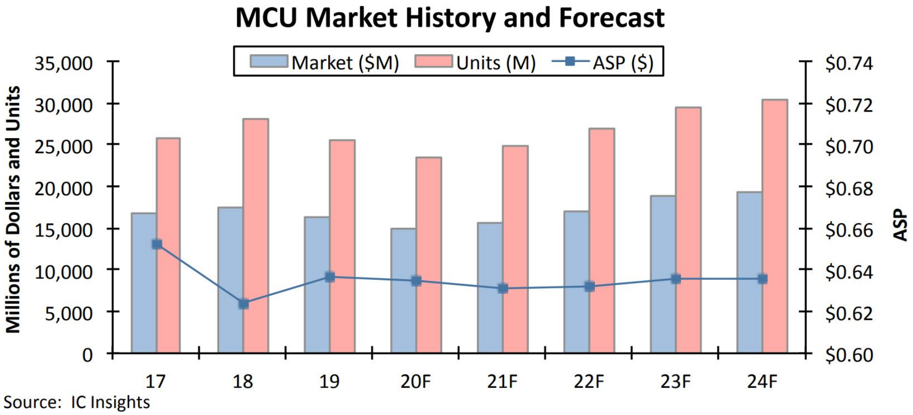
SHERLOC: Secure and Holistic Control-Flow Violation Detection on Embedded Systems

Xi Tan and Ziming Zhao
CyberspACe securiTy and forensIcs lab (CactiLab)
University at Buffalo

Microcontroller-based Embedded Systems



Source: IC Insights

It is estimated that the world has over 250 billion microcontrollers [1].

More than 4.4 billion Cortex-M MCUs were shipped in the 4th quarter of 2020 alone [2].

All images on this page are from the Internet.

[1] David, R., et al, (2021). Tensorflow lite micro: Embedded machine learning for TinyML systems. *Proceedings of Machine Learning and Systems*, *3*, 800-811.
[2] www.arm.com/company/news/2021/02/arm-ecosystem-ships-record-6-billion-arm-based-chips-in-a-single-quarter.

Microcontroller-based System Characteristics

Hardware (Cortex-M as an example)

RISC architecture. Sixteen 32-bit general-purpose registers.

No MMU, but a 32-bit physical memory space.

Unprivileged memory access instructions, pointer authentication code, streamlined TrustZone.

Software

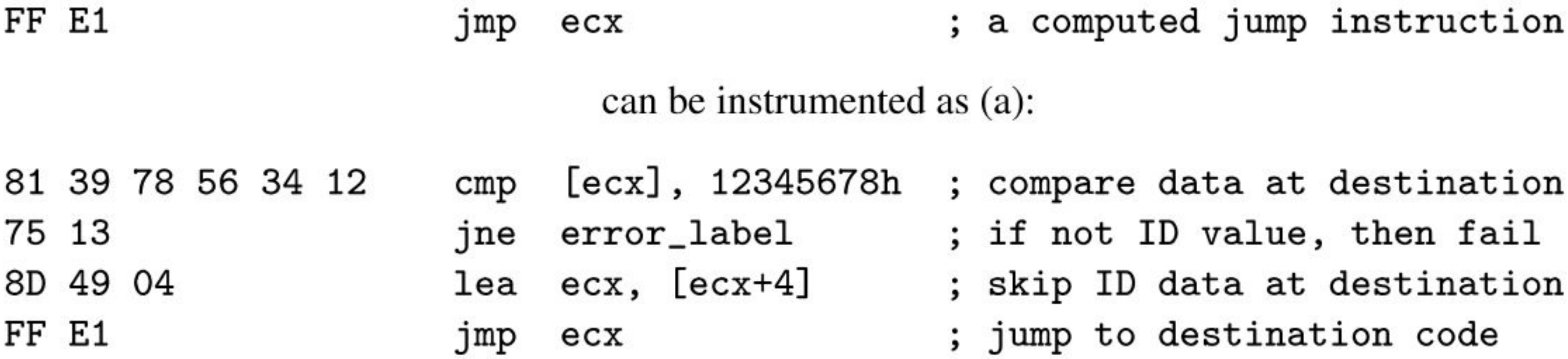
Developed in memory-unsafe languages, e.g., C

Most systems do not adopt privilege separation.

Functionality implemented in Interrupt Service Routines (ISR)

Control-Flow Integrity (CFI)
Inlined CFI Enforcement

Instrument at source code or binary level



```
FF E1                  jmp  ecx                ; a computed jump instruction
```

can be instrumented as (a):

```
81 39 78 56 34 12      cmp  [ecx], 12345678h   ; compare data at destination
75 13                  jne  error_label        ; if not ID value, then fail
8D 49 04               lea  ecx, [ecx+4]       ; skip ID data at destination
FF E1                  jmp  ecx                ; jump to destination code
```

Example CFI instrumentations of an x86 computed jump instruction [1]

[1] Erlingsson, M. A. M. B. U., & Jigatti, J. Control-flow integrity. ACM conference on Computer and communications security (CCS) 2005.

