SoK: Where's the "up"?! A Comprehensive (bottom-up) Study on the Security of Arm Cortex-M Systems

Xi Tan[†‡], Zheyuan Ma[†‡], Sandro Pinto[*], Le Guan[⋆], Ning Zhang[§], Jun Xu[∘], Zhiqiang Lin[⋄], Hongxing Hu[†], Ziming Zhao[†‡]

[†]*University at Buffalo* [*]*Universidade do Minho* [⋆]*University of Georgia*
[§]*Washington University in St. Louis* [∘]*University of Utah* [⋄]*Ohio State University* [‡]*CactiLab*
{xitan, zheyuanm, hongxinh, zimingzh}@buffalo.edu, sandro.pinto@dei.uminho.pt, leguan@uga.edu, zhang.ning@wustl.edu, junxzm@cs.utah.edu, zlin@cse.ohio-state.edu

Cortex-M MCUs

Figure from: https://gsasindia.com/blog/cortex-a-cortex-r-and-cortex-m/

Research Questions

Q1: Hardware

What are the security features, limitations, and issues at the Cortex-M microarchitecture, instruction set architecture (ISA), and beyond?

Q2: Software

What are the security mechanisms and flaws of Cortex-M based software systems?

Q3: Implementation

What are the nature and severity of the publicly disclosed vulnerabilities in the Cortex-M based software systems?

Q4: Research

What defenses for Cortex-M systems have been explored in the literature, and what are their limitations?

Methodology

3000+

Pages

Analyzing hardware primitives and offerings from official documents

7000+

Firmware

Collecting and analyzing real-world firmware

500+

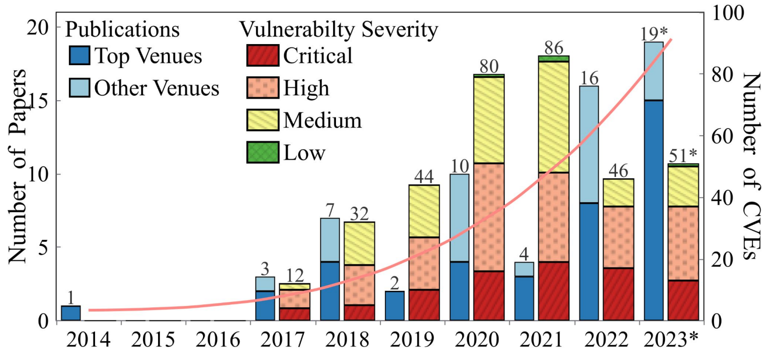
CVEs

Spanning nearly six years, classification

50+

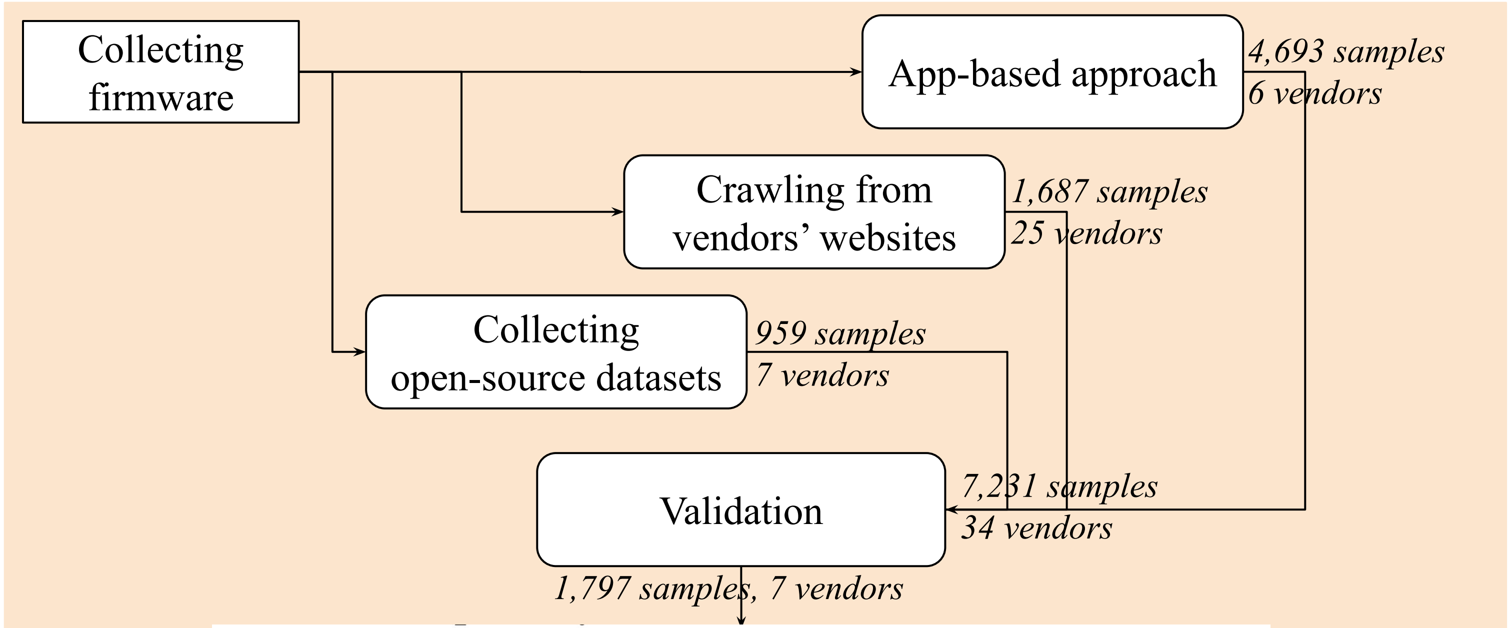
Pubs

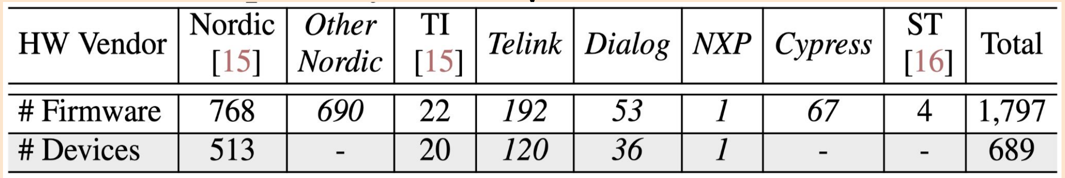
From systems and security conferences



Related paper and CVE growing trends from year 2014 to 2023

Firmware Collection





HW Vendor	Nordic [15]	*Other Nordic*	TI [15]	*Telink*	*Dialog*	*NXP*	*Cypress*	ST [16]	Total
# Firmware	768	*690*	22	*192*	*53*	*1*	*67*	4	1,797
# Devices	513	-	20	*120*	*36*	*1*	-	-	689

“

Answer the questions!

Hardware Limitations & Issues (Answers to Q1)

- ➔ Lack of Memory Protection Mechanisms
  - ◆ E.g., **No MMU/IOMMU**, a small number of MPU regions and limited sizes
- ➔ Inherited limitations from Cortex-A
  - ◆ E.g., No intrinsic encryption to protect the secure state memory
- ➔ Vendor-Agnostic ISA Issues
  - ◆ E.g., Fast state switch mechanism exploitable for **privilege escalation** [1]
- ➔ …

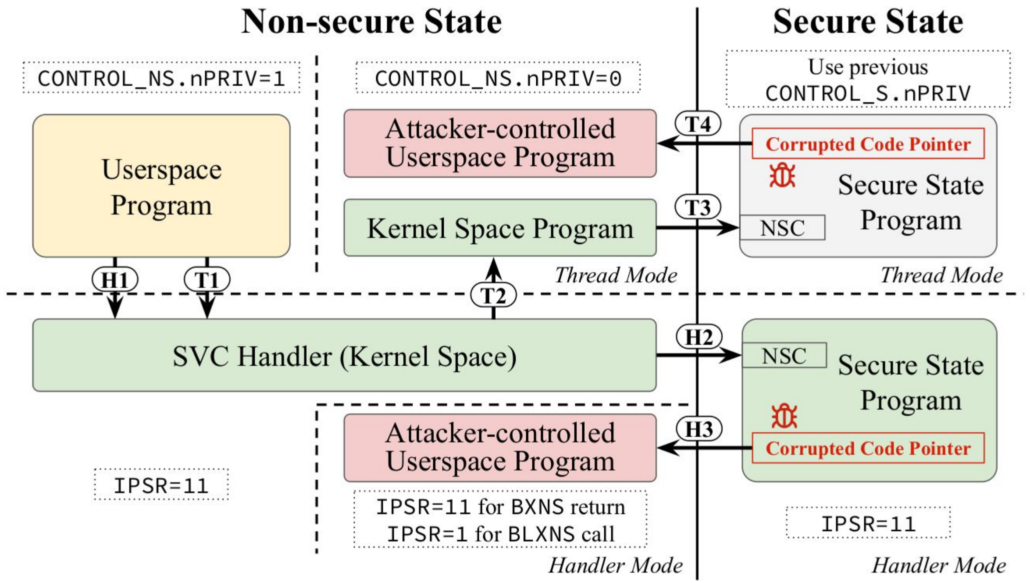
[1] Ma, Z., Tan, X., Ziarek, L., Zhang, N., Hu, H. and Zhao, Z., 2023, July. Return-to-Non-Secure Vulnerabilities on ARM Cortex-M TrustZone: Attack and Defense. In 2023 60th ACM/IEEE Design Automation Conference (DAC) (pp. 1-6). IEEE.

