RContainer: A Secure Container Architecture through Extending ARM CCA Hardware Primitives

Qihang Zhou[1], Wenzhuo Cao[1,2], Xiaoqi Jia[1,2,✉], Peng Liu[3],
Shengzhi Zhang[4], **Jiayun Chen**[1,2], Shaowen Xu[1,2], Zhenyu Song[1]

[1]Institute of Information Engineering,CAS, [2]University of Chinese Academy of Sciences,
[3]Penn State University, [4]Boston University

中国科学院信息工程研究所
INSTITUTE OF INFORMATION ENGINEERING,CAS
中国科学院大学
University of Chinese Academy of Sciences
Penn
UNIVERSITY of PENNSYLVANIA
BOSTON UNIVERSITY

Problem Statement

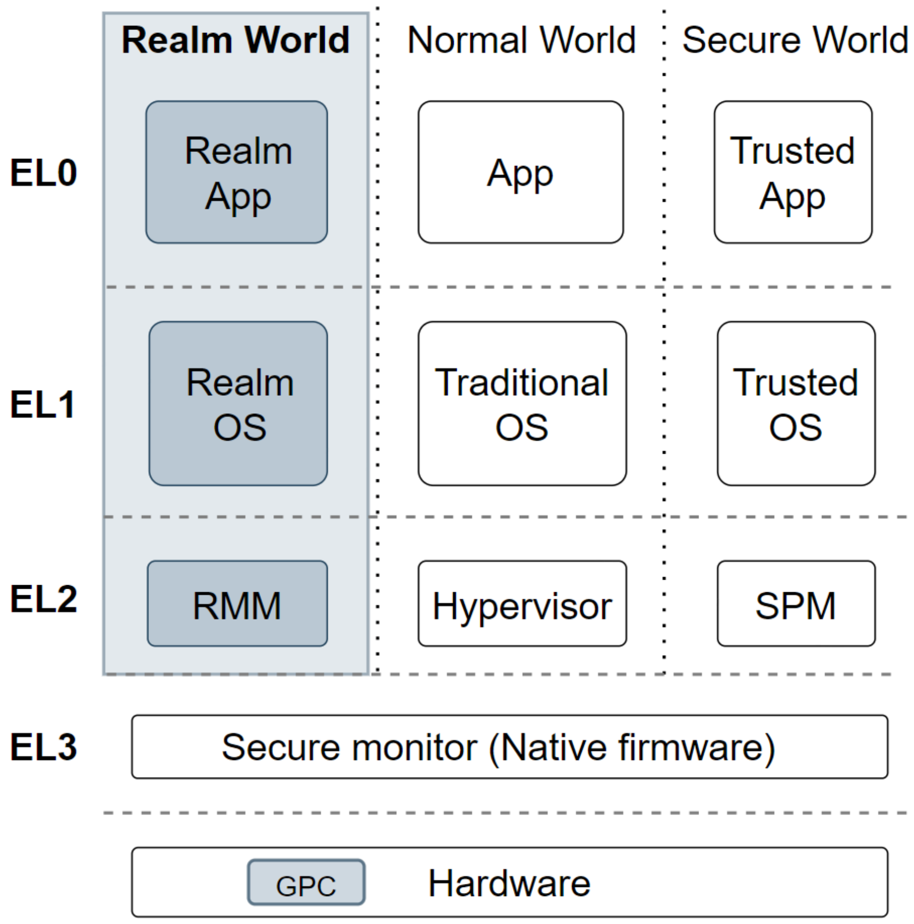
- The issues in container security
  - Weak isolation
  - Heavy overhead
  - Large TCB in the highest privilege

Security Insight

- Isolation between containers in both userspace and kernel space
- Minimizing the highest-privilege code
- Scalable security features

ARM Confidential Compute Architecture (CCA)

- Confidential computing introduced in ARMv9-A
- Four physical address spaces (PAS):
  - Normal PAS=>Normal World
  - Secure PAS =>Secure World
  - **Realm PAS =>Realm World** (New added in CCA)
  - Root PAS =>Root World (EL3)
- Granule Protection Check (GPC)
  - Granule Protection Table (GPT)
  - GPTBR_EL3



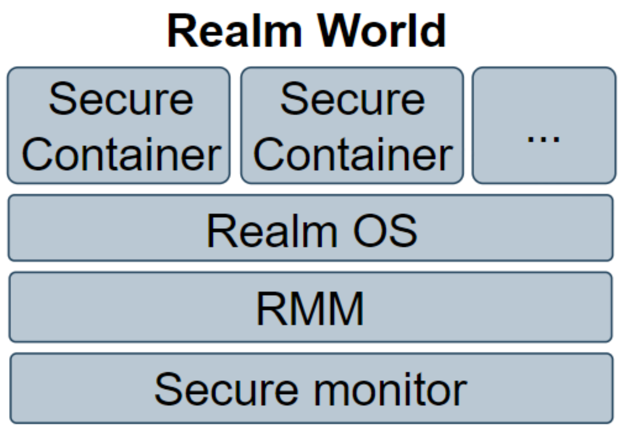
Granule Protection Table

- Fine-grained memory protection by defining access permissions for physical memory granules
- Maintained by firmware in EL3
- The GPT check occurs after the MMU check, and its result takes precedence over the MMU

