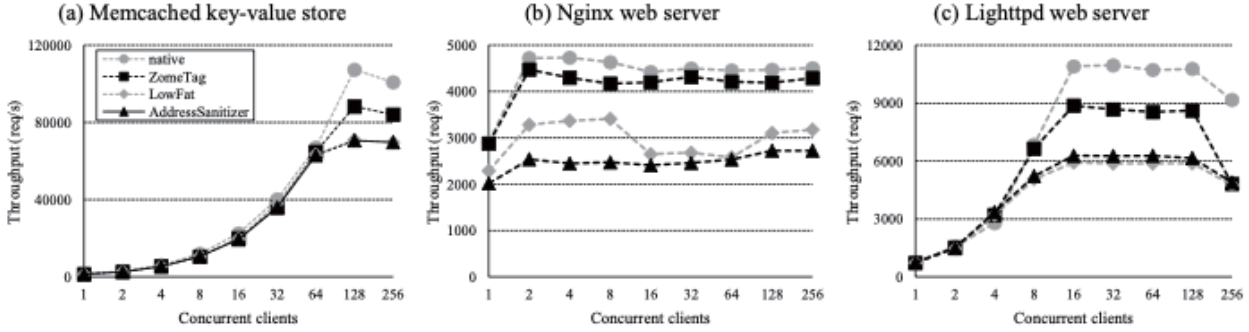
- PARSEC 3.0 benchmark (multi-threaded overhead)
  - **ZOMETAG: 24.6%~30%**
  - LowFat: 67.1%~82.8%
  - ASAN: 93.7%~101%



Evaluation

- Memcached (v1.4.15) using memaslp benchmark
  - **ZOMETAG: 13% decrease in throughput**
  - LowFAT: 22% decrease in throughput
  - ASAN: 23% decrease in throughput

- Web server applications: Nginx (v1.4.0), Lighttpd (v1.4.59) using ApachBench
  - **ZOMETAG: 94% throughput**
  - ASAN: 57% throughput
  - LOWFAT: 67% throughput



Evaluation

- **BugBench**: Applications containing spatial errors in real programs

Application	Source code	Bug type	Detected
bc-1.06	id->a_name=next_array++; a_names[id->a_name]=name;	heap	Zone-based
	sprintf(genstr, "F%d,%s.%s[",...);	global	Tag-based
gzip-1.2.4	strcpy(ifname, iname);	global	Tag-based
man-1.5h1	tmp_section_list[i++] = my_strdup (p);	stack	Tag-based
ncompress	strcpy(tempname,*fileptr);	stack	Tag-based
polymorph-0.4.0	strcpy(target, optarg);	global	Tag-based
	strcpy(newname, "");	stack	Tag-based