Our Discovery: ret2ns Attacks

Appeared at Design Automation Conferences (**DAC**) 2023 [1]

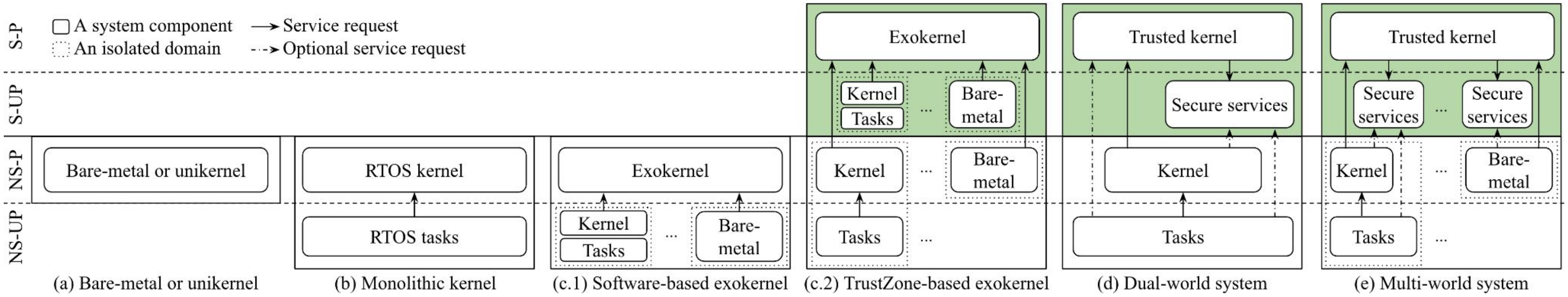
Open-Source: https://github.com/CactiLab/ret2ns-Cortex-M-TrustZone

Streamlined TrustZone design on Cortex-M introduces new attack surfaces: Return-to-Non-Secure Vulnerabilities (ret2ns)



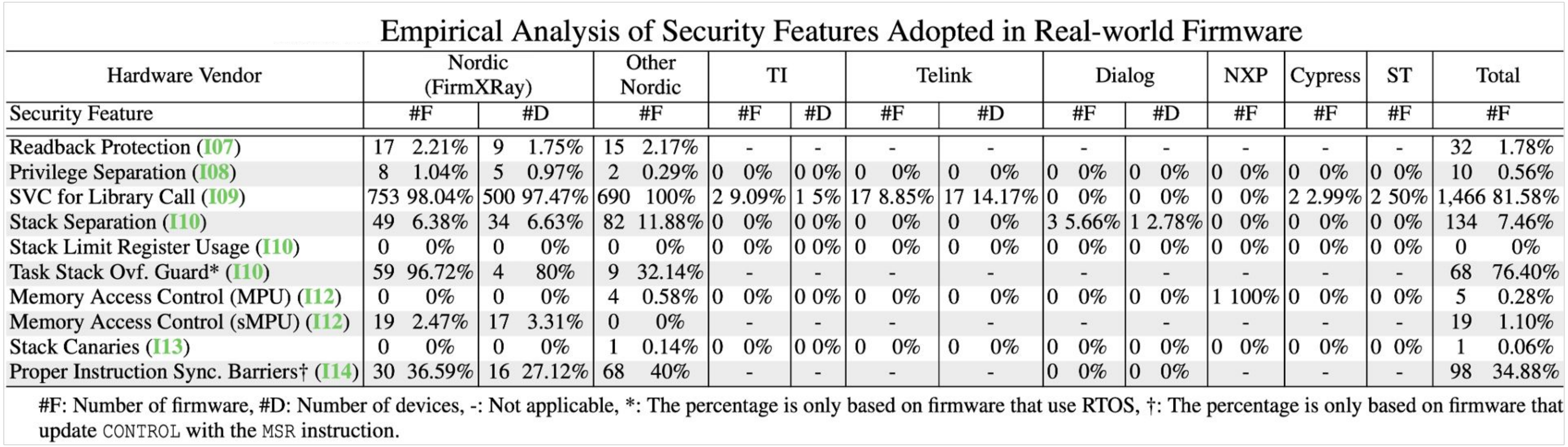
[1] Ma, Z., Tan, X., Ziarek, L., Zhang, N., Hu, H. and Zhao, Z., 2023, July. Return-to-Non-Secure Vulnerabilities on ARM Cortex-M TrustZone: Attack and Defense. In 2023 60th ACM/IEEE Design Automation Conference (DAC) (pp. 1-6). IEEE.

Software Architectures (Answers to Q2)



Despite the research progress towards more secure architectures **(c, d, and e)** for Cortex-M systems, a large number of the real-world firmware (99.44%) in our dataset are simply **bare-metal systems and unikernels (a),**
while 0.56% of the firmware in our dataset fall into **monolithic kernels (b)**.

Software Architectural Issues (Answers to Q2)

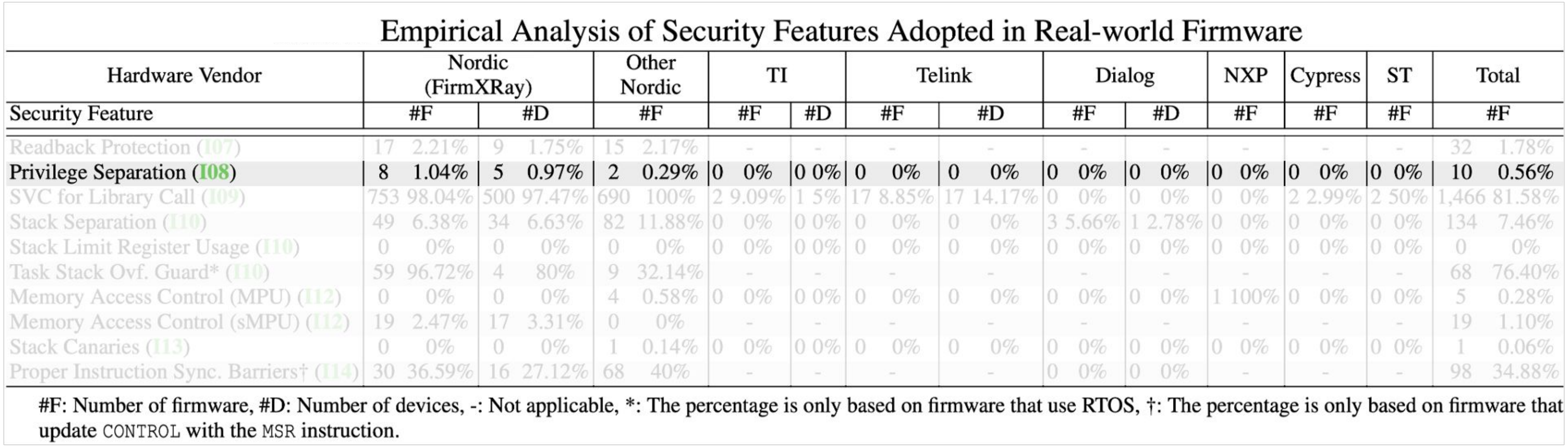


Empirical Analysis of Security Features Adopted in Real-world Firmware

Hardware Vendor	Nordic (FirmXRay)		Other Nordic	TI		Telink		Dialog		NXP	Cypress	ST	Total
Security Feature	#F	#D	#F	#F	#D	#F	#D	#F	#D	#F	#F	#F	#F
Readback Protection (I07)	17 2.21%	9 1.75%	15 2.17%	-	-	-	-	-	-	-	-	-	32 1.78%
Privilege Separation (I08)	8 1.04%	5 0.97%	2 0.29%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	10 0.56%
SVC for Library Call (I09)	753 98.04%	500 97.47%	690 100%	2 9.09%	1 5%	17 8.85%	17 14.17%	0 0%	0 0%	0 0%	2 2.99%	2 50%	1,466 81.58%
Stack Separation (I10)	49 6.38%	34 6.63%	82 11.88%	0 0%	0 0%	0 0%	0 0%	3 5.66%	1 2.78%	0 0%	0 0%	0 0%	134 7.46%
Stack Limit Register Usage (I10)	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%
Task Stack Ovf. Guard* (I10)	59 96.72%	4 80%	9 32.14%	-	-	-	-	-	-	-	-	-	68 76.40%
Memory Access Control (MPU) (I12)	0 0%	0 0%	4 0.58%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	1 100%	0 0%	0 0%	5 0.28%
Memory Access Control (sMPU) (I12)	19 2.47%	17 3.31%	0 0%	-	-	-	-	-	-	-	-	-	19 1.10%
Stack Canaries (I13)	0 0%	0 0%	1 0.14%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	1 0.06%
Proper Instruction Sync. Barriers† (I14)	30 36.59%	16 27.12%	68 40%	-	-	-	-	0 0%	0 0%	-	-	-	98 34.88%

#F: Number of firmware, #D: Number of devices, -: Not applicable, *: The percentage is only based on firmware that use RTOS, †: The percentage is only based on firmware that update CONTROL with the MSR instruction.