Challenge

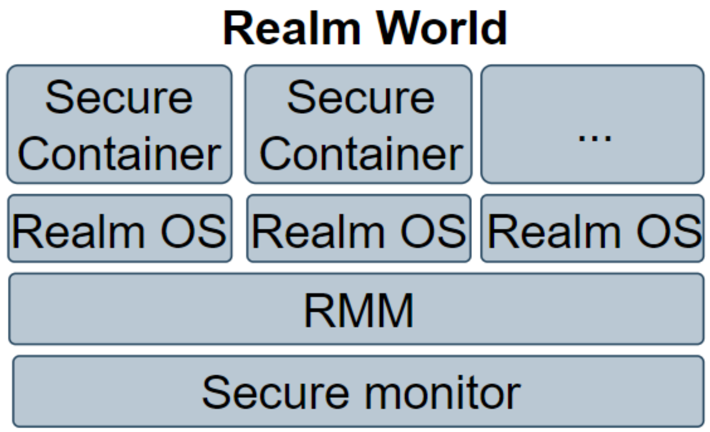
- **C1**: Containers are not really suitable for deployment in Realm World

  - (1) Multiple containers in one realm OS



Sharing OS leads to weak isolation

  - (2) One container in one realm OS

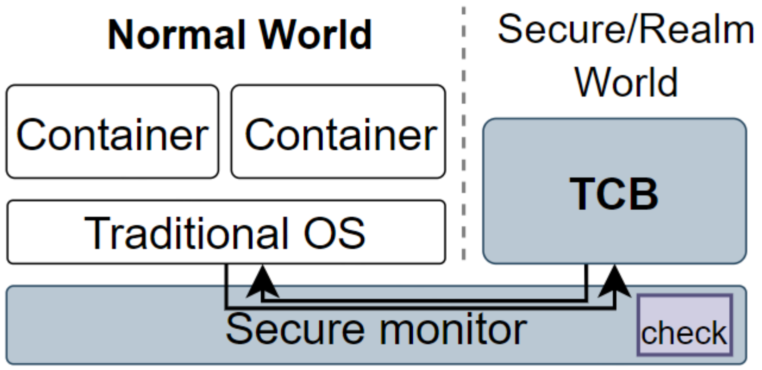


Mirroring hypervisor-based solutions leads to heavy overhead and large TCB

Challenge

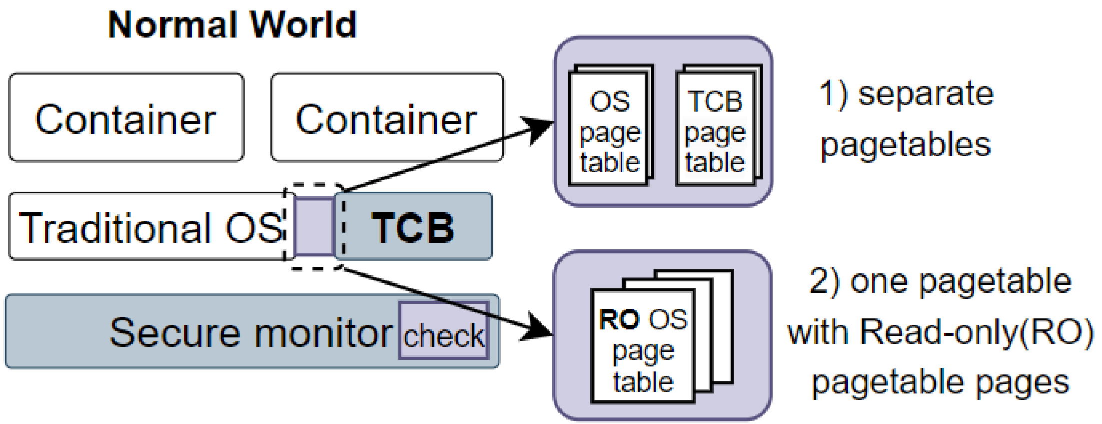
- **C2**: How to achieve tamper-proof protection of the TCB when only a small portion of the TCB is running with the highest privilege