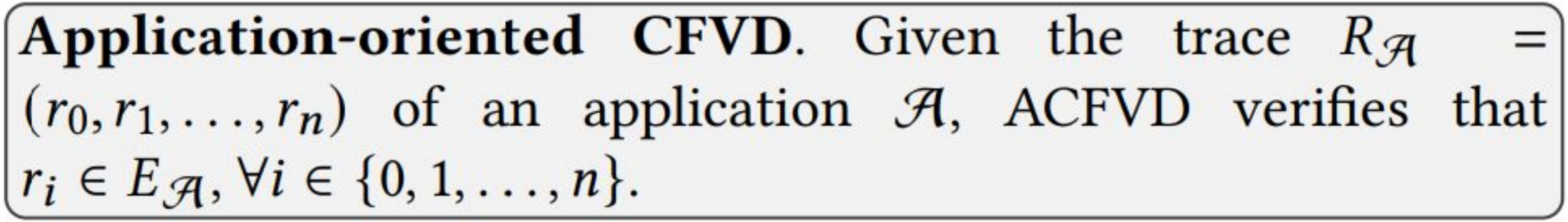
Inlined CFI Enforcement for Microcontroller Systems

- Memory constraints
  - Change the memory layout of the code
  - Increase the code size
- Coarse-grained forward-edge protection: label-based
- Shadow stacks need to be protected
- Existing approaches, e.g., CFICare [1], TZmCFI [2], utilize TrustZone to secure shadow stack but introduce a high run-time overhead,

[1] Nyman, T., Ekberg, J. E., Davi, L., & Asokan, N. (2017). CFI CaRE: Hardware-supported call and return enforcement for commercial microcontrollers. In Research in Attacks, Intrusions, and Defenses: 20th International Symposium, RAID 2017.
[2] Kawada, T., Honda, S., Matsubara, Y., & Takada, H. (2021). TZmCFI: RTOS-aware control-flow integrity using trustzone for Armv8-M. International Journal of Parallel Programming.

Control-Flow Violation Detection (CFVD)

- Do not instrument code but verify instruction trace generated by a hardware tracer [1, 2]
- Require kernel modification; kernel is in the TCB
- Only work on unprivileged application but not kernel



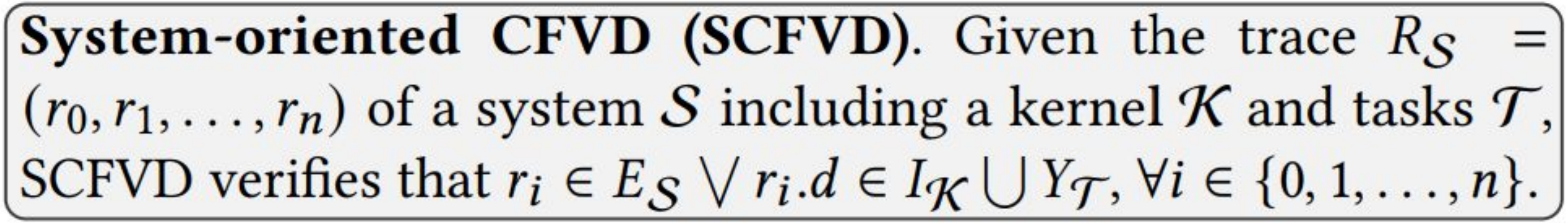
Application-oriented CFVD. Given the trace $R_{\mathcal{A}} = (r_0, r_1, \ldots, r_n)$ of an application $\mathcal{A}$, ACFVD verifies that $r_i \in E_{\mathcal{A}}, \forall i \in \{0, 1, \ldots, n\}$.

[1] Xinyang Ge, Weidong Cui, and Trent Jaeger. 2017. Griffin: Guarding control flows using intel processor trace. ACM SIGPLAN Notices

[2] Yufei Gu, Qingchuan Zhao, Yinqian Zhang, and Zhiqiang Lin. 2017. PT-CFI: Transparent backward-edge CFVD using intel processor trace. In ACM on Conference on Data and Application Security and Privacy (CODASPY).

System-oriented CFVD

- Most MCU functionalities are implemented in Interrupt Service Routines (ISR).
- Scheduling- and interrupt-aware



System-oriented CFVD (SCFVD). Given the trace $R_{\mathcal{S}} = (r_0, r_1, \ldots, r_n)$ of a system $\mathcal{S}$ including a kernel $\mathcal{K}$ and tasks $\mathcal{T}$, SCFVD verifies that $r_i \in E_{\mathcal{S}} \bigvee r_i.d \in I_{\mathcal{K}} \bigcup Y_{\mathcal{T}}, \forall i \in \{0, 1, \ldots, n\}$.