Software Architectural Issues (Answers to Q2)



Empirical Analysis of Security Features Adopted in Real-world Firmware

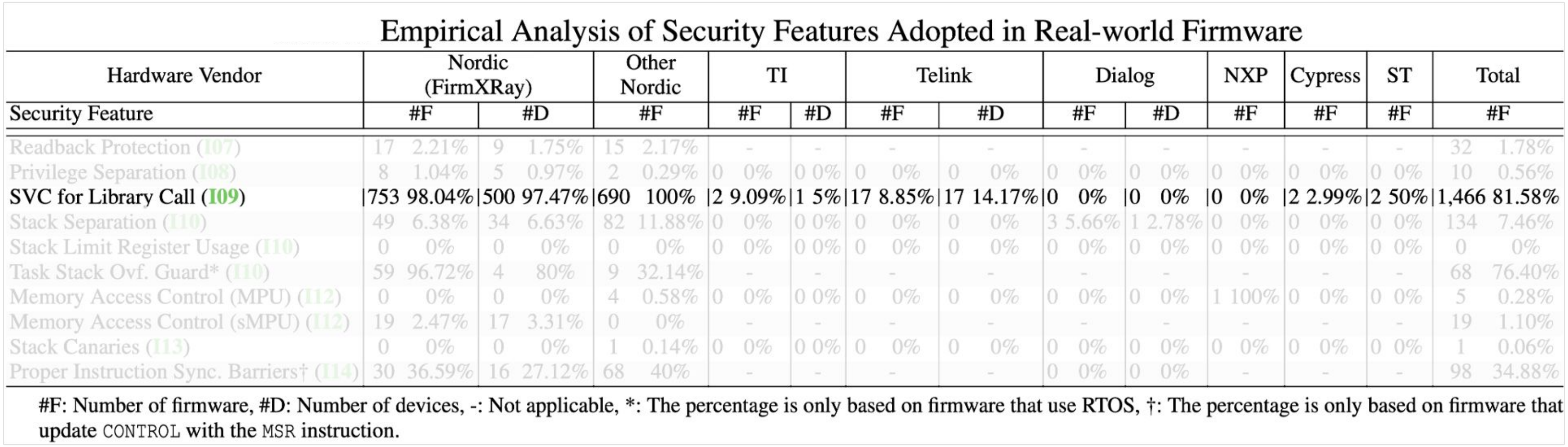
Hardware Vendor	Nordic (FirmXRay)		Other Nordic	TI		Telink		Dialog		NXP	Cypress	ST	Total
Security Feature	#F	#D	#F	#F	#D	#F	#D	#F	#D	#F	#F	#F	#F
Readback Protection (I07)	17 2.21%	9 1.75%	15 2.17%	-	-	-	-	-	-	-	-	-	32 1.78%
Privilege Separation (I08)	8 1.04%	5 0.97%	2 0.29%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	10 0.56%
SVC for Library Call (I09)	753 98.04%	500 97.47%	690 100%	2 9.09%	1 5%	17 8.85%	17 14.17%	0 0%	0 0%	0 0%	2 2.99%	2 50%	1,466 81.58%
Stack Separation (I10)	49 6.38%	34 6.63%	82 11.88%	0 0%	0 0%	0 0%	0 0%	3 5.66%	1 2.78%	0 0%	0 0%	0 0%	134 7.46%
Stack Limit Register Usage (I10)	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%
Task Stack Ovf. Guard* (I10)	59 96.72%	4 80%	9 32.14%	-	-	-	-	-	-	-	-	-	68 76.40%
Memory Access Control (MPU) (I12)	0 0%	0 0%	4 0.58%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	1 100%	0 0%	0 0%	5 0.28%
Memory Access Control (sMPU) (I12)	19 2.47%	17 3.31%	0 0%	-	-	-	-	-	-	-	-	-	19 1.10%
Stack Canaries (I13)	0 0%	0 0%	1 0.14%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	1 0.06%
Proper Instruction Sync. Barriers† (I14)	30 36.59%	16 27.12%	68 40%	-	-	-	-	0 0%	0 0%	-	-	-	98 34.88%

#F: Number of firmware, #D: Number of devices, -: Not applicable, *: The percentage is only based on firmware that use RTOS, †: The percentage is only based on firmware that update `CONTROL` with the `MSR` instruction.

Privilege separation is **seldom** used

Do we really need privilege separation?

Software Architectural Issues (Answers to Q2)



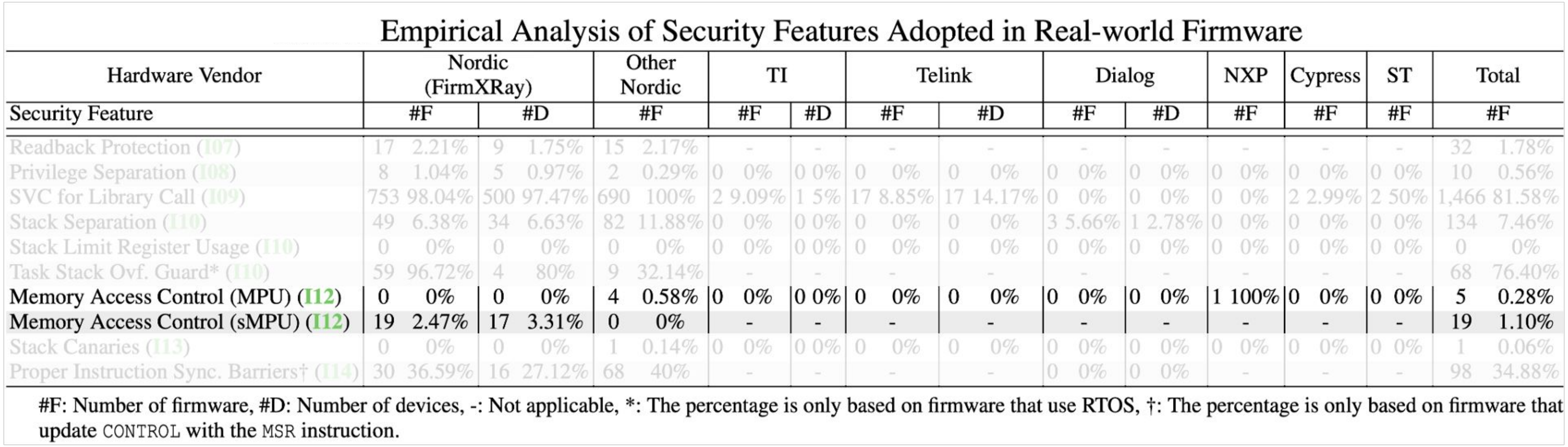
Empirical Analysis of Security Features Adopted in Real-world Firmware

Hardware Vendor	Nordic (FirmXRay)		Other Nordic	TI		Telink		Dialog		NXP	Cypress	ST	Total
Security Feature	#F	#D	#F	#F	#D	#F	#D	#F	#D	#F	#F	#F	#F
Readback Protection (I07)	17 2.21%	9 1.75%	15 2.17%	-	-	-	-	-	-	-	-	-	32 1.78%
Privilege Separation (I08)	8 1.04%	5 0.97%	2 0.29%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	10 0.56%
SVC for Library Call (I09)	753 98.04%	500 97.47%	690 100%	2 9.09%	1 5%	17 8.85%	17 14.17%	0 0%	0 0%	0 0%	2 2.99%	2 50%	1,466 81.58%
Stack Separation (I10)	49 6.38%	34 6.63%	82 11.88%	0 0%	0 0%	0 0%	0 0%	3 5.66%	1 2.78%	0 0%	0 0%	0 0%	134 7.46%
Stack Limit Register Usage (I10)	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%
Task Stack Ovf. Guard* (I10)	59 96.72%	4 80%	9 32.14%	-	-	-	-	-	-	-	-	-	68 76.40%
Memory Access Control (MPU) (I12)	0 0%	0 0%	4 0.58%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	1 100%	0 0%	0 0%	5 0.28%
Memory Access Control (sMPU) (I12)	19 2.47%	17 3.31%	0 0%	-	-	-	-	-	-	-	-	-	19 1.10%
Stack Canaries (I13)	0 0%	0 0%	1 0.14%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	1 0.06%
Proper Instruction Sync. Barriers† (I14)	30 36.59%	16 27.12%	68 40%	-	-	-	-	0 0%	0 0%	-	-	-	98 34.88%

#F: Number of firmware, #D: Number of devices, -: Not applicable, *: The percentage is only based on firmware that use RTOS, †: The percentage is only based on firmware that update `CONTROL` with the `MSR` instruction.

Supervisor call (SVC) is used for **library call**, not privilege elevation

Software Architectural Issues (Answers to Q2)



Empirical Analysis of Security Features Adopted in Real-world Firmware

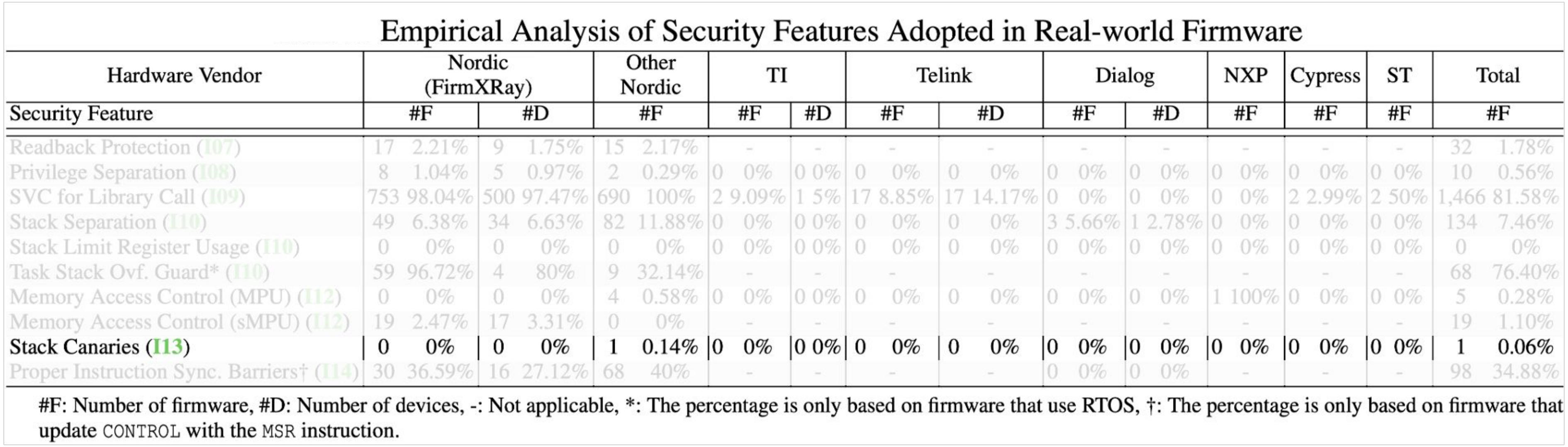
Hardware Vendor	Nordic (FirmXRay)		Other Nordic	TI		Telink		Dialog		NXP	Cypress	ST	Total
Security Feature	#F	#D	#F	#F	#D	#F	#D	#F	#D	#F	#F	#F	#F
Readback Protection (I07)	17 2.21%	9 1.75%	15 2.17%	-	-	-	-	-	-	-	-	-	32 1.78%
Privilege Separation (I08)	8 1.04%	5 0.97%	2 0.29%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	10 0.56%
SVC for Library Call (I09)	753 98.04%	500 97.47%	690 100%	2 9.09%	1 5%	17 8.85%	17 14.17%	0 0%	0 0%	0 0%	2 2.99%	2 50%	1,466 81.58%
Stack Separation (I10)	49 6.38%	34 6.63%	82 11.88%	0 0%	0 0%	0 0%	0 0%	3 5.66%	1 2.78%	0 0%	0 0%	0 0%	134 7.46%
Stack Limit Register Usage (I10)	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%
Task Stack Ovf. Guard* (I10)	59 96.72%	4 80%	9 32.14%	-	-	-	-	-	-	-	-	-	68 76.40%
Memory Access Control (MPU) (I12)	0 0%	0 0%	4 0.58%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	1 100%	0 0%	0 0%	5 0.28%
Memory Access Control (sMPU) (I12)	19 2.47%	17 3.31%	0 0%	-	-	-	-	-	-	-	-	-	19 1.10%
Stack Canaries (I13)	0 0%	0 0%	1 0.14%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	1 0.06%
Proper Instruction Sync. Barriers† (I14)	30 36.59%	16 27.12%	68 40%	-	-	-	-	0 0%	0 0%	-	-	-	98 34.88%