- (1) Deploy TCB in Secure or Realm World



Frequent cross-world interaction and large TCB

- (2) Deploy TCB in Normal World by pagetable control



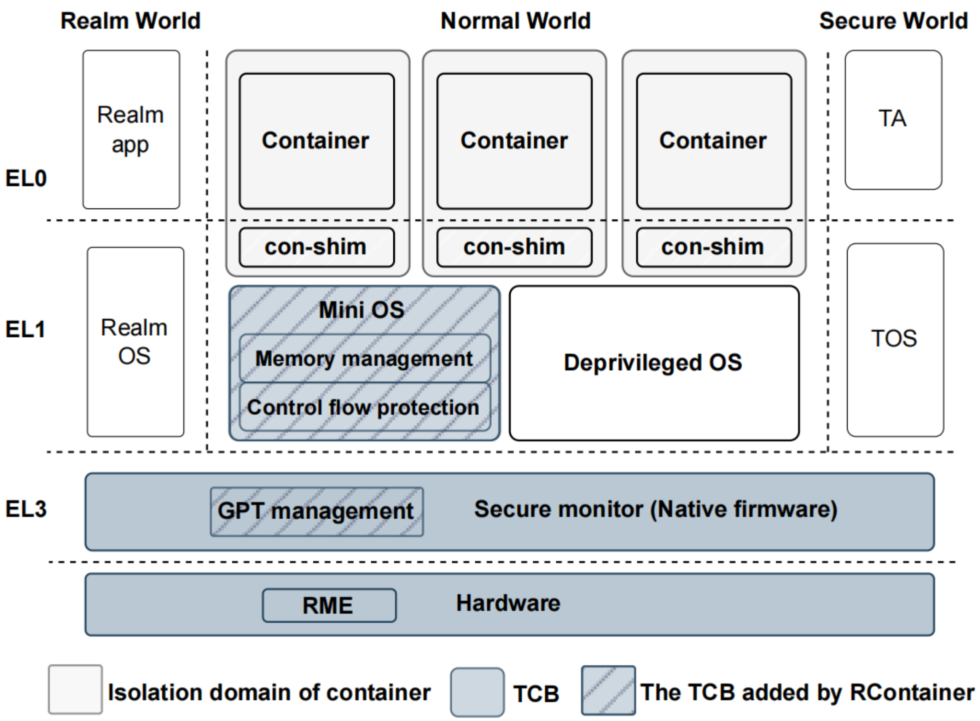
Frequent flush TLB

Threat Model

- System is initially benign but may be compromised after system boot
- Container, OS, hypervisor, SPM, RMM in Normal/Secure/Realm World may be compromised
- Physical/Side-channel/denial-of-service attacks are out of scope
  - Partial DoS can be considered, e.g., memory-related DoS

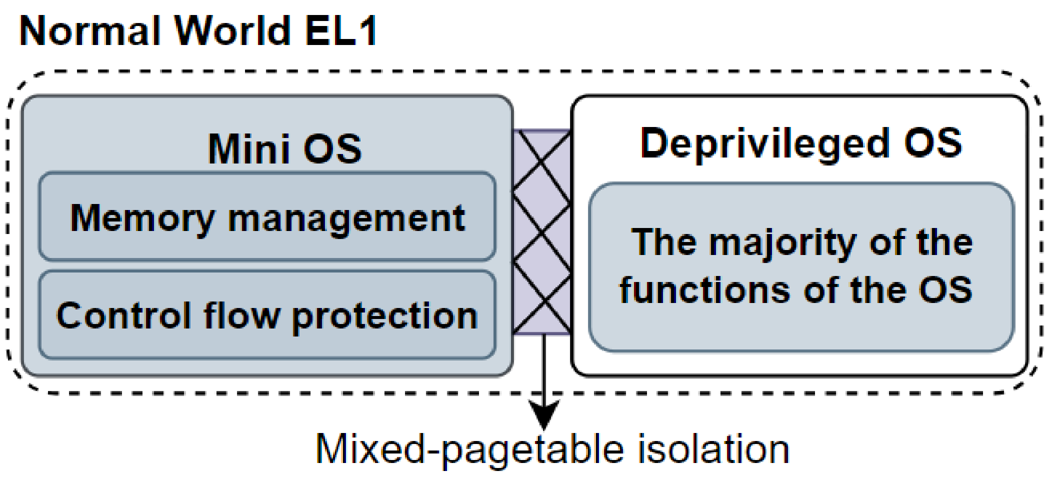
RContainer Architecture

- New secure container architecure protecting containers on OS while enforcing strong isolation among containers with minimal TCB
  - A *mini-OS* in EL1 to deprivilege OS
  - Shim-style isolation (multiple *con-shims)* to limit the impact of containers on the kernel



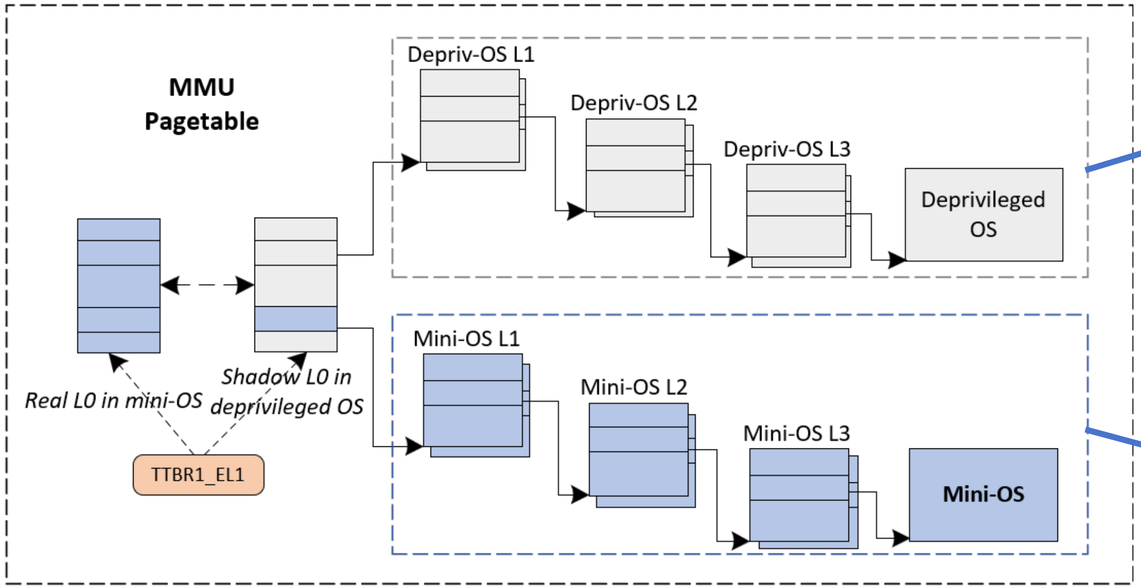
Mini-OS

- A compact and basic OS running in EL1 alongside the deprivileged OS
  - Mixed-pagetable for tamper-proof protection
  - Memory management and control flow protection