Challenges for SCFVD

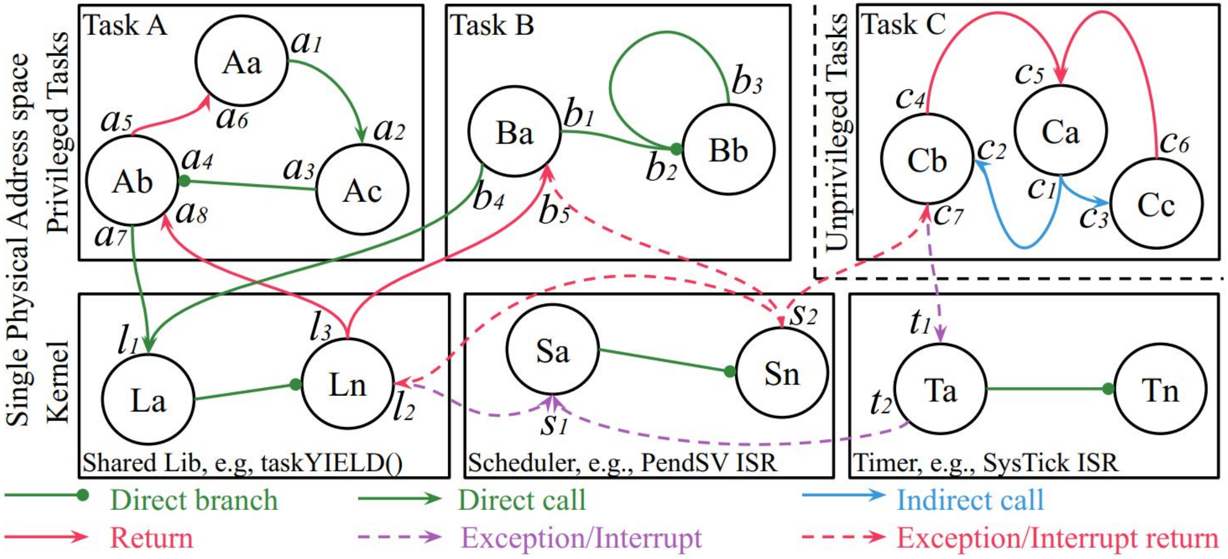


Figure 1: Example legitimate control-flow transfers of a system with an RTOS kernel, two privileged tasks, and one unprivileged task in a single physical address space.

Interrupts that cannot be predetermined, E.g., <c7, t1>

Challenges for SCFVD

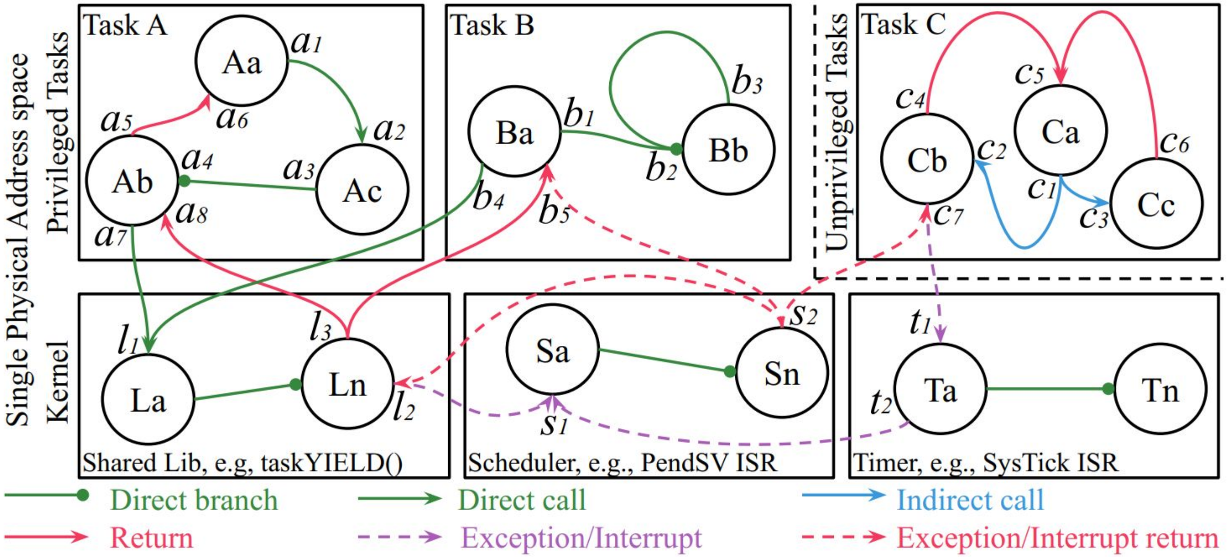


Figure 1: Example legitimate control-flow transfers of a system with an RTOS kernel, two privileged tasks, and one unprivileged task in a single physical address space.

The locations the scheduler yield control to cannot be predetermined, E.g., $\langle s2, b5\rangle$ and $\langle s2, c7\rangle$

Challenges for SCFVD

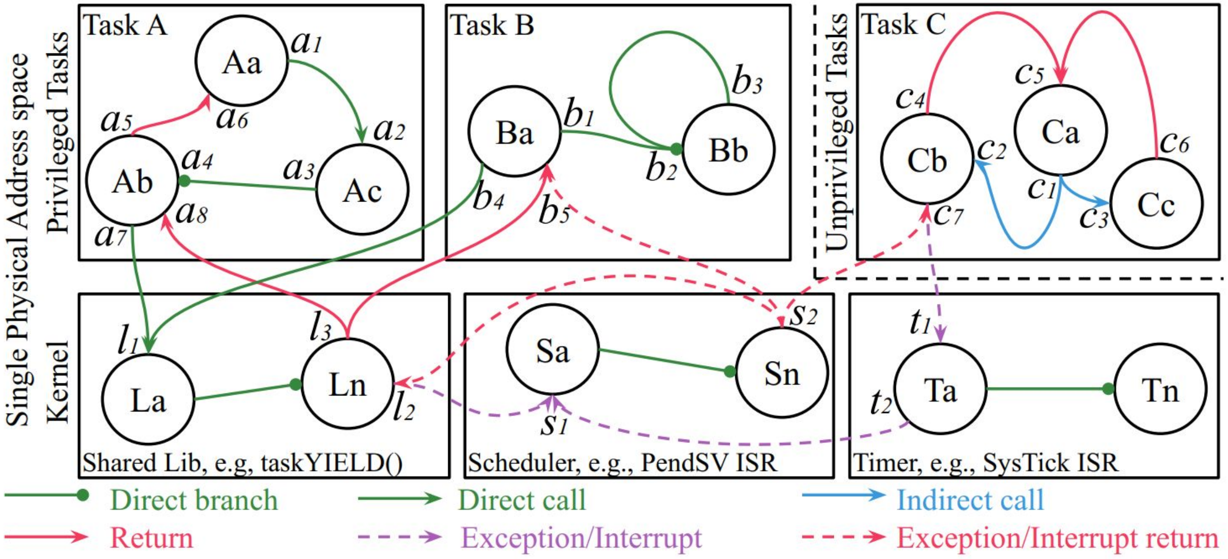


Figure 1: Example legitimate control-flow transfers of a system with an RTOS kernel, two privileged tasks, and one unprivileged task in a single physical address space.