#F: Number of firmware, #D: Number of devices, -: Not applicable, *: The percentage is only based on firmware that use RTOS, †: The percentage is only based on firmware that update `CONTROL` with the `MSR` instruction.

No or weak memory access control; **executable stack**

vendor-specific implementation of MPU (**sMPU**)

Software Architectural Issues (Answers to Q2)



Empirical Analysis of Security Features Adopted in Real-world Firmware

Hardware Vendor	Nordic (FirmXRay)		Other Nordic	TI		Telink		Dialog		NXP	Cypress	ST	Total
Security Feature	#F	#D	#F	#F	#D	#F	#D	#F	#D	#F	#F	#F	#F
Readback Protection (I07)	17 2.21%	9 1.75%	15 2.17%	-	-	-	-	-	-	-	-	-	32 1.78%
Privilege Separation (I08)	8 1.04%	5 0.97%	2 0.29%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	10 0.56%
SVC for Library Call (I09)	753 98.04%	500 97.47%	690 100%	2 9.09%	1 5%	17 8.85%	17 14.17%	0 0%	0 0%	0 0%	2 2.99%	2 50%	1,466 81.58%
Stack Separation (I10)	49 6.38%	34 6.63%	82 11.88%	0 0%	0 0%	0 0%	0 0%	3 5.66%	1 2.78%	0 0%	0 0%	0 0%	134 7.46%
Stack Limit Register Usage (I10)	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%
Task Stack Ovf. Guard* (I10)	59 96.72%	4 80%	9 32.14%	-	-	-	-	-	-	-	-	-	68 76.40%
Memory Access Control (MPU) (I12)	0 0%	0 0%	4 0.58%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	1 100%	0 0%	0 0%	5 0.28%
Memory Access Control (sMPU) (I12)	19 2.47%	17 3.31%	0 0%	-	-	-	-	-	-	-	-	-	19 1.10%
Stack Canaries (I13)	0 0%	0 0%	1 0.14%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	1 0.06%
Proper Instruction Sync. Barriers† (I14)	30 36.59%	16 27.12%	68 40%	-	-	-	-	0 0%	0 0%	-	-	-	98 34.88%

#F: Number of firmware, #D: Number of devices, -: Not applicable, *: The percentage is only based on firmware that use RTOS, †: The percentage is only based on firmware that update `CONTROL` with the `MSR` instruction.

Stack canaries used to be effective and low cost on modern computers, while it is **rarely** shown on Cortex-M firmware [1]

[1] Tan, X., Mohan, S., Armanuzzaman, M., Ma, Z., Liu, G., Eastman, A., Hu, H. and Zhao, Z., 2024, April. Is the Canary Dead? On the Effectiveness of Stack Canaries on Microcontroller Systems. In Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing (pp. 1350-1357).

Software Architectural Issues (Answers to Q2)

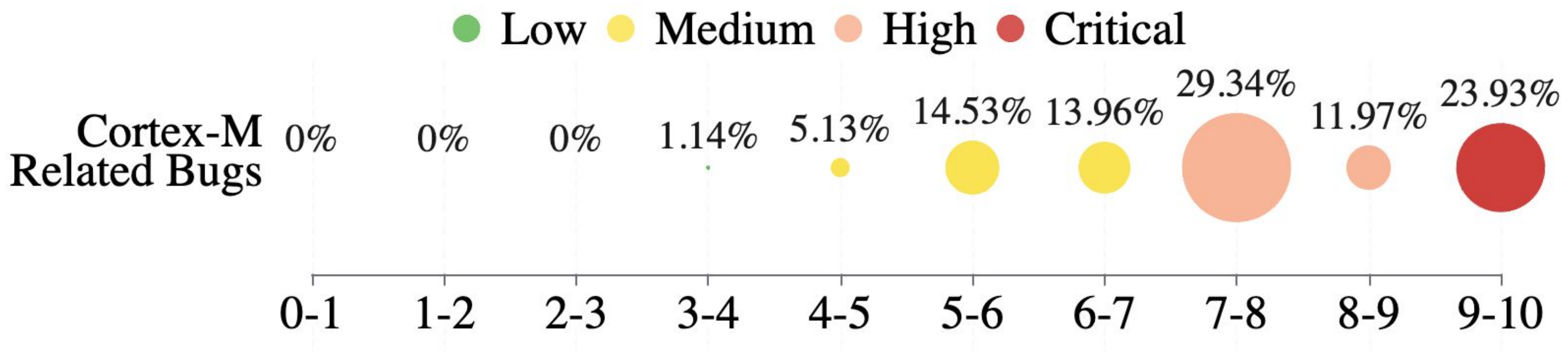
Insights

The real-world firmware samples in our dataset **barely** use the security features of Cortex-M.

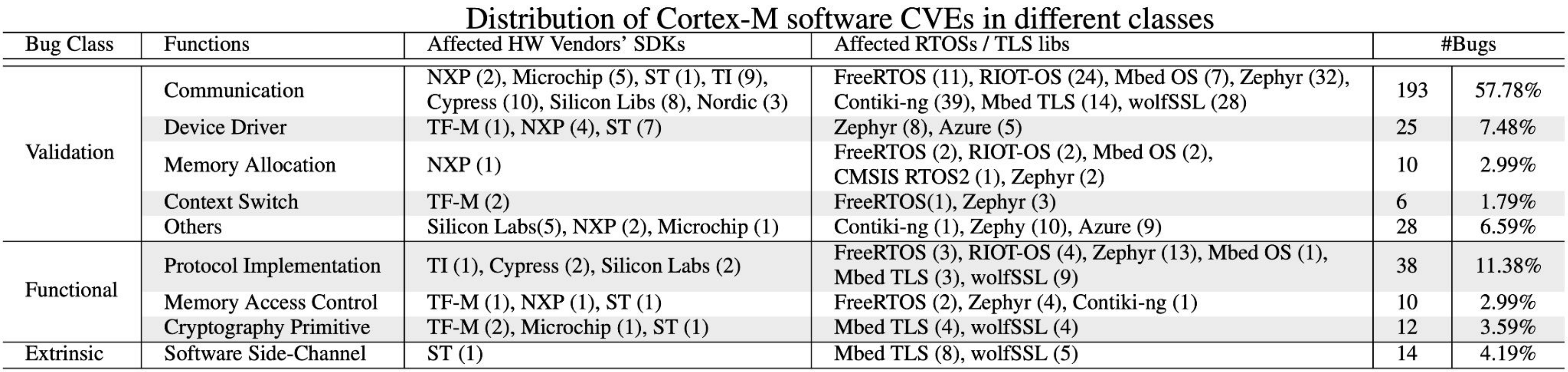
They largely **lack** the security mitigations that are widely deployed on modern systems.

Some software- and compiler-based mitigations, e.g., stack canaries, are less effective on MCU-based systems and should be **redesigned**.

Software Implementation Issues (Answers to Q3)



Software Implementation Issues (Answers to Q3)