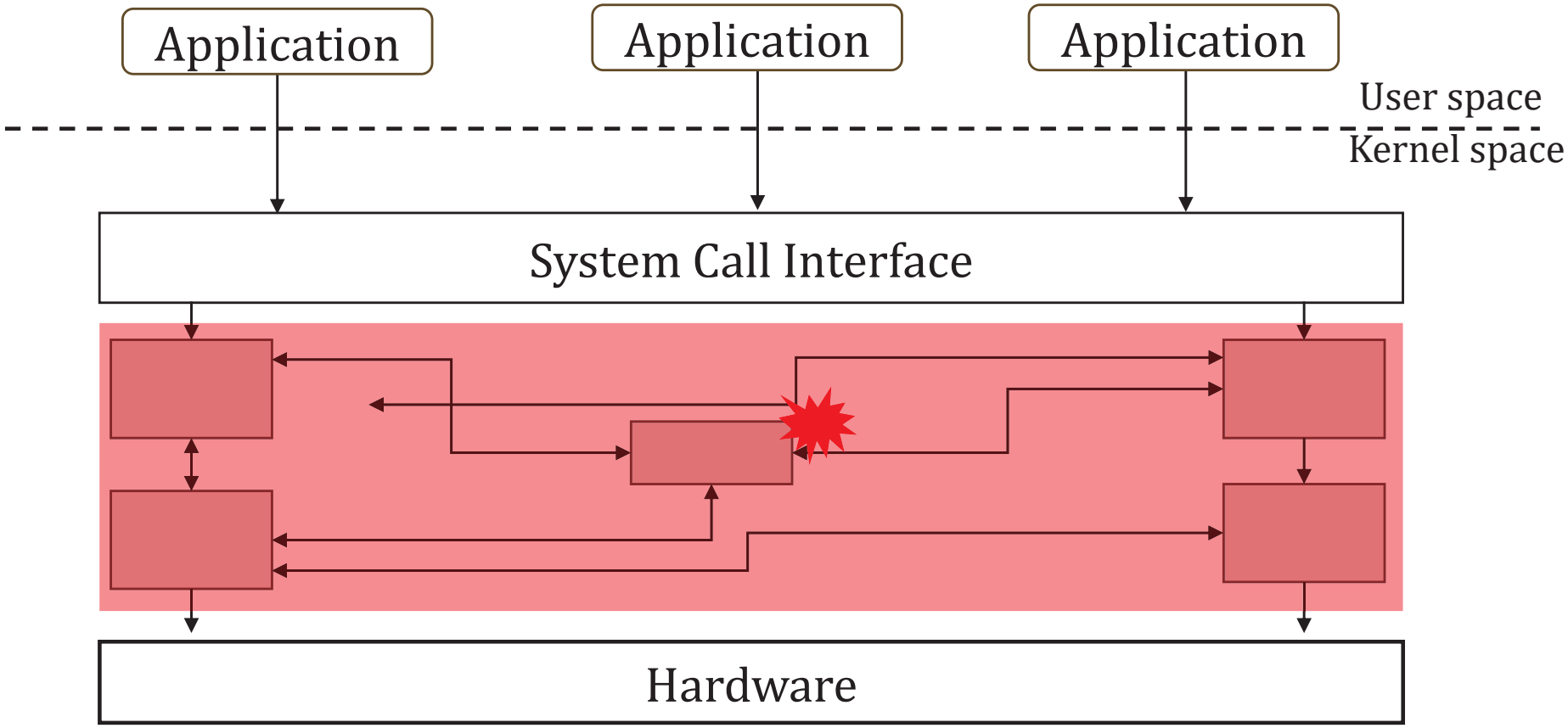
Sfitag: Efficient Software Fault Isolation with Memory Tagging for ARM Kernel

Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security

CS분야 우수학술대회

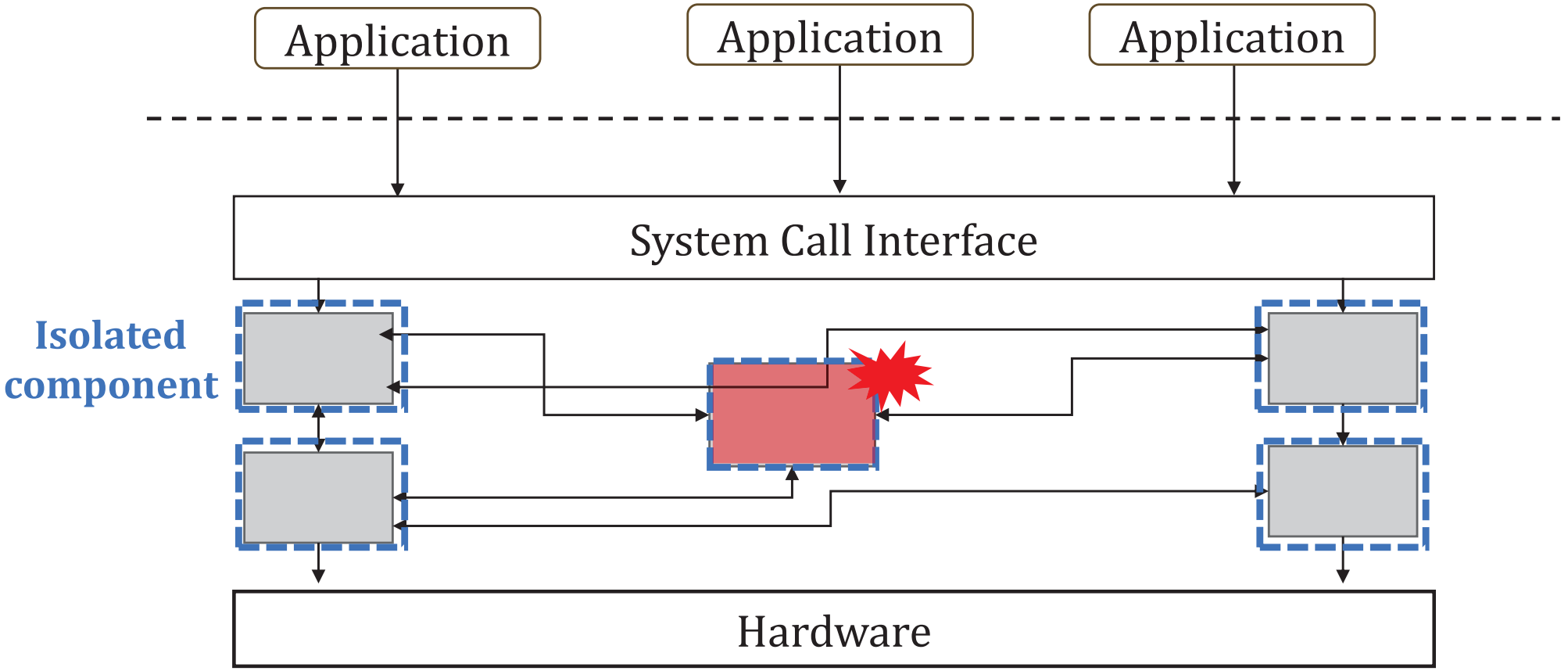
Commodity kernels are monolithic

- **Problem**: Kernel extensions (i.e., device drivers) are an abundant source of vulnerabilities in OS



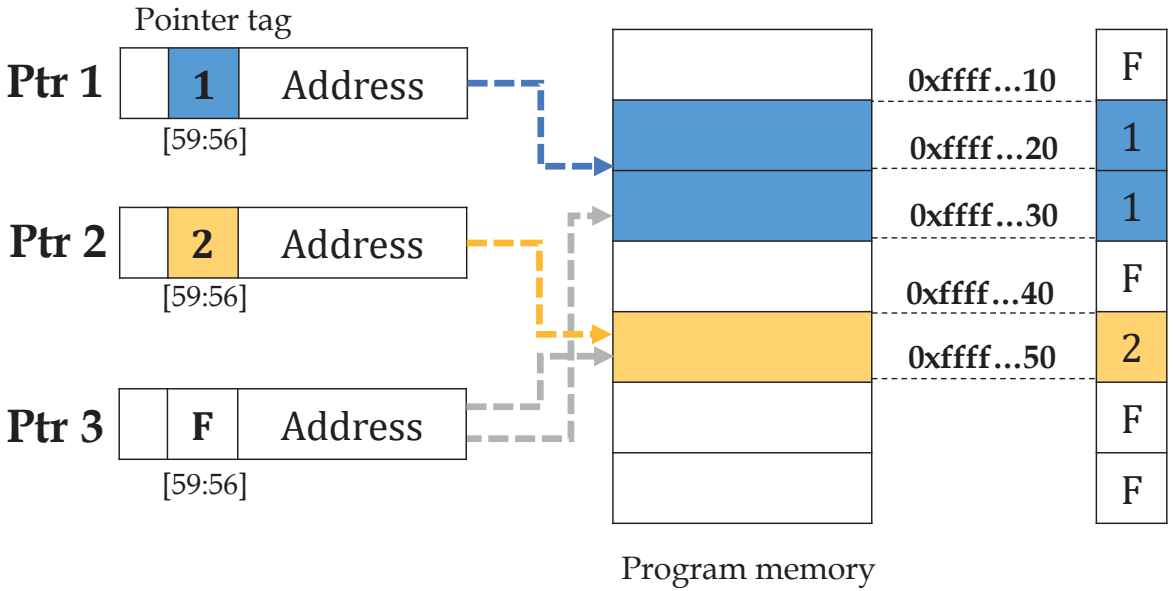
Applying Isolation to Kernel

- Isolation
  - A security mechanism to **reduce the likelihood of attacks** by decreasing the effective code size.