The trace, tracing operation, and CF verification must be secured from the privileged but potentially compromised kernel.

Sherloc Design

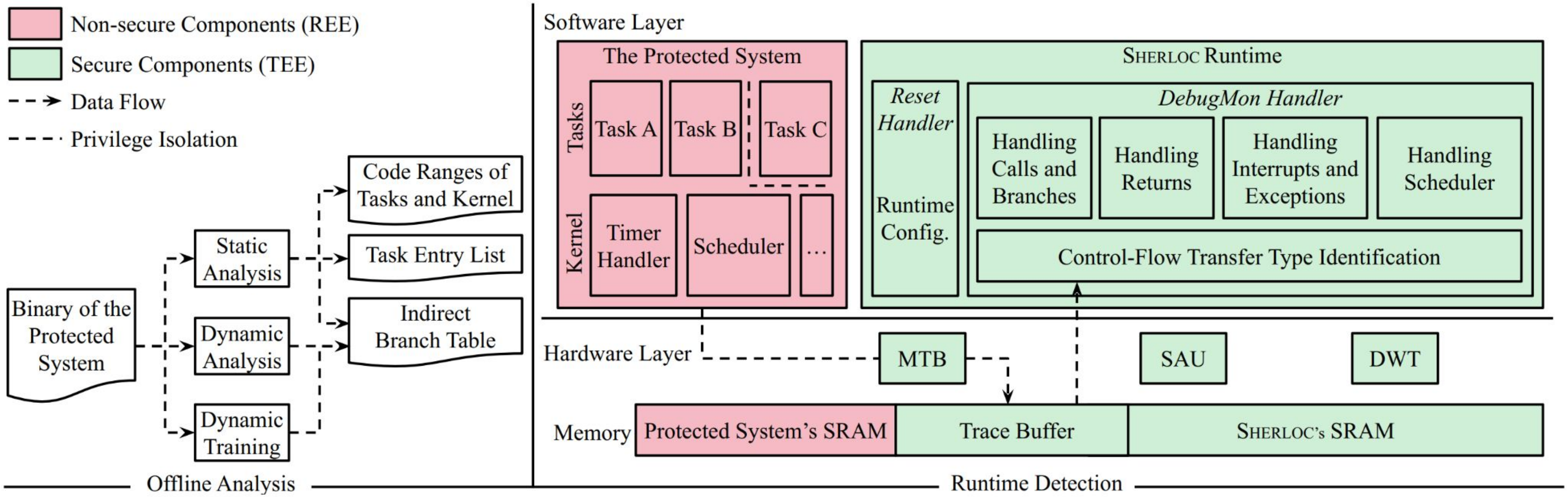
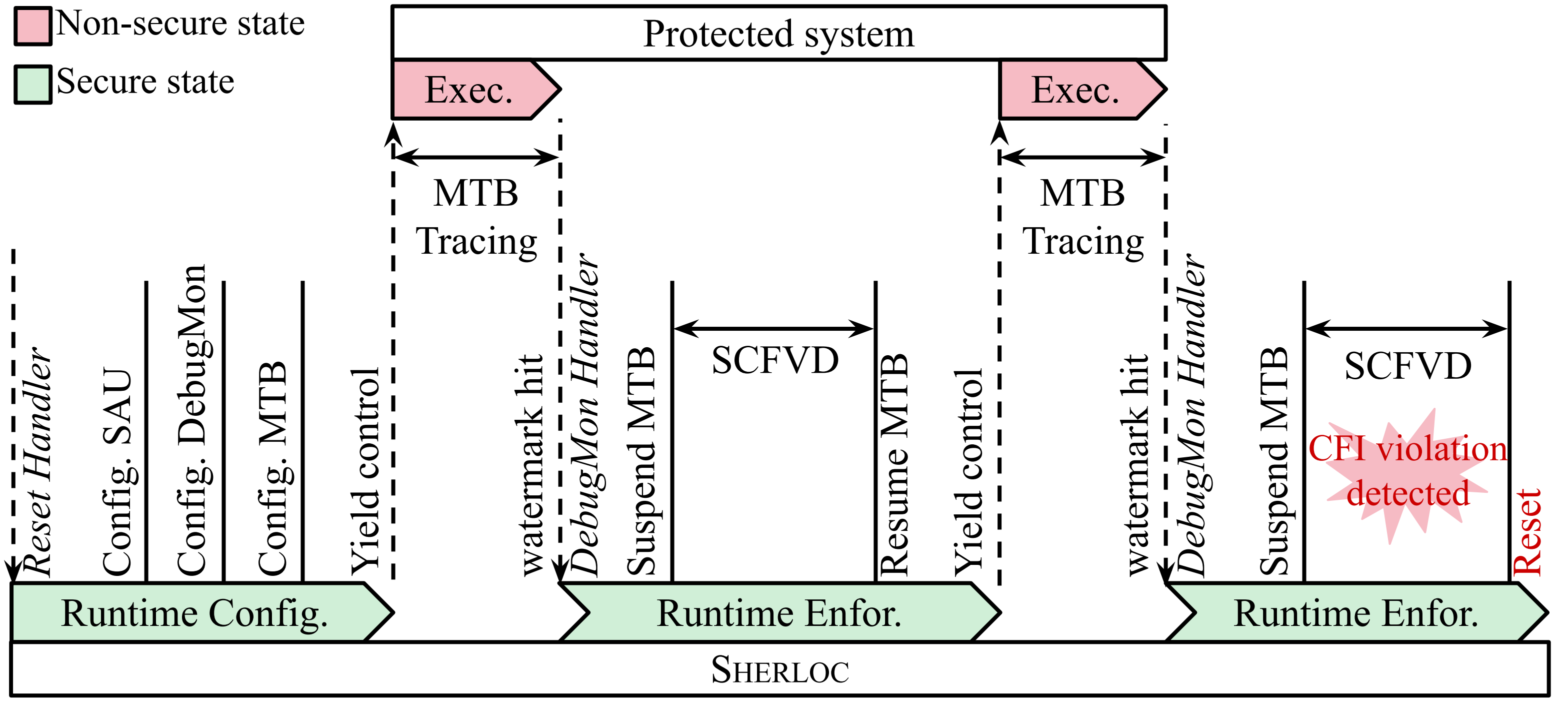
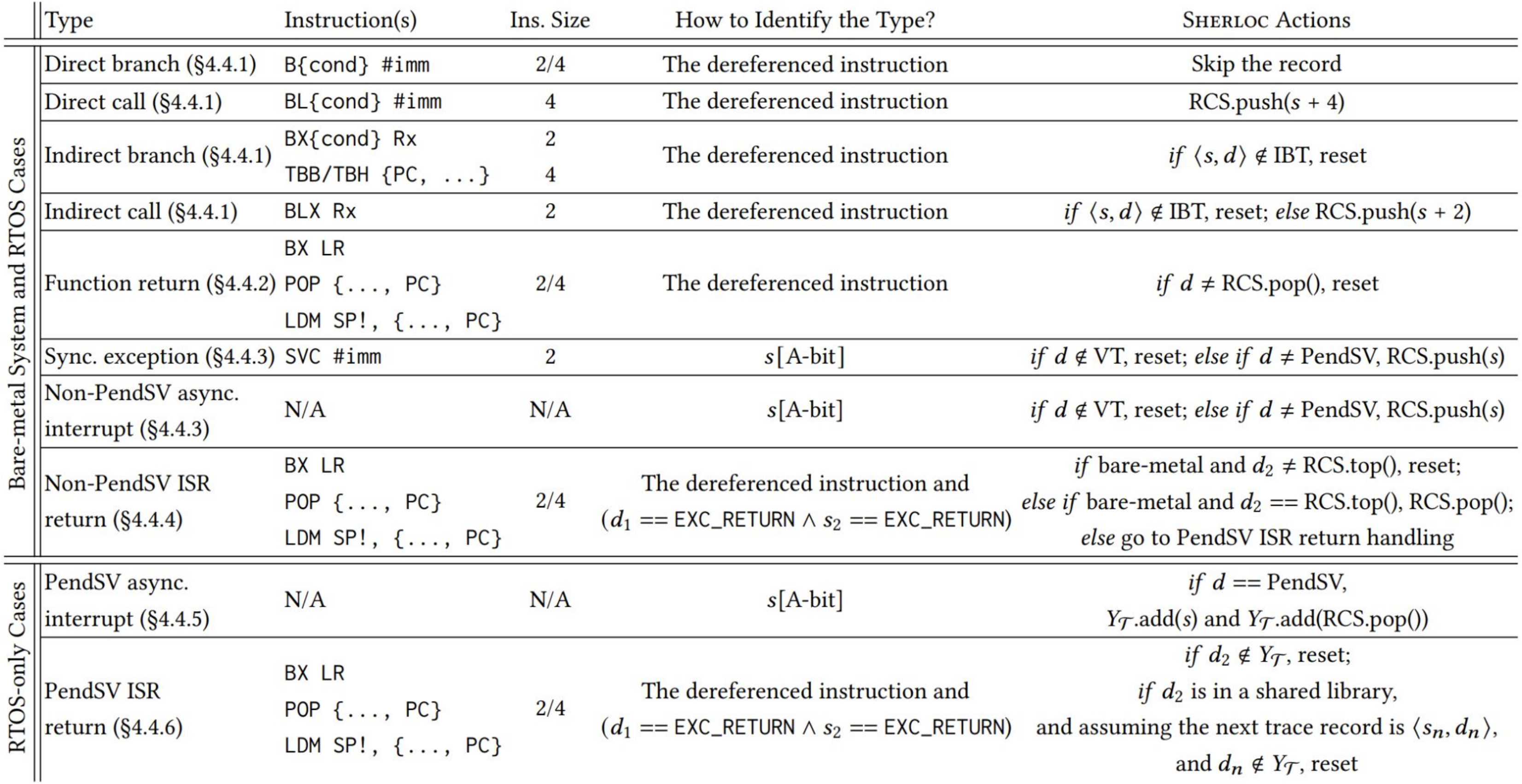


Figure 2: SHERLOC comprises offline analysis and runtime configuration and enforcement modules. The unmodified protected system program runs in the non-secure state, whereas SHERLOC runtime modules execute in the secure state.