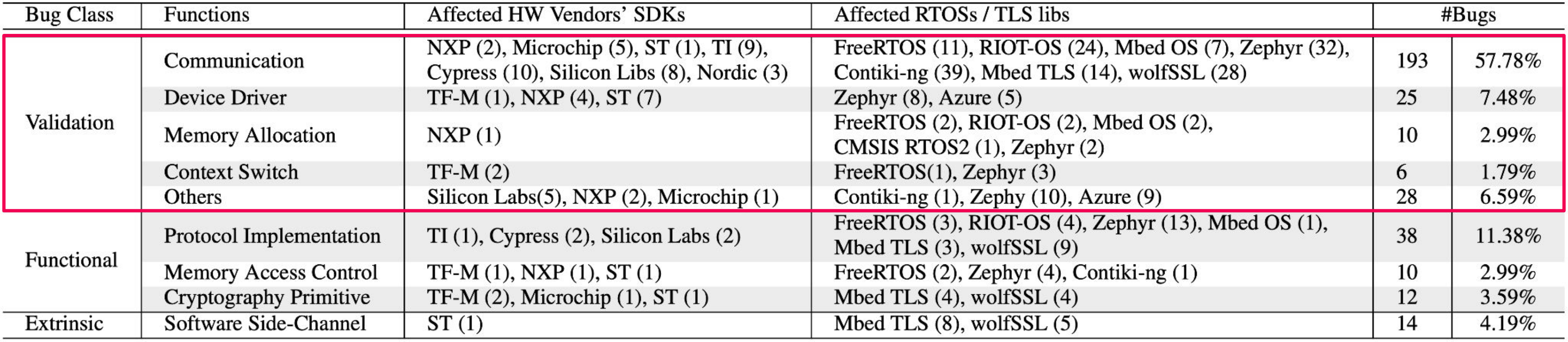


Distribution of Cortex-M software CVEs in different classes

Bug Class	Functions	Affected HW Vendors' SDKs	Affected RTOSs / TLS libs	#Bugs	
Validation	Communication	NXP (2), Microchip (5), ST (1), TI (9), Cypress (10), Silicon Libs (8), Nordic (3)	FreeRTOS (11), RIOT-OS (24), Mbed OS (7), Zephyr (32), Contiki-ng (39), Mbed TLS (14), wolfSSL (28)	193	57.78%
	Device Driver	TF-M (1), NXP (4), ST (7)	Zephyr (8), Azure (5)	25	7.48%
	Memory Allocation	NXP (1)	FreeRTOS (2), RIOT-OS (2), Mbed OS (2), CMSIS RTOS2 (1), Zephyr (2)	10	2.99%
	Context Switch	TF-M (2)	FreeRTOS(1), Zephyr (3)	6	1.79%
	Others	Silicon Labs(5), NXP (2), Microchip (1)	Contiki-ng (1), Zephy (10), Azure (9)	28	6.59%
Functional	Protocol Implementation	TI (1), Cypress (2), Silicon Labs (2)	FreeRTOS (3), RIOT-OS (4), Zephyr (13), Mbed OS (1), Mbed TLS (3), wolfSSL (9)	38	11.38%
	Memory Access Control	TF-M (1), NXP (1), ST (1)	FreeRTOS (2), Zephyr (4), Contiki-ng (1)	10	2.99%
	Cryptography Primitive	TF-M (2), Microchip (1), ST (1)	Mbed TLS (4), wolfSSL (4)	12	3.59%
Extrinsic	Software Side-Channel	ST (1)	Mbed TLS (8), wolfSSL (5)	14	4.19%

Software Implementation Issues (Answers to Q3)

Distribution of Cortex-M software CVEs in different classes

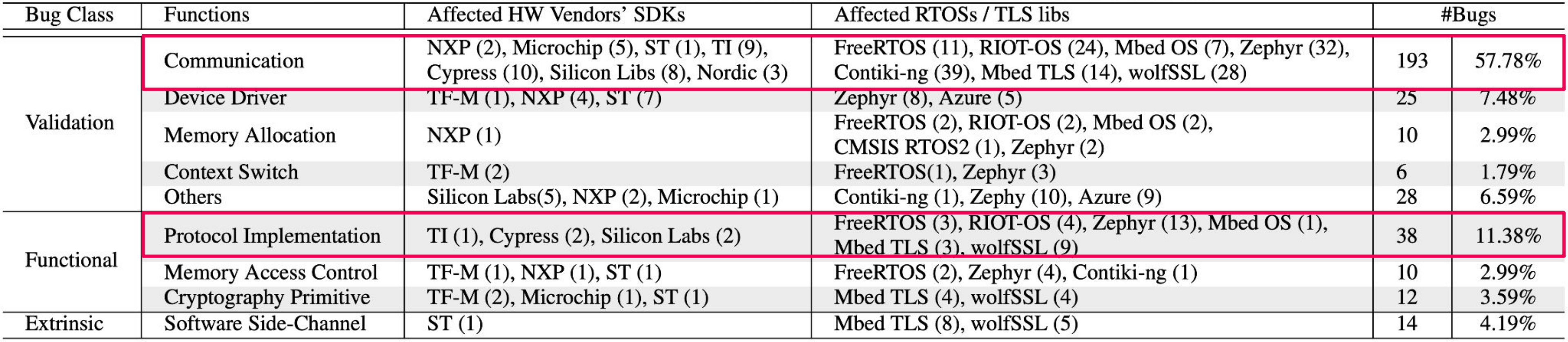


Bug Class	Functions	Affected HW Vendors' SDKs	Affected RTOSs / TLS libs	#Bugs	
Validation	Communication	NXP (2), Microchip (5), ST (1), TI (9), Cypress (10), Silicon Libs (8), Nordic (3)	FreeRTOS (11), RIOT-OS (24), Mbed OS (7), Zephyr (32), Contiki-ng (39), Mbed TLS (14), wolfSSL (28)	193	57.78%
	Device Driver	TF-M (1), NXP (4), ST (7)	Zephyr (8), Azure (5)	25	7.48%
	Memory Allocation	NXP (1)	FreeRTOS (2), RIOT-OS (2), Mbed OS (2), CMSIS RTOS2 (1), Zephyr (2)	10	2.99%
	Context Switch	TF-M (2)	FreeRTOS(1), Zephyr (3)	6	1.79%
	Others	Silicon Labs(5), NXP (2), Microchip (1)	Contiki-ng (1), Zephy (10), Azure (9)	28	6.59%
Functional	Protocol Implementation	TI (1), Cypress (2), Silicon Labs (2)	FreeRTOS (3), RIOT-OS (4), Zephyr (13), Mbed OS (1), Mbed TLS (3), wolfSSL (9)	38	11.38%
	Memory Access Control	TF-M (1), NXP (1), ST (1)	FreeRTOS (2), Zephyr (4), Contiki-ng (1)	10	2.99%
	Cryptography Primitive	TF-M (2), Microchip (1), ST (1)	Mbed TLS (4), wolfSSL (4)	12	3.59%
Extrinsic	Software Side-Channel	ST (1)	Mbed TLS (8), wolfSSL (5)	14	4.19%

- ❑ Validation bugs refer to bugs that mishandle or improperly validate input and output data. Examples are out-of-bounds read and write and improper parameter validation.
- ❑ 76.63% of all collected bugs belong to this category.

Software Implementation Issues (Answers to Q3)

Distribution of Cortex-M software CVEs in different classes



Bug Class	Functions	Affected HW Vendors' SDKs	Affected RTOSs / TLS libs	#Bugs	
Validation	Communication	NXP (2), Microchip (5), ST (1), TI (9), Cypress (10), Silicon Libs (8), Nordic (3)	FreeRTOS (11), RIOT-OS (24), Mbed OS (7), Zephyr (32), Contiki-ng (39), Mbed TLS (14), wolfSSL (28)	193	57.78%
	Device Driver	TF-M (1), NXP (4), ST (7)	Zephyr (8), Azure (5)	25	7.48%
	Memory Allocation	NXP (1)	FreeRTOS (2), RIOT-OS (2), Mbed OS (2), CMSIS RTOS2 (1), Zephyr (2)	10	2.99%
	Context Switch	TF-M (2)	FreeRTOS(1), Zephyr (3)	6	1.79%
	Others	Silicon Labs(5), NXP (2), Microchip (1)	Contiki-ng (1), Zephy (10), Azure (9)	28	6.59%
Functional	Protocol Implementation	TI (1), Cypress (2), Silicon Labs (2)	FreeRTOS (3), RIOT-OS (4), Zephyr (13), Mbed OS (1), Mbed TLS (3), wolfSSL (9)	38	11.38%
	Memory Access Control	TF-M (1), NXP (1), ST (1)	FreeRTOS (2), Zephyr (4), Contiki-ng (1)	10	2.99%
	Cryptography Primitive	TF-M (2), Microchip (1), ST (1)	Mbed TLS (4), wolfSSL (4)	12	3.59%
Extrinsic	Software Side-Channel	ST (1)	Mbed TLS (8), wolfSSL (5)	14	4.19%

- ❑ Validation bugs refer to bugs that mishandle or improperly validate input and output data. Examples are out-of-bounds read and write and improper parameter validation.
- ❑ 76.63% of all collected bugs belong to this category.
- ❑ Protocols (communication validation & implementation) introduce over a half of vulnerabilities

Software Implementation Issues (Answers to Q3)

Insights

Most Cortex-M based production systems are written in **memory-unsafe** languages, e.g., C, and they suffer from memory corruption vulnerabilities.

Microcontroller developers may **not realize** the absence of features like an MMU can pose greater risks than microprocessors.

Without privilege separation, **any** bug can be critical and compromise the entire system.

?

Can those limitations and issues be addressed?

!Security Research!

Security Research (Answers to Q4)

- 🟩 D01. Mitigating micro. attacks
- 🟩 D02. Secure cross-state communication
- 🟩 D03. Privilege separation
- 🟩 D04. Compartmentalization
- 🟩 D05. Virtualization
- 🟩 D06. Multi-world systems
- 🟩 D07. Stack and return address integrity
- 🟩 D08. Forward-edge CFI
- 🟩 D09. Compiler-based software diversity
- 🟩 D10. ASLR
- 🟩 D11. Formal verification
- 🟩 D12. Software-based XOM
- 🟩 D13. Secure multiprogramming with ...
- 🟩 D14. Software-based control-flow ...
- 🟩 D15 - D16. Firmware update
- 🟩 D17 - D20. Vulnerability discovery