Mixed-pagetable

- Same MMU pagetable but different GPTs: Priv-GPT and OS-GPT



OS-GPT for Deprivileged OS

Components	*Address*	*Attributes*	*GPI*
Deprivileged OS-related	L1 PTP	Normal PAS	1001
	L2 PTP	Normal PAS	1001
	L3 PTP	Normal PAS	1001
	Page	Normal PAS	1001
Mini-OS-related	L1 PTP	No-access PAS	0000
	L2 PTP	No-access PAS	0000
	L3 PTP	No-access PAS	0000
	Page	No-access PAS	0000

Priv-GPT for Mini-OS

Components	*Address*	*Attributes*	*GPI*
Deprivileged OS-related	L1 PTP	Normal PAS	1001
	L2 PTP	Normal PAS	1001
	L3 PTP	Normal PAS	1001
	Page	Normal PAS	1001
Mini-OS-related	L1 PTP	Normal PAS	1001
	L2 PTP	Normal PAS	1001
	L3 PTP	Normal PAS	1001
	Page	Normal PAS	1001

Security Capabilities of Mini-OS

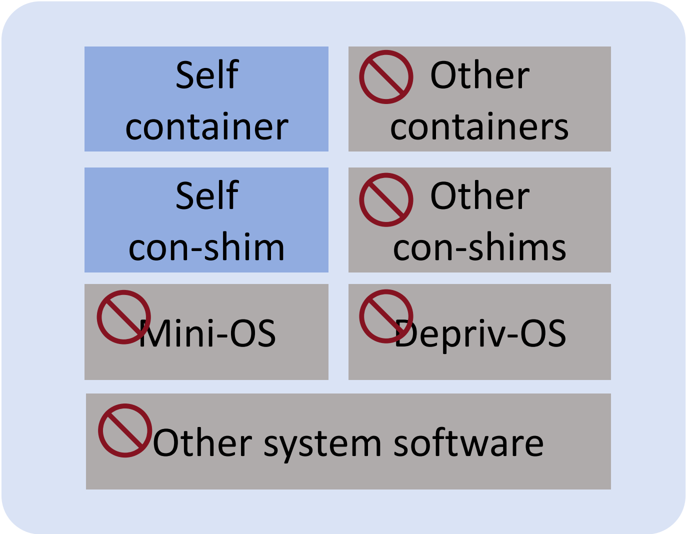
- Memory management
  - Maintenance of GPTs at the software level
  - Lightweight memory allocator
- Control flow protection
  - Exception interposing
  - Responsible for switching between OS and different containers

Shim-style Isolation

- Isolation between containers in both userspace and kernel space
  - Observation:
    - While most attacks originate in the control plane, they ultimately impact the data plane
    - The data plane requires stronger isolation for containers
  - Containers are instantiated within the kernel's data plane through multiple com-shims
    - Kernel boundary points, e.g., system call entry/exit point
    - Container-specific private data structures, e.g., task_struct
    - Shared global variables, e.g., nr_files

Shim-style Isolation

- Each com-shim has a separate shim-GPT
  - Each con-shim/container is limited to accessing only its own memory



Memory access permission in one shim-GPT

Container Lifecycle Protection

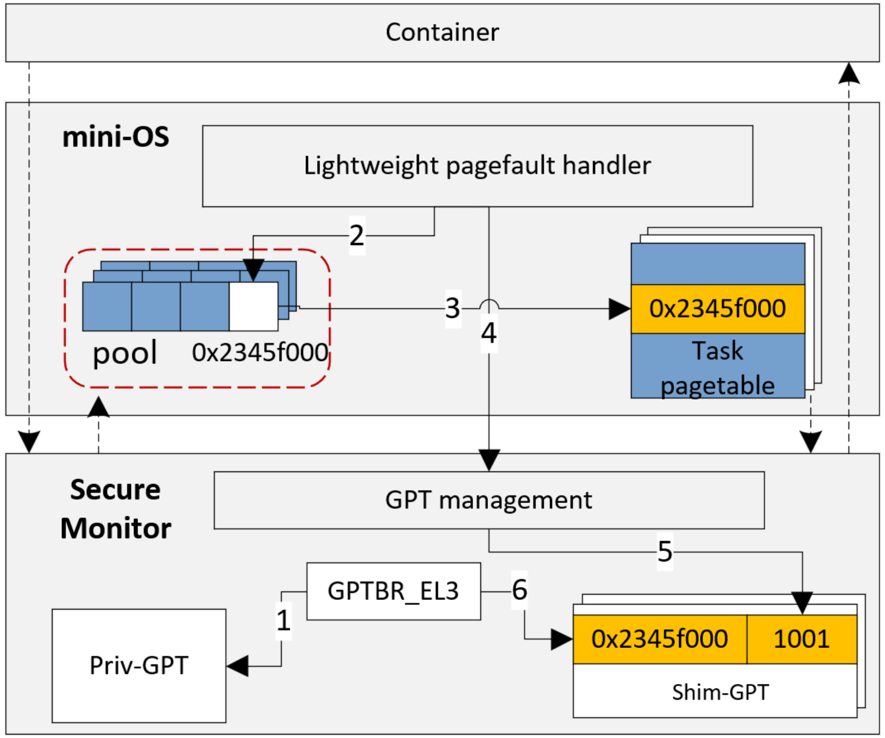
- Boot integrity
  - The deprivileged OS is loaded and measured by the EL3 secure monitor, then boots normally until launching the mini-OS
  - The mini-OS allocates memory for the con-shim and records the system call stack, shared memory, and private data
    - Create a shim-GPT and set these memory to be inaccessible within the OS-GPT