Sherloc Timeline



Sherloc Holistic Enforcement Policy



	Type	Instruction(s)	Ins. Size	How to Identify the Type?	Sherloc Actions
Bare-metal System and RTOS Cases	Direct branch (§4.4.1)	`B{cond} #imm`	2/4	The dereferenced instruction	Skip the record
	Direct call (§4.4.1)	`BL{cond} #imm`	4	The dereferenced instruction	RCS.push(s + 4)

	Indirect branch (§4.4.1)	`BX{cond} Rx`
		`TBB/TBH {PC, ...}`
			2
			4
				The dereferenced instruction	*if* $\langle s, d\rangle \notin$ IBT, reset

	Indirect call (§4.4.1)	`BLX Rx`	2	The dereferenced instruction	*if* $\langle s, d\rangle \notin$ IBT, reset; *else* RCS.push(s + 2)

	Function return (§4.4.2)	`BX LR`
		`POP {..., PC}`
		`LDM SP!, {..., PC}`
			2/4	The dereferenced instruction	*if* $d \neq$ RCS.pop(), reset

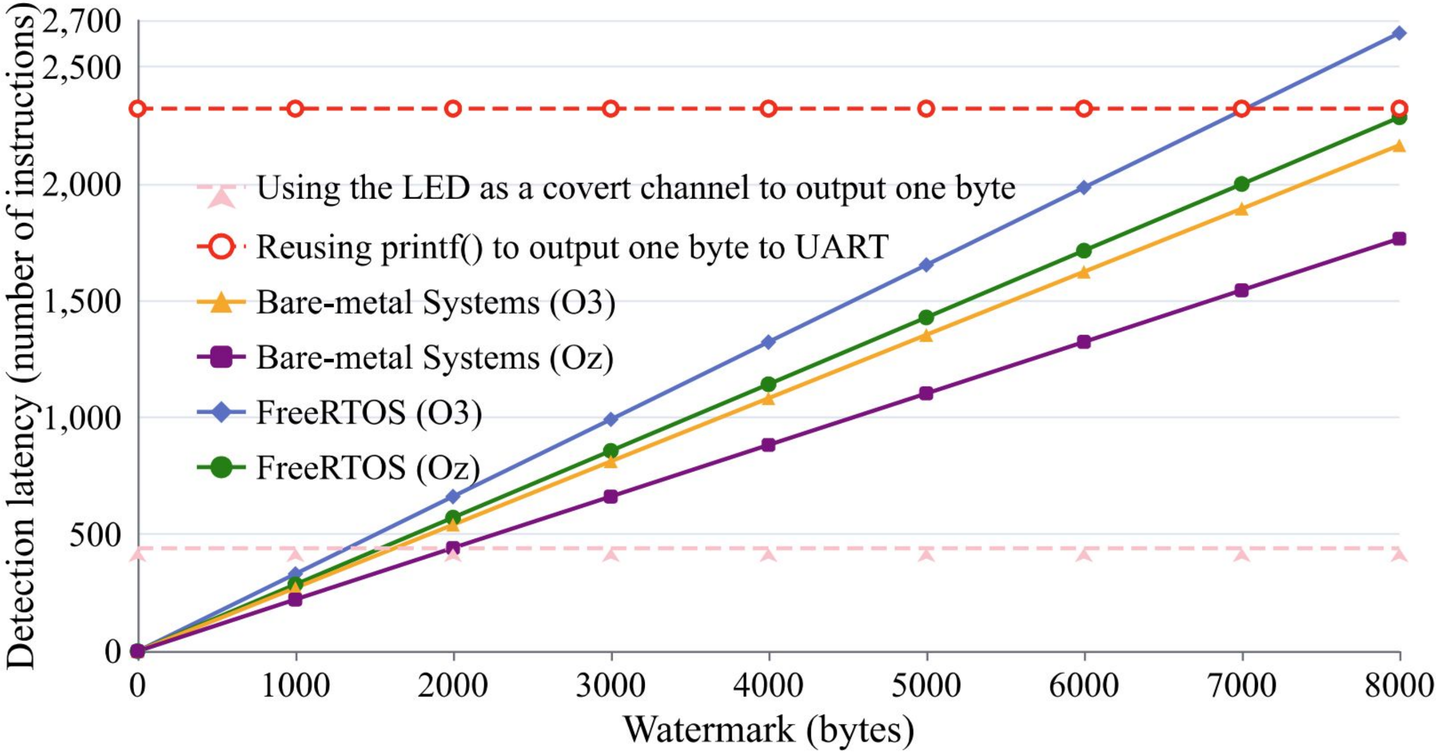
	Sync. exception (§4.4.3)	`SVC #imm`	2	s[A-bit]	*if* $d \notin$ VT, reset; *else if* $d \neq$ PendSV, RCS.push(s)
	Non-PendSV async. interrupt (§4.4.3)	N/A	N/A	s[A-bit]	*if* $d \notin$ VT, reset; *else if* $d \neq$ PendSV, RCS.push(s)