🟩 Security Research

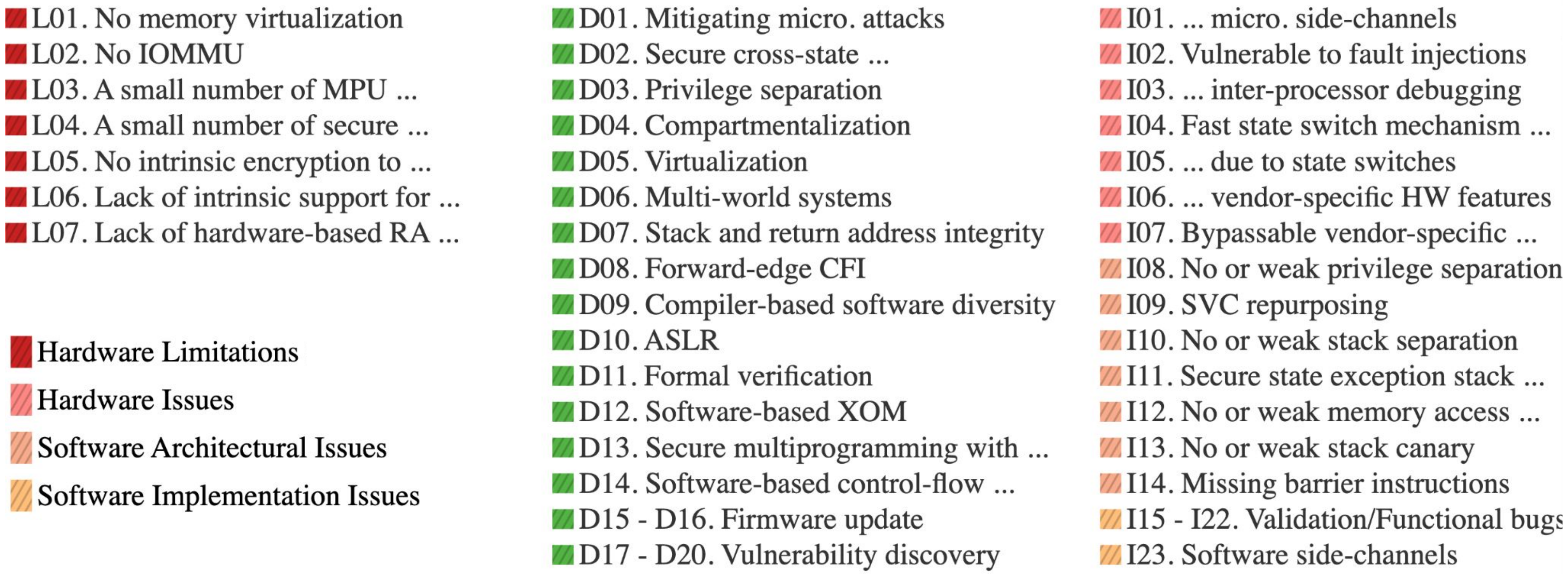
Security Research (Answers to Q4)

- ▨ L01. No memory virtualization
- ▨ L02. No IOMMU
- ▨ L03. A small number of MPU ...
- ▨ L04. A small number of secure ...
- ▨ L05. No intrinsic encryption to ...
- ▨ L06. Lack of intrinsic support for ...
- ▨ L07. Lack of hardware-based RA ...

- ▨ I01. ... micro. side-channels
- ▨ I02. Vulnerable to fault injections
- ▨ I03. ... inter-processor debugging
- ▨ I04. Fast state switch mechanism ...
- ▨ I05. ... due to state switches
- ▨ I06. ... vendor-specific HW features
- ▨ I07. Bypassable vendor-specific ...
- ▨ I08. No or weak privilege separation
- ▨ I09. SVC repurposing
- ▨ I10. No or weak stack separation
- ▨ I11. Secure state exception stack ...
- ▨ I12. No or weak memory access ...
- ▨ I13. No or weak stack canary
- ▨ I14. Missing barrier instructions
- ▨ I15 - I22. Validation/Functional bugs
- ▨ I23. Software side-channels

- ▨ Hardware Limitations
- ▨ Hardware Issues
- ▨ Software Architectural Issues
- ▨ Software Implementation Issues

Security Research (Answers to Q4)



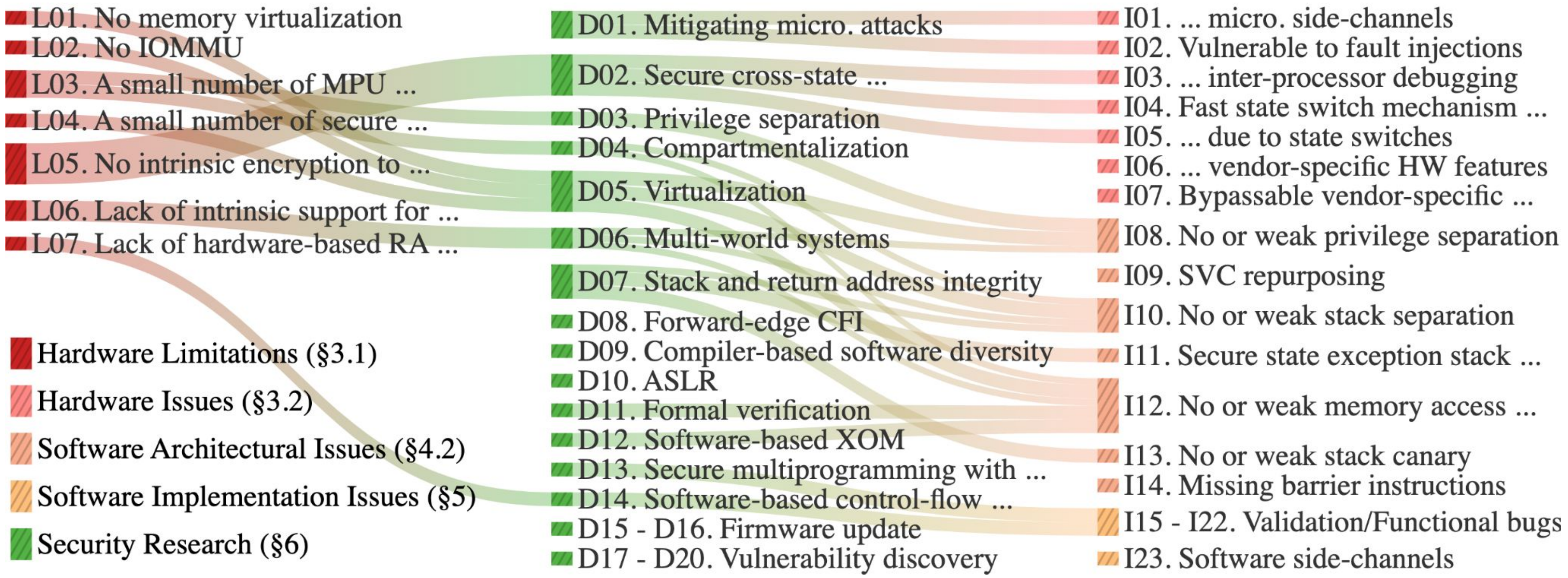
- 🟥 L01. No memory virtualization
- 🟥 L02. No IOMMU
- 🟥 L03. A small number of MPU ...
- 🟥 L04. A small number of secure ...
- 🟥 L05. No intrinsic encryption to ...
- 🟥 L06. Lack of intrinsic support for ...
- 🟥 L07. Lack of hardware-based RA ...

- 🟥 Hardware Limitations
- ▨ Hardware Issues
- 🟧 Software Architectural Issues
- 🟨 Software Implementation Issues

- 🟩 D01. Mitigating micro. attacks
- 🟩 D02. Secure cross-state ...
- 🟩 D03. Privilege separation
- 🟩 D04. Compartmentalization
- 🟩 D05. Virtualization
- 🟩 D06. Multi-world systems
- 🟩 D07. Stack and return address integrity
- 🟩 D08. Forward-edge CFI
- 🟩 D09. Compiler-based software diversity
- 🟩 D10. ASLR
- 🟩 D11. Formal verification
- 🟩 D12. Software-based XOM
- 🟩 D13. Secure multiprogramming with ...
- 🟩 D14. Software-based control-flow ...
- 🟩 D15 - D16. Firmware update
- 🟩 D17 - D20. Vulnerability discovery

- ▨ I01. ... micro. side-channels
- ▨ I02. Vulnerable to fault injections
- ▨ I03. ... inter-processor debugging
- ▨ I04. Fast state switch mechanism ...
- ▨ I05. ... due to state switches
- ▨ I06. ... vendor-specific HW features
- ▨ I07. Bypassable vendor-specific ...
- 🟧 I08. No or weak privilege separation
- 🟧 I09. SVC repurposing
- 🟧 I10. No or weak stack separation
- 🟧 I11. Secure state exception stack ...
- 🟧 I12. No or weak memory access ...
- 🟧 I13. No or weak stack canary
- 🟧 I14. Missing barrier instructions
- 🟨 I15 - I22. Validation/Functional bugs
- 🟨 I23. Software side-channels

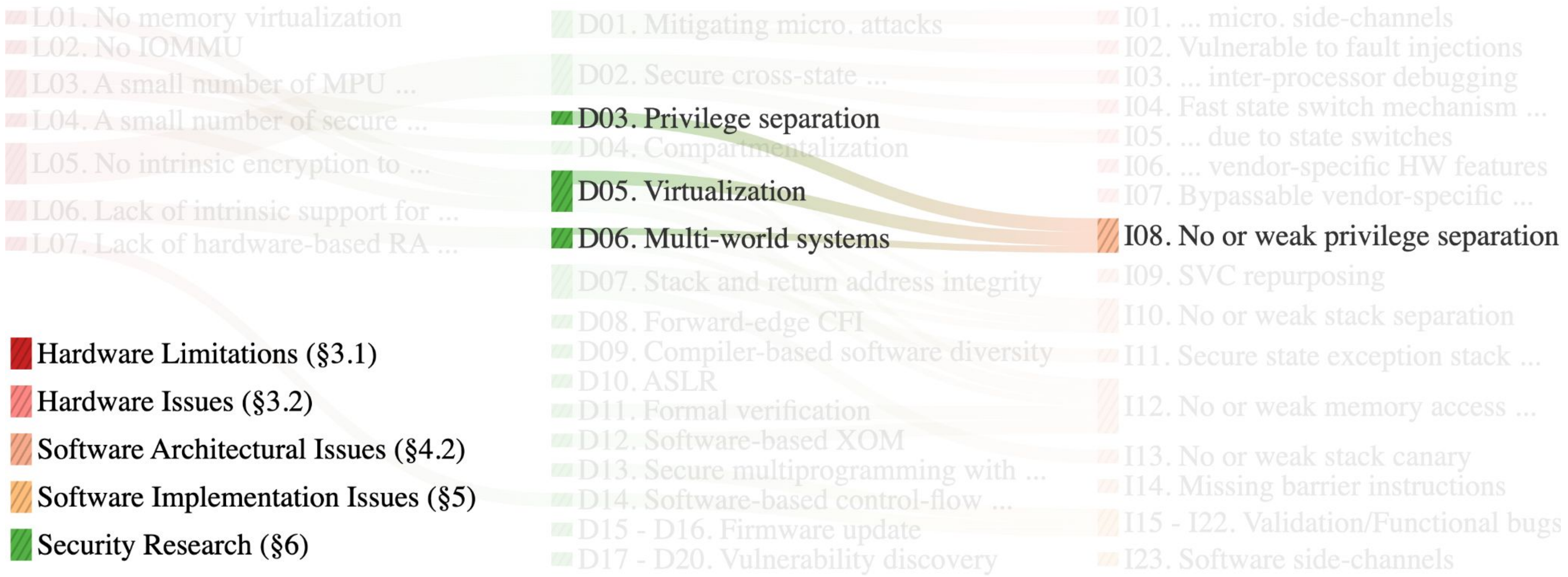
Security Research (Answers to Q4)