Container Lifecycle Protection

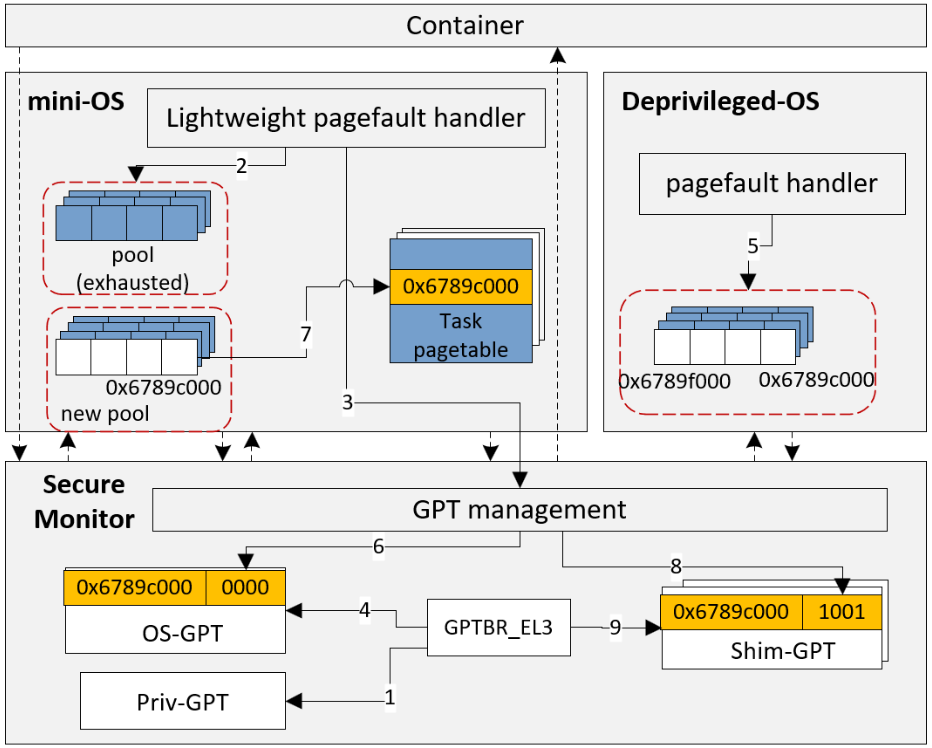
- Task
  - When creating tasks, mini-OS validates and logs the addresses of the new task structure and pagetable
  - When terminating tasks, mini-OS removes the task structure from the container's con-shim and clears the corresponding physical memory

Container Lifecycle Protection

- Memory—pagefault handling



Fast pagefault workflow



Slow pagefault workflow

Container Lifecycle Protection

- I/O
  - Disk I/O relies on encryption and decryption
  - A global SMMU-GPT for the deprivileged OS defaults all memory attributes to No-access, preventing arbitrary DMA memory access
  - Network I/O relies on secure network transmission protocols
- IPC
  - Shared memory is allocated and tracked by the mini-OS
  - File description related IPC is encrypted by the mini-OS

Implementation

RContainer Call	Description
rc_create_shim	Create new con-shim for a container
rc_destroy_shim	Destroy con-shim of a container
rc_create_container	Create a new container
rc_destroy_container	Destroy a container
rc_malloc_mm	Allocate memory for container/con-shim
rc_set_pte	Update PTE of a process/thread in container
rc_copy_page	Copy page to a container
rc_set_vma	Update vma of a process/thread in container
rc_set_iopte	Update IO PTE of IO device
rc_ipc_in	Handle ipc within a container
rc_ipc_out	Handle ipc between containers
rc_task_clone	Run a new process/thread in a container
rc_task_exec	Run program in a new address space in a container
rc_task_exit	Exit a process/thread in a container
rc_switch_to_depriv	Switch contexts to Deprivileged OS
rc_switch_to_miniOS	Switch contexts to mini-OS

- FVP prototype for security evaluation
  - ARM64v9.4-A Fixed Virtual Platform
  - Linux 6.2-rc2, Trusted Firmware-A v2.8.0, Docker-1.5
- Hardware prototype for performance evaluation
  - Firefly-RK3399 ARMv8 SoC development board
  - Linux-firefly-4.4.149, Trusted Firmware-A-1.3, Docker 25.0.0-beta.1