	Non-PendSV ISR return (§4.4.4)	`BX LR`
		`POP {..., PC}`
		`LDM SP!, {..., PC}`
			2/4	The dereferenced instruction and
				(d_1 == EXC_RETURN $\wedge$ s_2 == EXC_RETURN)
					if bare-metal and $d_2 \neq$ RCS.top(), reset;
					else if bare-metal and d_2 == RCS.top(), RCS.pop();
					else go to PendSV ISR return handling

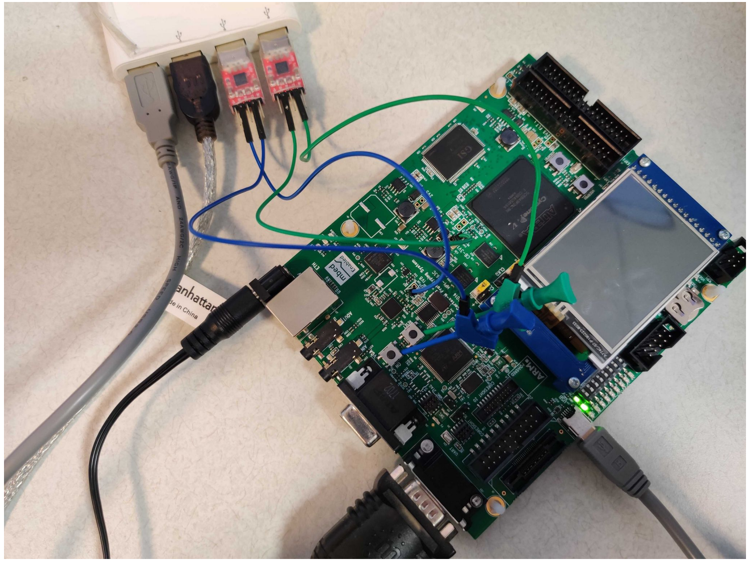
RTOS-only Cases	PendSV async. interrupt (§4.4.5)	N/A	N/A	s[A-bit]	*if* d == PendSV, $Y_{\mathcal{T}}$.add(s) and $Y_{\mathcal{T}}$.add(RCS.pop())

	PendSV ISR return (§4.4.6)	`BX LR`
		`POP {..., PC}`
		`LDM SP!, {..., PC}`
			2/4	The dereferenced instruction and
				(d_1 == EXC_RETURN $\wedge$ s_2 == EXC_RETURN)
					if $d_2 \notin Y_{\mathcal{T}}$, reset;
					if d_2 is in a shared library,
					and assuming the next trace record is $\langle s_n, d_n\rangle$,
					and $d_n \notin Y_{\mathcal{T}}$, reset