L01. No memory virtualization
L02. No IOMMU
L03. A small number of MPU ...
L04. A small number of secure ...
L05. No intrinsic encryption to ...
L06. Lack of intrinsic support for ...
L07. Lack of hardware-based RA ...
Hardware Limitations (§3.1)
Hardware Issues (§3.2)
Software Architectural Issues (§4.2)
Software Implementation Issues (§5)
Security Research (§6)
D01. Mitigating micro. attacks
D02. Secure cross-state ...
D03. Privilege separation
D04. Compartmentalization
D05. Virtualization
D06. Multi-world systems
D07. Stack and return address integrity
D08. Forward-edge CFI
D09. Compiler-based software diversity
D10. ASLR
D11. Formal verification
D12. Software-based XOM
D13. Secure multiprogramming with ...
D14. Software-based control-flow ...
D15 - D16. Firmware update
D17 - D20. Vulnerability discovery
I01. ... micro. side-channels
I02. Vulnerable to fault injections
I03. ... inter-processor debugging
I04. Fast state switch mechanism ...
I05. ... due to state switches
I06. ... vendor-specific HW features
I07. Bypassable vendor-specific ...
I08. No or weak privilege separation
I09. SVC repurposing
I10. No or weak stack separation
I11. Secure state exception stack ...
I12. No or weak memory access ...
I13. No or weak stack canary
I14. Missing barrier instructions
I15 - I22. Validation/Functional bugs
I23. Software side-channels

The connection indicate the issues a research direction attempts to address and the limitations it needs to overcome.

[1] https://github.com/CactiLab/code-SoK-Cortex-M/blob/main/4_relations_interactive_fig.html

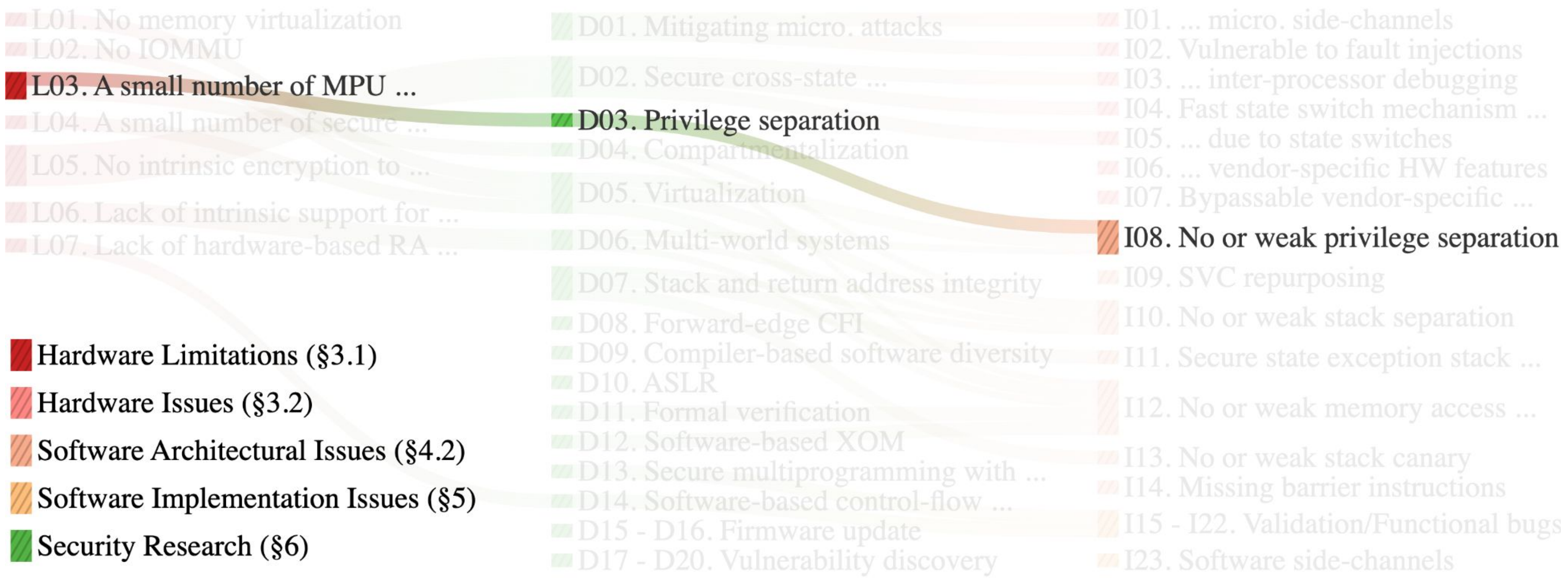
Security Research (Answers to Q4)


L01. No memory virtualization
L02. No IOMMU
L03. A small number of MPU ...
L04. A small number of secure ...
L05. No intrinsic encryption to ...
L06. Lack of intrinsic support for ...
L07. Lack of hardware-based RA ...
D01. Mitigating micro. attacks
D02. Secure cross-state ...
D03. Privilege separation
D04. Compartmentalization
D05. Virtualization
D06. Multi-world systems
D07. Stack and return address integrity
D08. Forward-edge CFI
D09. Compiler-based software diversity
D10. ASLR
D11. Formal verification
D12. Software-based XOM
D13. Secure multiprogramming with ...
D14. Software-based control-flow ...
D15 - D16. Firmware update
D17 - D20. Vulnerability discovery
I01. ... micro. side-channels
I02. Vulnerable to fault injections
I03. ... inter-processor debugging
I04. Fast state switch mechanism ...
I05. ... due to state switches
I06. ... vendor-specific HW features
I07. Bypassable vendor-specific ...
I08. No or weak privilege separation
I09. SVC repurposing
I10. No or weak stack separation
I11. Secure state exception stack ...
I12. No or weak memory access ...
I13. No or weak stack canary
I14. Missing barrier instructions
I15 - I22. Validation/Functional bugs
I23. Software side-channels
Hardware Limitations (§3.1)
Hardware Issues (§3.2)
Software Architectural Issues (§4.2)
Software Implementation Issues (§5)
Security Research (§6)

The connection indicate the issues a research direction attempts to address and the limitations it needs to overcome.

For instance, to address the issue of **no or weak privilege separation [I08]**, mitigation (**Privilege separation [D03]**, **Virtualization [D05]**, and **Multi-world systems [D06]**) have been proposed, and they overcome some limitations.

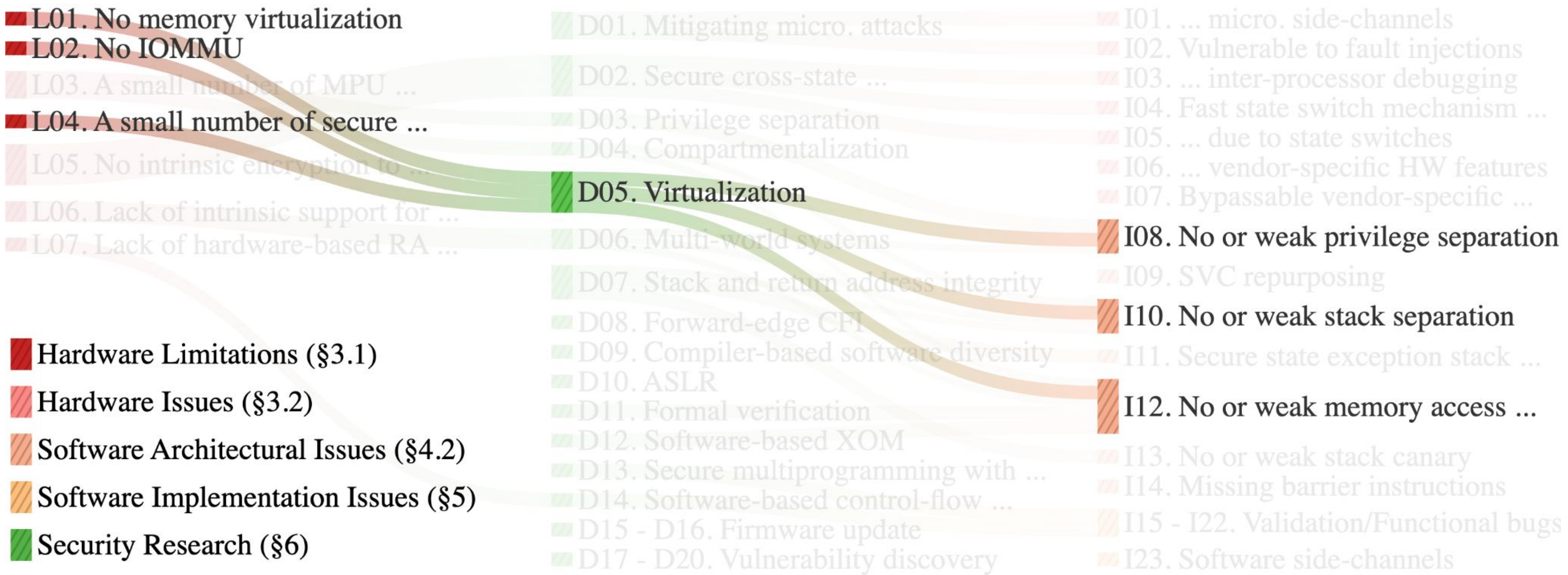
Security Research (Answers to Q4)