Security Evaluation

- Simulated and evaluated 30 CVEs
- Most attacks occur at runtime

CVE-*	Description[1]
2024-21626	Internal file descriptor leak in runc
2022-23222	Pointer arithmetic availability via *_OR_NULL pointer
2021-32606	User-after-free in isotp_setsockopt in net/can/isotp.c
2021-28972	User-tolerable buffer overflow during dev name entry
2020-14386	Kernel memory corruption due to arithmetic flaw
2020-8835	Out-of-bound access due to unrestricted register bound
2019-14271	Code injection occurs when the nsswitch loads a library
2019-10144	Do not isolate containers' processes when 'rkt enter'
2019-5736	Mishandling of file descriptor in /proc/self/exe
2018-18955	Improper handling of nested user namespace in write
2018-15664	Improper archive operations on a frozen filesystem
2018-15514	Unverify the validity of the deserialized .NET objects
2017-1000112	Memory corruption from UFO/non-UFO path switch
2017-7308	Improperly validation of certain block-size data
2016-9962	Improper execution to file-descriptors
2016-7117	Use-after-free in __sys_recvmmsg in net/socket.c
2016-5195	Race condition in mm/gup.c for handling CoW
2016-3697	Improper treats a numeric UID as username
2016-1582	Improper rights when switching container privilege
2016-1581	Improper permissions for ZFS.img when loop setup
2016-1576	Improper restricted mount namespace
2015-3630	Use weak permission for /proc/ operation
2015-3629	Unverified symlink when respawning a container
2015-3627	Open unverified file descriptor before chroot
2015-1335	Improper directory traversal operation in lxc-start
2014-9357	Improper handling of untrusted archive extraction
2014-6407	Symlink and hardlink when pulling docker images
2013-6441	Use read-write permissions when mounting /sbin/init
2010-4258	Improper handling of KERNEL_DS get_fs value
2010-2959	Integer overflow to function pointer overwrite

Native ATF Analysis

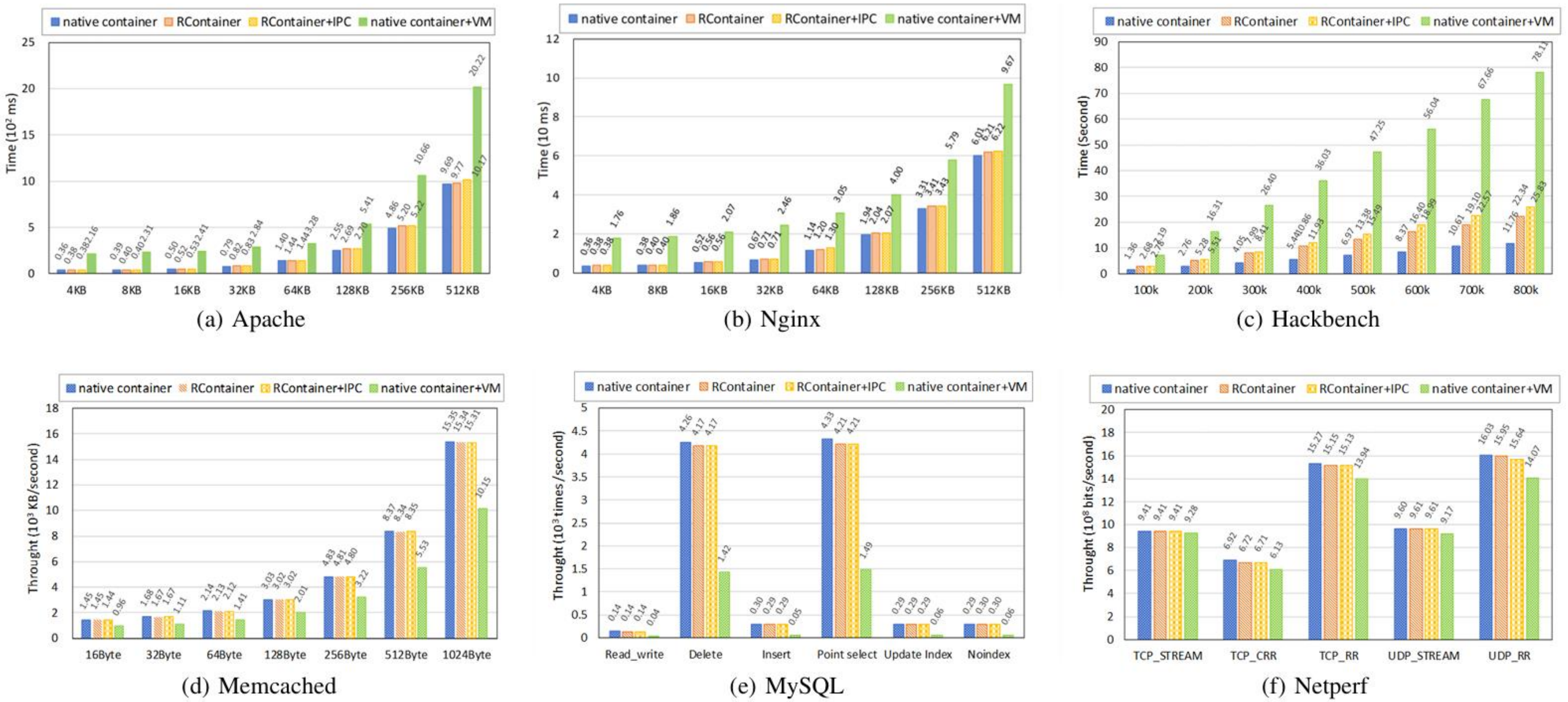
Function	SLoC
Platform bootup	218,909 (51.92%)
TrustZone support	17,460 (4.14%)
Realm World support	17,408 (4.13%)
Normal runtime support	11,387 (2.70%)
multi-Platforms/drivers	156,457 (37.11%)
Total	421,621

- Security functions should belong to runtime code
- The runtime code proportion in the native ATF is relatively small
  - About **2.7%** (11k SLoC/421k SLoC)