Security Analysis: Latency Estimation

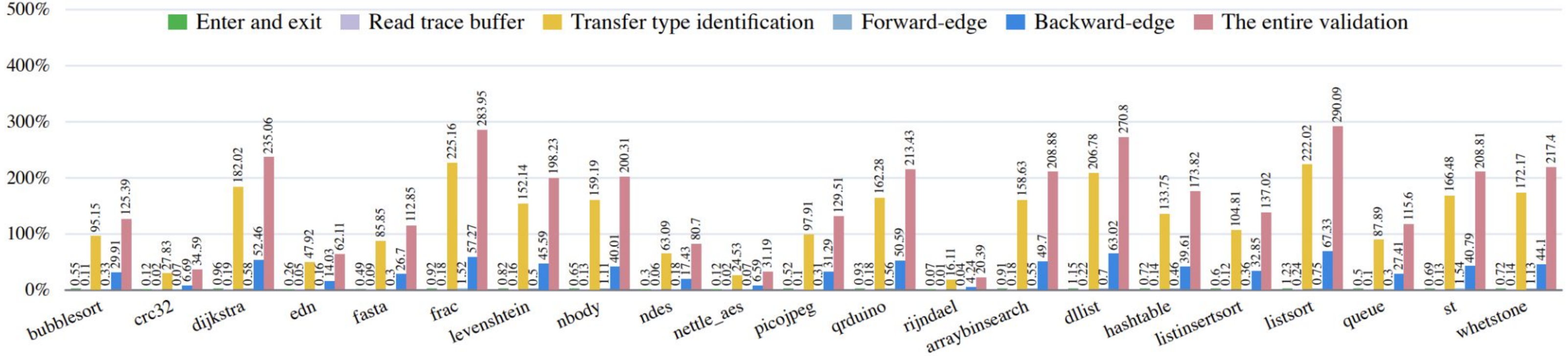


Experiment

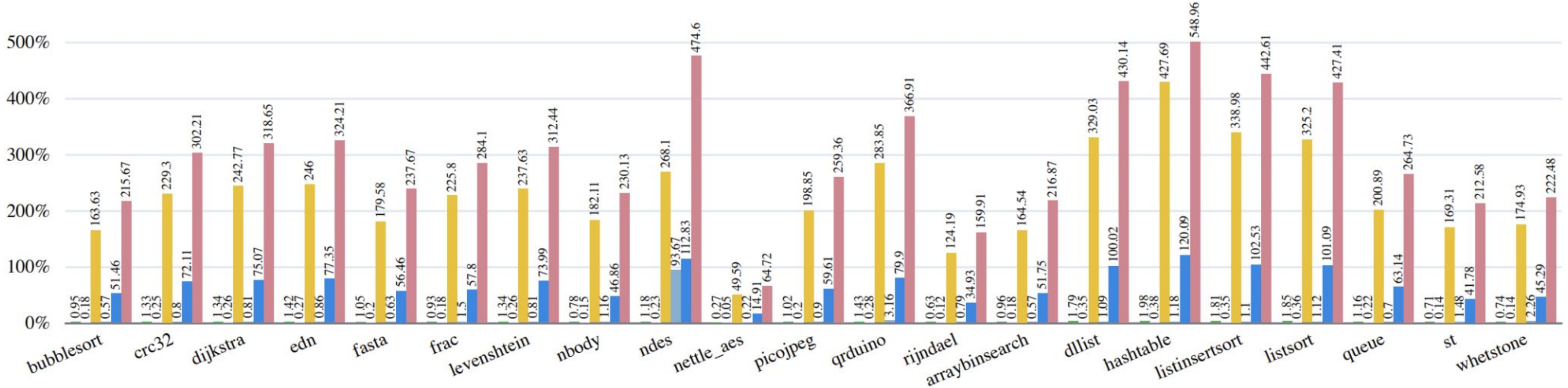


- Prototype: ARM Versatile Express Cortex-M Prototyping System MPS2+ configured as a Cortex-M33 CPU
- Benchmark:
  - BEEBS
  - Blinky
  - FreeRTOS
- Optimization level: -O3 and -Oz

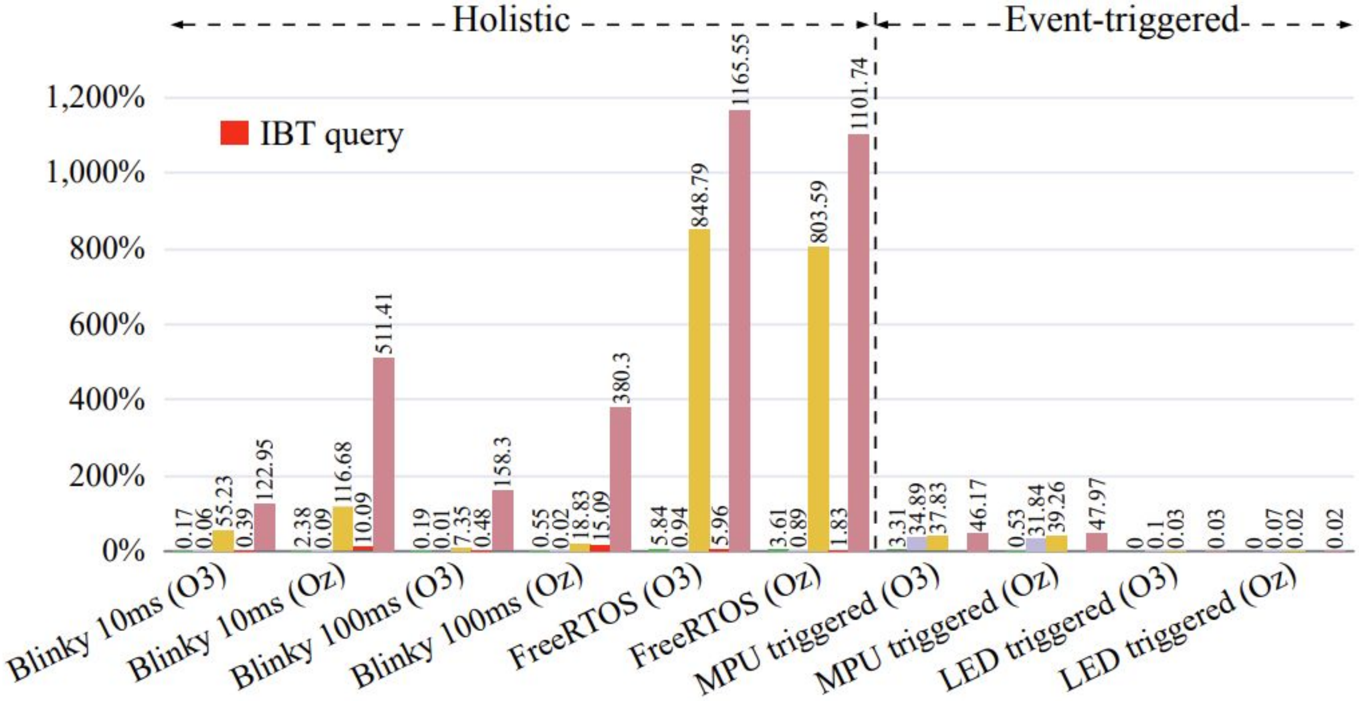
Performance Evaluation - BEEBS



(a) Performance overhead (%) of BEEBS programs compiled with O3 optimization.

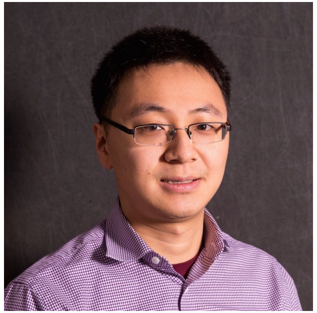


Performance Evaluation - BEEBS



Thank you!

Xi Tan



Ziming Zhao

Open-sourced at:

https://github.com/CactiLab/Sherloc-Cortex-M-CFVD

Control-Flow Integrity (CFI)

Control-flow integrity (CFI) is a security property that can prevent control-flow hijacking by dictating that indirect control-flow transfers, including forward edges (indirect call and branch) and backward edges (return), must follow a predetermined control-flow graph (CFG).

[1] Erlingsson, M. A. M. B. U., & Jigatti, J. Control-flow integrity. ACM conference on Computer and communications security (CCS) 2005.