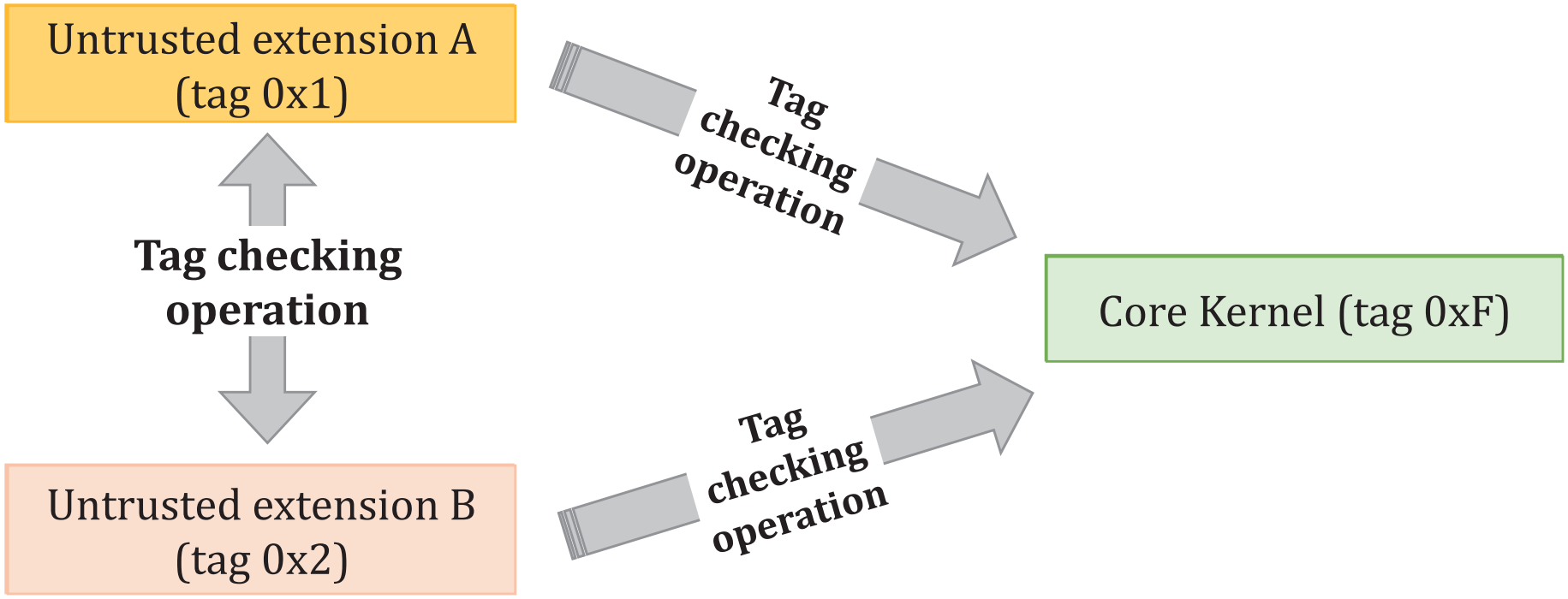
Solution: Replace domain crossing costs with cheap hardware tag checks

- SFITAG also utilizes ARM MTE to enforce memory isolatin.
  - Pointers are allowed to access the memory with the matching tag
  - Tag Unchecked access
    - Pointer tag: 0b1111 (0xF)



Solution: Hardware-supported kernel isolation

- Isolate kernel extensions by forcing them to use ARM MTE tag values



Evaluation

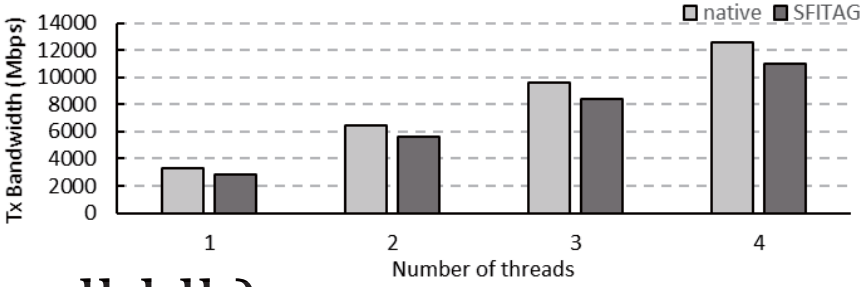
- Experimental setup
  - ODROID-C4 development board with ARMv8-A
  - kernel version 4.9.236
  - LLVM 9.0 compiler framework

- Software-only device
  - Dummy network driver, Null block driver
    - Utilize kernel subsystems with the tightest performance budgets
    - Help understand the overhead isolation

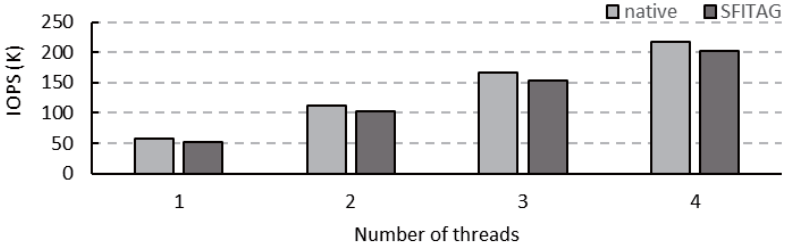
- Hardware device
  - STMicroelectronics 1Gbps Ethernet driver (stmicro)