TCB Complexity

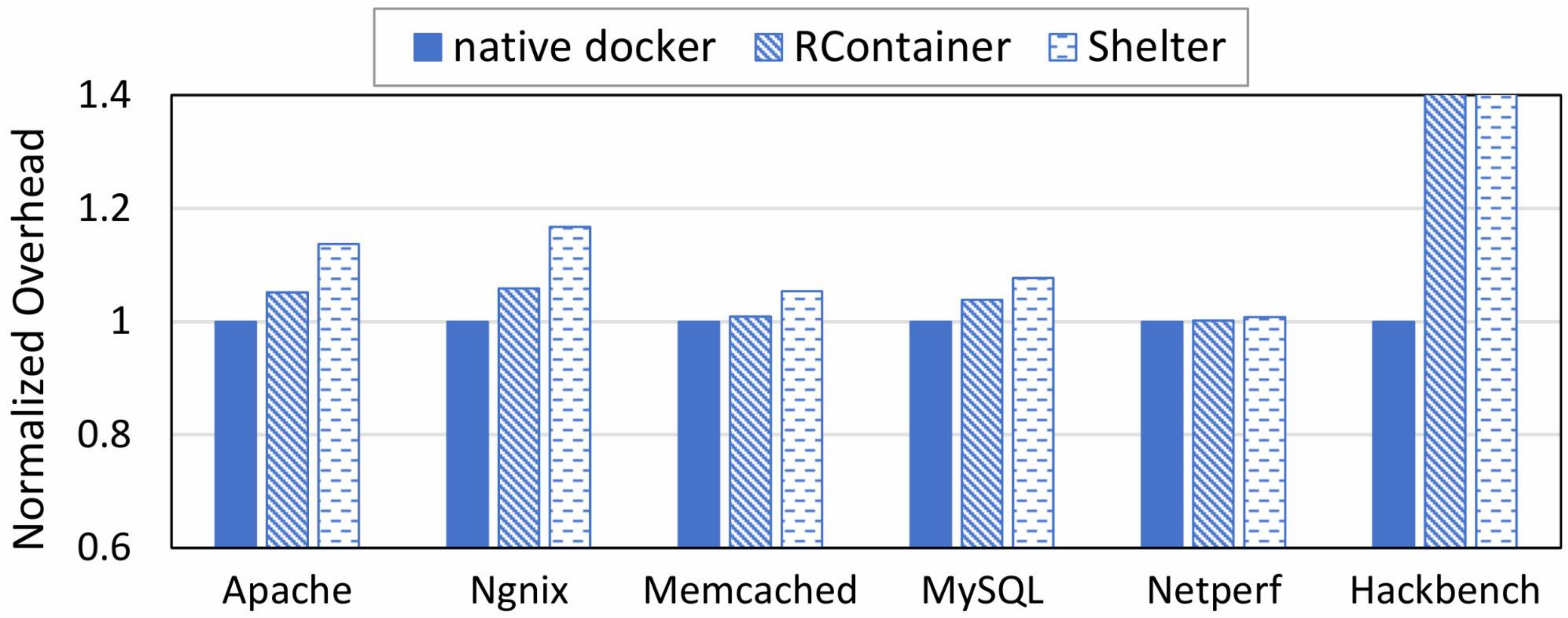
- RContainer introduces an additional 2,647 SLoC of TCB
  - 133 SLoC in EL0
  - 2,384 SLoC in EL1
  - 130 SLoC in EL3 (ATF)
- TCB in EL3 comparison with Shelter
  - ***130 SLoC (basically stable) vs 2k SLoC (continuously growing)***
    - Even with new security features, RContainer won't greatly increase EL3's runtime TCB

Application Workloads



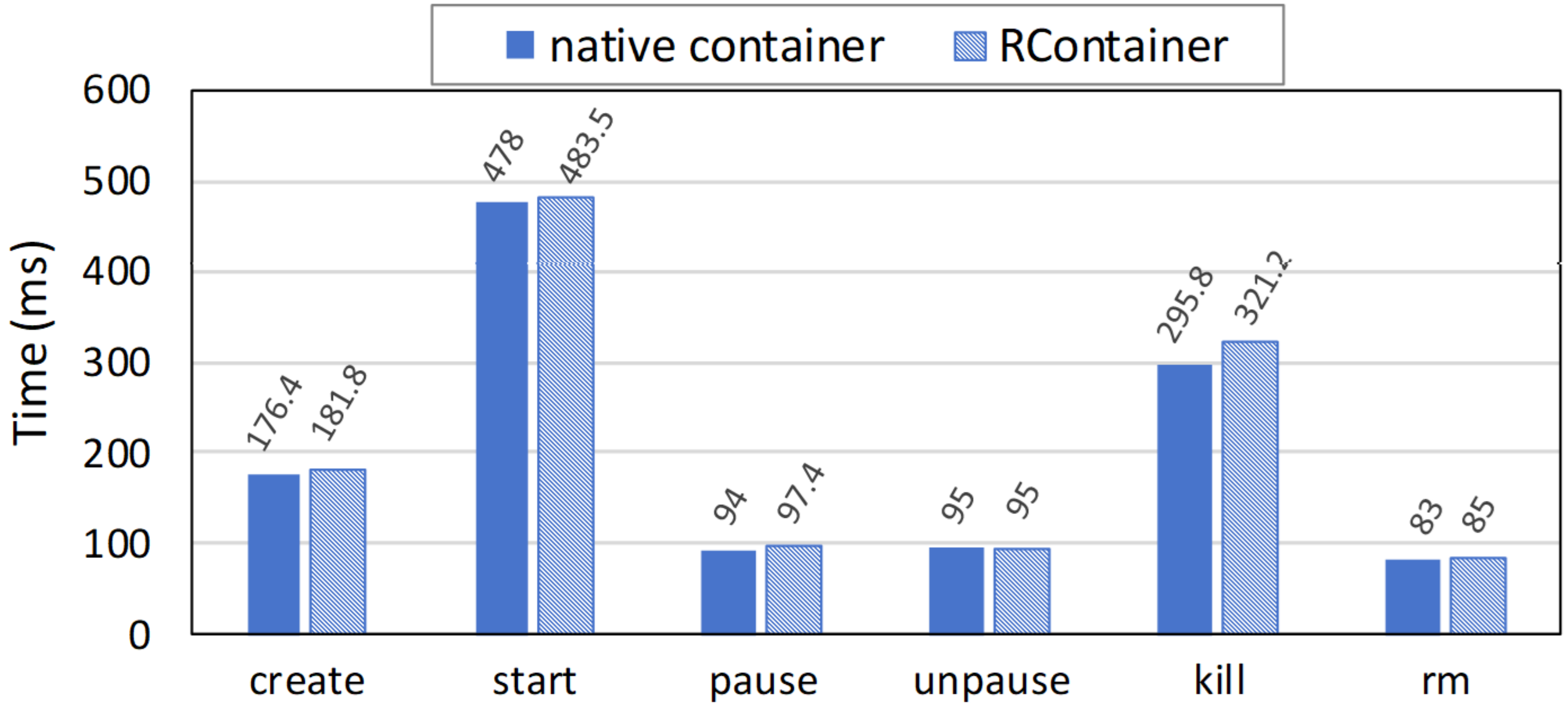
- < 10% overhead on real-world application workloads
- Much better than virtualization solutions
- Overhead on Hackbench is the worst in RContainer