L01. No memory virtualization
L02. No IOMMU
L03. A small number of MPU ...
L04. A small number of secure ...
L05. No intrinsic encryption to ...
L06. Lack of intrinsic support for ...
L07. Lack of hardware-based RA ...
D01. Mitigating micro. attacks
D02. Secure cross-state ...
D03. Privilege separation
D04. Compartmentalization
D05. Virtualization
D06. Multi-world systems
D07. Stack and return address integrity
D08. Forward-edge CFI
D09. Compiler-based software diversity
D10. ASLR
D11. Formal verification
D12. Software-based XOM
D13. Secure multiprogramming with ...
D14. Software-based control-flow ...
D15 - D16. Firmware update
D17 - D20. Vulnerability discovery
I01. ... micro. side-channels
I02. Vulnerable to fault injections
I03. ... inter-processor debugging
I04. Fast state switch mechanism ...
I05. ... due to state switches
I06. ... vendor-specific HW features
I07. Bypassable vendor-specific ...
I08. No or weak privilege separation
I09. SVC repurposing
I10. No or weak stack separation
I11. Secure state exception stack ...
I12. No or weak memory access ...
I13. No or weak stack canary
I14. Missing barrier instructions
I15 - I22. Validation/Functional bugs
I23. Software side-channels
Hardware Limitations (§3.1)
Hardware Issues (§3.2)
Software Architectural Issues (§4.2)
Software Implementation Issues (§5)
Security Research (§6)

The connection indicate the issues a research direction attempts to address and the limitations it needs to overcome.

E.g., the privilege separation needs to overcome the limitation of limited size of configurable MPU regions.

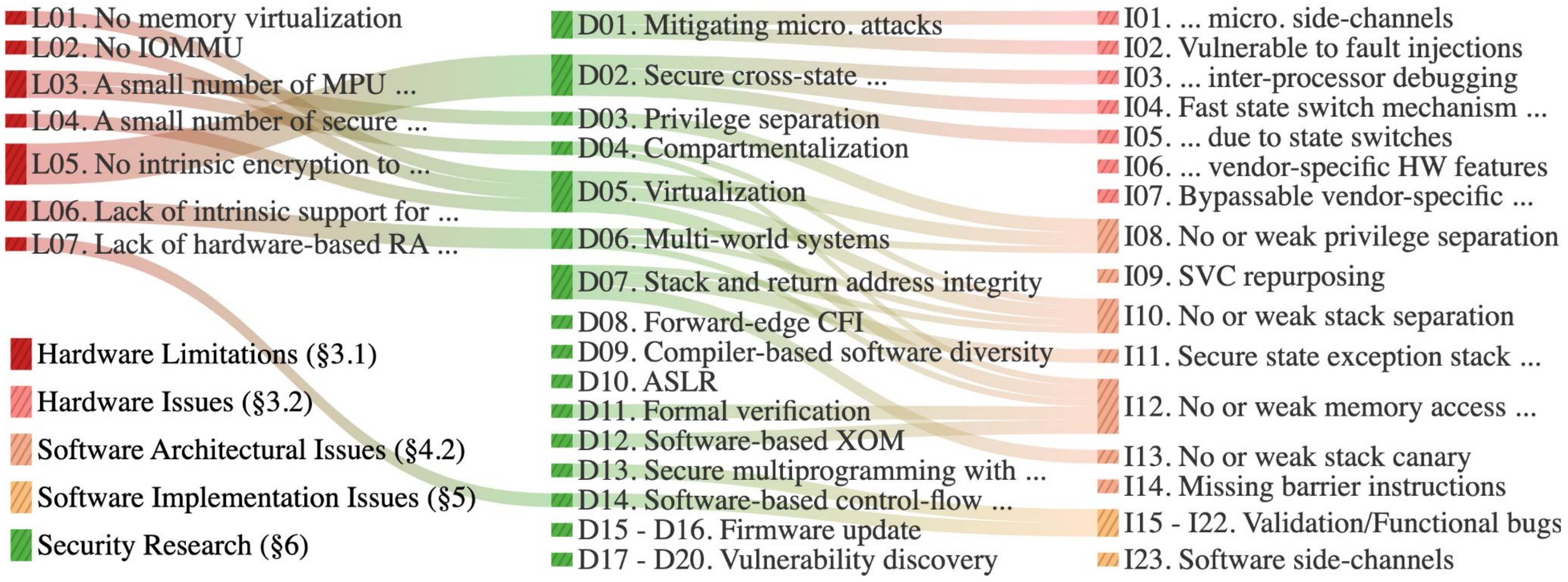
Security Research (Answers to Q4)


L01. No memory virtualization
L02. No IOMMU
L03. A small number of MPU ...
L04. A small number of secure ...
L05. No intrinsic encryption to ...
L06. Lack of intrinsic support for ...
L07. Lack of hardware-based RA ...
Hardware Limitations (§3.1)
Hardware Issues (§3.2)
Software Architectural Issues (§4.2)
Software Implementation Issues (§5)
Security Research (§6)
D01. Mitigating micro. attacks
D02. Secure cross-state ...
D03. Privilege separation
D04. Compartmentalization
D05. Virtualization
D06. Multi-world systems
D07. Stack and return address integrity
D08. Forward-edge CFI
D09. Compiler-based software diversity
D10. ASLR
D11. Formal verification
D12. Software-based XOM
D13. Secure multiprogramming with ...
D14. Software-based control-flow ...
D15 - D16. Firmware update
D17 - D20. Vulnerability discovery
I01. ... micro. side-channels
I02. Vulnerable to fault injections
I03. ... inter-processor debugging
I04. Fast state switch mechanism ...
I05. ... due to state switches
I06. ... vendor-specific HW features
I07. Bypassable vendor-specific ...
I08. No or weak privilege separation
I09. SVC repurposing
I10. No or weak stack separation
I11. Secure state exception stack ...
I12. No or weak memory access ...
I13. No or weak stack canary
I14. Missing barrier instructions
I15 - I22. Validation/Functional bugs
I23. Software side-channels

The connection indicate the issues a research direction attempts to address and the limitations it needs to overcome.

E.g., the virtualization needs to consider how to virtualize the memory without the memory management unit (MMU). In addition, this defense can address more than one issues.

Security Research (Answers to Q4)



L01. No memory virtualization
L02. No IOMMU
L03. A small number of MPU ...
L04. A small number of secure ...
L05. No intrinsic encryption to ...
L06. Lack of intrinsic support for ...
L07. Lack of hardware-based RA ...
Hardware Limitations (§3.1)
Hardware Issues (§3.2)
Software Architectural Issues (§4.2)
Software Implementation Issues (§5)
Security Research (§6)
D01. Mitigating micro. attacks
D02. Secure cross-state ...
D03. Privilege separation
D04. Compartmentalization
D05. Virtualization
D06. Multi-world systems
D07. Stack and return address integrity
D08. Forward-edge CFI
D09. Compiler-based software diversity
D10. ASLR
D11. Formal verification
D12. Software-based XOM
D13. Secure multiprogramming with ...
D14. Software-based control-flow ...
D15 - D16. Firmware update
D17 - D20. Vulnerability discovery
I01. ... micro. side-channels
I02. Vulnerable to fault injections
I03. ... inter-processor debugging
I04. Fast state switch mechanism ...
I05. ... due to state switches
I06. ... vendor-specific HW features
I07. Bypassable vendor-specific ...
I08. No or weak privilege separation
I09. SVC repurposing
I10. No or weak stack separation
I11. Secure state exception stack ...
I12. No or weak memory access ...
I13. No or weak stack canary
I14. Missing barrier instructions
I15 - I22. Validation/Functional bugs
I23. Software side-channels

[1] https://github.com/CactiLab/code-SoK-Cortex-M/blob/main/4_relations_interactive_fig.html

Security Research (Answers to Q4)

Insights

The research shifts the exact **same** defenses from microprocessor-based systems on Cortex-M systems, e.g., enforcing isolation and confinement, stack integrity, and control flow integrity,

The research develops solutions **intrinsically** linked to the MCU characteristics, e.g., cross-state communication.

A **gap** between industrial implementation and academic security research.

?

What should we do next?

Recommendations and Future Directions

- ➔ Recommendations to research community
  - ◆ Explore the pros and cons of hardware features for security
    - E.g., fast state switch for TrustZone-M [1]
    - MTB for control-flow violation detection [2]
  - ◆ Explore diverse IoT attack models and scenarios to identify new research problems and challenges
  - ◆ Investigate how to facilitate the practical adoption of academic research results

[1] Ma, Z., Tan, X., Ziarek, L., Zhang, N., Hu, H. and Zhao, Z., 2023, July. Return-to-Non-Secure Vulnerabilities on ARM Cortex-M TrustZone: Attack and Defense. In 2023 60th ACM/IEEE Design Automation Conference (DAC) (pp. 1-6). IEEE.
[2] Tan, X. and Zhao, Z., 2023, November. Sherloc: Secure and holistic control-flow violation detection on embedded systems. In Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (pp. 1332-1346).

Recommendations and Future Directions

- Recommendations to developers
  - Secure the protocol implementations
  - Implement privilege separation or employ RTOSs with distinct privilege levels
  - (Partially) Transit into memory-safe languages

Takeaways

➔ Cortex-M architecture offers weaker memory management interfaces than popular microprocessors, creating challenges to enforce memory isolation and security

➔ The streamlined design of Cortex-M features potentially introduces new vulnerabilities

➔ A gap between real-world implementation and security research

➔ Open-source resources: https://github.com/CactiLab/SoK-Cortex-M
- Cortex-M hardware feature test suits
- Firmware database and analysis tool
- CVE details and classification

Thank You!