Software-only device

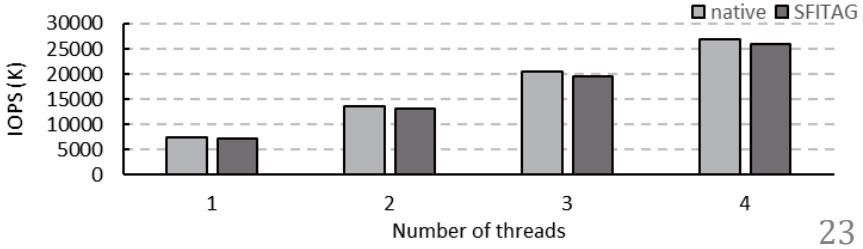
- Nullnet driver
  - Iperf3 benchmark (Transmission IOPS)
  - the non-isolated driver (native) : 3281 Mbps
  - **SFITAG: 2880 Mbps (87.7%)**



- block device (null-blk)
  - fio benchmark
  - 512B packet size : Native: 57K IOPS, **SFIKE: 52.6K IOPS (92.28%)**
  - 1MB packet size :Native: 7427 IOPS, **SFIKE: 7141 IOPS (96.15%)**



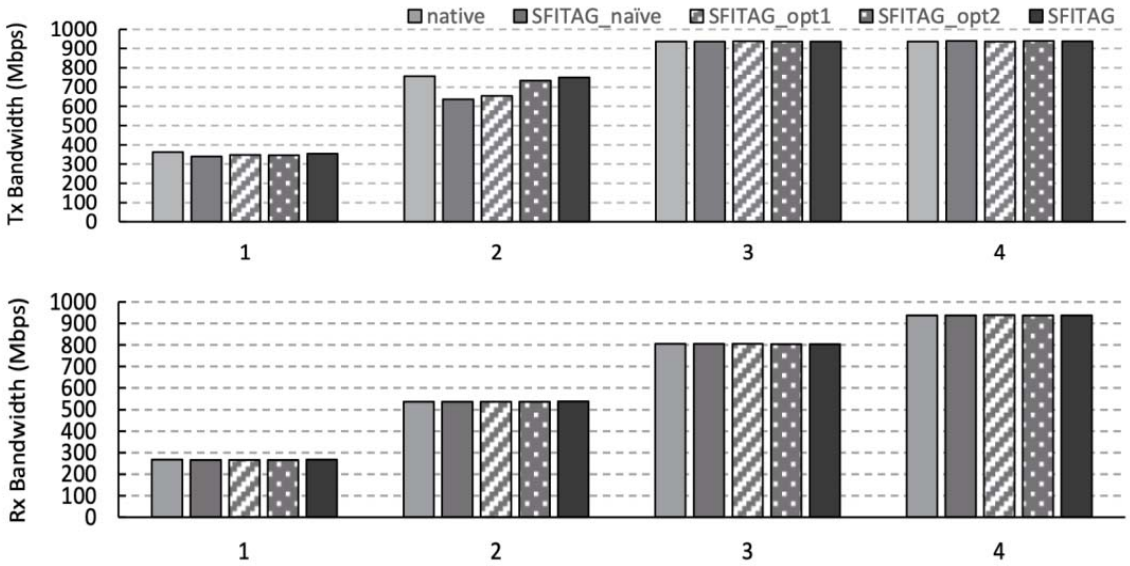
<512B packet size>



<1M packet size>

Hardware device

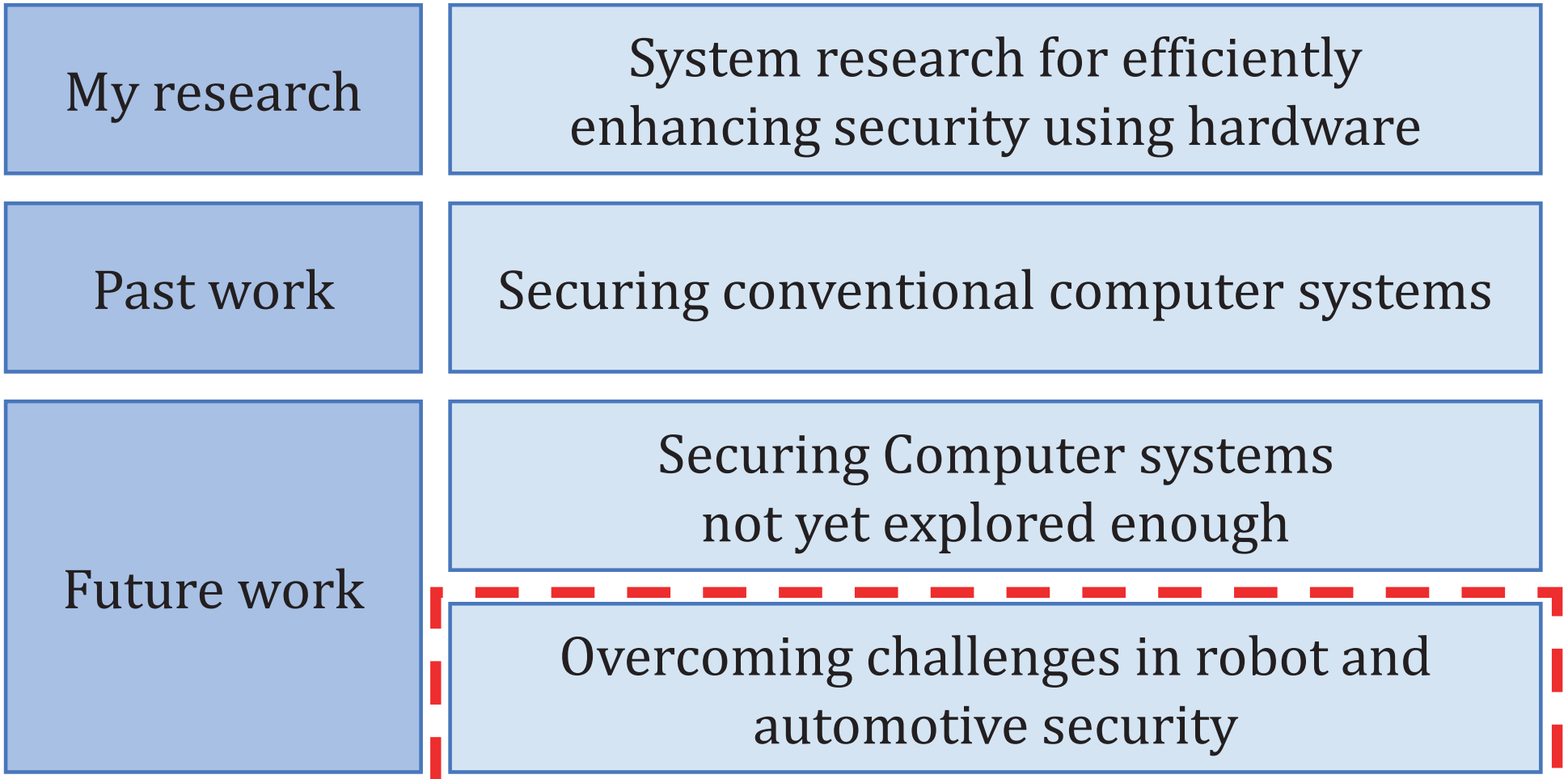
- STMicroelectronics 1Gbps Ethernet driver (stmicro)
  - Iperf3 (TCP transmit and receive bandwidth)
  - Explore several optimization techniques for SFITAG
    - SFITAG-naïve: Tagging all sub-objects of an object
    - SFIKE-opt1: only the sub-objects actively used by the driver are tagged
    - SFIKE-opt2: the driver private objects are not tagged
    - SFITAG: full utilization of optimizations



- Transmit path
  - SFITAG_naïve (5.6% degradation)
  - SFITAG_opt1 (4.4%)
  - SFITAG_opt2 (1.9%)
  - SFITAG (1%)

- Receive path
  - SFITAG_naïve (3% degradation)
  - SFITAG_opt1 (1.9%)
  - SFITAG_opt2 (1.1%)
  - SFITAG (0.8%)

Future work


My research
System research for efficiently enhancing security using hardware
Past work
Securing conventional computer systems
Future work
Securing Computer systems not yet explored enough
Overcoming challenges in robot and automotive security
Overcoming challenges in automotive security

Thank you.