Performance Comparison with Shelter



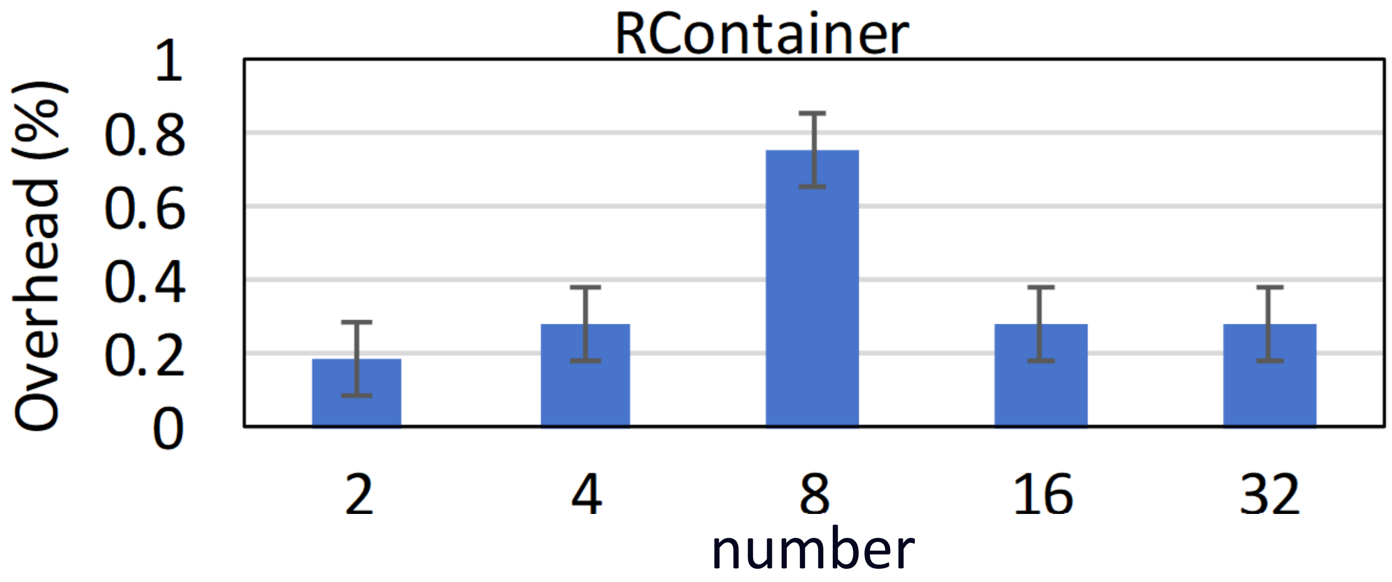
- The average overhead in RContainer is reduced by **5.7%** compared to Shelter

Container Lifecycle Cost



- < 10% overhead on busybox:1.36.1-glibc

Concurrent Overhead



- < 1% overhead on kernel build (Linux-4.19.309) with allnoconfig

Conclusion

- A new secure container architecture via extending ARM CCA
  - Protect containers on untrusted OS
  - Enforce strong isolation among containers both in userspace and kernel space
- Lower performance overhead without container modification
- Minimal TCB in highest privilege/exception level

THANKS!

Q&A

Contact us: zhouqihang@iie.ac.cn

chenjiayun@iie.ac.cn

中国科学院信息工程研究所
INSTITUTE OF INFORMATION ENGINEERING,CAS

中国科学院大学
University of Chinese Academy of Sciences

Penn
UNIVERSITY of PENNSYLVANIA

BOSTON UNIVERSITY